Représentation des connaissances sémantiques lexicales de la Théorie Sens-Texte : conceptualisation, représentation, et opérationnalisation des définitions lexicographiques

Maxime Lefrançois

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS
École Doctorale des Sciences et Technologies de l'Information et de la Communication

THÈSE DE DOCTORAT ÈS SCIENCES

présentée et soutenue par

Maxime LEFRANÇOIS

Représentation des connaissances sémantiques lexicales de la Théorie Sens-Texte :

Conceptualisation, représentation, et opérationnalisation des définitions lexicographiques

Thèse dirigée par **Fabien GANDON** et codirigée par **Christian BOITET**
Préparée à Wimmics, Inria Sophia Antipolis - Méditerranée

Soutenue publiquement le 24 juin 2014 devant le jury composé de :

Nathalie AUSSENAC-GILLES (DR. CNRS) – rapporteur
Université de Toulouse, IRIT, CNRS, Toulouse

Igor BOGUSLAVSKY (Pr.) – rapporteur
Universidad Politécnica de Madrid & Russian Academy of Sciences, Moscow

Marie-Laure MUGNIER (Pr.) – rapporteur
Université Montpellier 2, LIRMM/Inria

Andrea TETTAMANZI (Pr.) – examinateur
Université de Nice Sophia Antipolis, I3S/Inria

Fabien GANDON (DR. Inria) – directeur de thèse
Inria Sophia Antipolis-Méditerranée

Christian BOITET (Pr.) – codirecteur de thèse
Université Joseph Fourier, LIG, Grenoble

Résumé étendu

Les linguistes, comme toute communauté d'intérêt, produisent des connaissances. Des besoins récurrents émergent de ces connaissances produites, auxquels le domaine de l'ingénierie des connaissances cherche à répondre. Dans cette thèse, nous proposons une démarche d'ingénierie des connaissances appliquée aux connaissances lexicales sémantiques du Dictionnaire Explicatif et Combinatoire de la Théorie linguistique Sens-Texte (TST). Nous nous intéressons en particulier à trois briques élémentaires de connaissances dans le DEC : les prédicats linguistiques, définis dans la théorie des Actants Sémantiques de Mel'čuk, les représentations linguistiques, et les définitions lexicographiques, symbolisées par une représentation sémantique dans la TST. Nous adoptons une méthodologie en trois étapes : extension de la conceptualisation, élaboration d'un formalisme de représentation des connaissances, et opérationnalisation de ce formalisme.

Nous introduisons la TST, précisons le cadre d'étude de notre travail, et définissons la méthodologie de notre étude d'ingénierie des connaissances dans la partie I de ce mémoire.

La partie II concerne la conceptualisation de la TST. Nous montrons que la hiérarchisation des prédicats sémantiques, avec héritage et spécialisation de leur structure actancielle, ne peut correspondre à une hiérarchisation des sens. Cela justifie alors d'étendre la TST par la définition d'un niveau sémantique profond pour représenter les sens. Nous introduisons donc la notion d'unité sémantique profonde, et de sa structure actancielle : un ensemble de positions actancielles lexicalisées, qui peuvent être obligatoires, optionnelles, ou interdites. Nous montrons que la hiérarchie des unités sémantiques profondes, avec héritage et spécialisation de la structure actancielle, peut correspondre à une hiérarchie des sens. Nous montrons ensuite que, dans la TST, une définition lexicographique est une représentation sémantique de l'instanciation prototypique de la situation linguistique dénotée par une lexie $\ulcorner \mathrm{L} \urcorner$. Nous définissons donc la notion de définition lexicographique formelle de $\ulcorner \mathrm{L} \urcorner$ au niveau sémantique profond, ainsi qu'au niveau du dictionnaire. Un premier prototype d'éditeur de définitions lexicographiques formelles a été développé pour une intégration future de nos travaux dans des éditeurs de lexicographie explicative et combinatoire.

La partie III concerne le choix d'un formalisme de représentation des connaissances pour la TST. Au premier abord, les formalismes du Web Sémantique et le formalisme des Graphes Conceptuels semblent les plus adaptés à la représentation des connaissances linguistiques. Cependant, nous montrons que ces formalismes ne peuvent représenter ni les prédicats linguistiques ni les définitions lexicographiques de manière formelle et naturelle à la fois. Une raison principale est que ces formalismes échouent à représenter les prédicats sémantiques, qui peuvent être considérés à la fois comme des concepts et comme des relations *n*-aires. Nous justifions ainsi la construction d'un nouveau formalisme de représentation des connaissances à base de graphes, que nous nommons formalisme des Graphes d'Unités. Les unités sémantiques profondes sont représentées par des types d'unité dans le dictionnaire, et par des instances de types d'unité dans les représentations linguistiques. Nous définissons formellement la hiérarchie des types d'unité et de leur structure actancielle. Les Graphes d'Unités sont composés d'instances de types d'unité interconnectées par des relations de dépendance actancielles ou circonstancielles. Nous introduisons finalement la notion de règle, puis de définition des types d'unité comme deux règles réciproques. Nous montrons que les définitions de types d'unité permettent de représenter le sous-ensemble des définitions lexicographiques dont toutes les composantes sont obligatoires.

La dernière partie étudie l'opérationnalisation du formalisme des graphes d'unités. Dans un premier temps, nous étudions le raisonnement logique, et les capacités de raisonnement dans le formalisme des Graphes d'Unités. Nous lui associons une sémantique d'interprétation basée sur l'algèbre relationnelle, et introduisons une base de règles axiomatiques d'inférence à partir de laquelle nous définissons la déduction logique. Nous étudions alors les conditions de décidabilité du problème de déduction logique, et montrons que le processus de déduction est correct vis à vis de la sémantique formelle, et complet dans certains cas. Dans un second temps, nous étudions l'opérationnalisation du formalisme des Graphes d'Unités sur le Web des données. Nous choisissons le méta-modèle OWL 2 RL et simulons la sémantique du formalisme par la base de règles OWL 2 RL/RDF, que nous augmentons d'un ensemble ad hoc de règles SPARQL. Nous instancions alors notre modèle pour la Théorie Sens-Texte, et alignons ce dernier modèle avec celui proposé par le groupe communautaire Ontology-Lexica du W3C.

Nos travaux de recherche ouvrent de nombreuses perspectives, tant sur l'enrichissement du formalisme des Graphes d'Unités pour représenter plus de connaissances de la TST, que sur des applications en ingénierie des connaissances, en lexicographie, et en TALN.

Remerciements

Je ne remercierai jamais assez Fabien Gandon pour m'avoir si bien accompagné durant toute la préparation de cette thèse. Du choix du sujet à la préparation de l'après-thèse. Fabien, c'est une chance et un honneur d'avoir été un de tes doctorants.

Je remercie Christian Boitet, pour sa rigueur et pour tous les pointeurs intéressants que je remportais avec moi après chacune de nos entrevues. Ce mémoire a été écrit dans la perspective de sa relecture minutieuse.

Un merci tout particulier à Alain Giboin, Romain Gugert, Olivier Corby, Alain Polguère, Sylvain Kahane. Cette thèse perdrait en cohérence si nous n'avions pas travaillé ensemble.

Merci aux membres passés et présents de l'équipe Wimmics. Tous les permanents, les post-doctorants, ingénieurs, doctorants, et stagiaires que j'ai côtoyé. L'âme de Wimmics évolue aux rythmes de vos départs et de vos arrivées, mais son humanité, sa richesse et sa joyeuseté restent des constantes grâce à chacun d'entre vous. Je tiens à saluer tout particulièrement Christine Foggia pour son aide incroyable pour toutes mes missions tordues.

Merci à mes collègues enseignants-chercheurs de l'UFR sciences, la Miage, et l'IUT Nice Côte d'Azur. J'ai aimé confirmer mon goût pour l'enseignement à vos côtés.

Je remercie ma famille pour leur soutien sans faille malgré la distance qui nous sépare. Joyeux anniversaire de 30 ans Alex !

Un merci tout particulier à Rachel Martin. Tu m'as aidé à garder conscience qu'un doctorant reste avant tout un être humain.

Merci à la famille Martin pour leur soutien et leur amitié.

Merci enfin à mes amis du COV, de CanapéTeam, et du Catrusoutrane. Merci à tous mes amis grimpeurs et non grimpeurs du 06 et d'ailleurs. Nos soirées, virées, l'escalade, la nature, le beau temps, ont contribué au delà de ce que vous pourrez jamais imaginer à mon équilibre, et à la réussite de cette thèse.

Table des matières

Liste des abréviations

DEC Dictionnaire Explicatif et Combinatoire

GC Graphe Conceptuel
GU Graphe d'Unités

IC Ingénierie des Connaissances
IUSemP Instance d'Unité Sémantique Profonde

LD Logiques de Description

modèle ST modèle Sens-Texte

OWL Web Ontology Language

PosA Position Actancielle
PosAInt Position Actancielle Interdite
PosAObl Position Actancielle Obligatoire
PosAOpt Position Actancielle Optionnelle
PosASyn Position Actancielle Syntaxique
PosASém Position Actancielle Sémantique

RC Représentation des Connaissances
RDF Resource Description Framework
RDFS RDF Schema
RMorph Représentation Morphologique
RPhon Représentation Phonologique
RSyn Représentation Syntaxique
RSynP Représentation Syntaxique Profonde
RSynS Représentation Syntaxique de Surface
RSém Représentation Sémantique
RSémP Représentation Sémantique Profonde
RSémS Représentation Sémantique de Surface

SPARQL SPARQL Protocol and RDF Query Language
SRelA Symbole de Relation Actancielle
SRelC Symbole de Relation Circonstancielle

TALN Traitement Automatique des Langues Naturelles
TST théorie linguistique Sens-Texte

TUC Type d'Unité Conjonctif

TUP Type d'Unité Primitif

TUSemP Type d'Unité Sémantique Profonde

TUSemS Type d'Unité Sémantique de Surface

UNL Universal Networking Language

URI Uniform Resource Identifier

USemP Unité Sémantique Profonde

W3C World Wide Web Consortium

Table des figures

Liste des définitions

Introduction générale

*_____ **L**a linguistique est une chose trop sérieuse pour être confiée aux seuls linguistes.*
Marina Yaguello, Alice au pays du langage

Contexte : la représentation des connaissances lexicales

Nous observons aujourd'hui une explosion des ressources numériques textuelles. Bien évidemment, l'Internet et le Web reflètent cette explosion. Des modèles et algorithmes de Traitement Automatique des Langues Naturelles (TALN) sont nécessaires pour que les machines soient capables d'interpréter le texte. Par ailleurs, certains linguistes s'intéressent à l'élaboration de ressources lexicales, mais leur utilisation est restée limitée en TALN à cause de la prédominance des approches basées sur les statistiques et l'apprentissage automatique, et de l'augmentation de la puissance de calcul disponible. Aujourd'hui, la tendance s'inverse et un consensus émerge au sein de la communauté du TALN pour dire que des connaissances linguistiques doivent être injectées dans les modèles et algorithmes afin de continuer à améliorer la qualité des systèmes de TALN. Dans cette thèse, nous nous intéressons aux connaissances linguistiques et aux façons de les représenter.

De multiples formalismes descriptifs ont été développés au sein de la communauté des linguistes. Certains sont plus précis d'un point de vue descriptif, et d'autres sont plus formels. L'un des formalismes les plus précis au niveau descriptif est développé depuis environ 1967 dans le cadre de la Théorie Sens-Texte (Mel'čuk, 1997). Le lexique, nommé Dictionnaire Explicatif et Combinatoire (DEC), y est central (cf. par exemple Mel'čuk *et al.*, 1999; Mel'čuk, 2006). Dans cette thèse, nous nous intéresserons en particulier aux connaissances lexicales du Dictionnaire Explicatif et Combinatoire, dont la forme est conçue par les lexicologues, et dont le contenu est développé par les lexicographes.

En général, la sémantique logique des formalismes descriptifs les plus précis reste majoritairement interne, c'est-à-dire dans la tête de leurs développeurs. C'est notamment le cas pour le DEC. Avec l'informatisation des connaissances lexicales, le besoin de rendre explicite la sémantique logique de ces formalismes se fait sentir, et ce dans plusieurs perspectives applicatives :

- Au niveau du lexique, l'inférence permet d'envisager des scénarios d'aide au lexicographe dans sa tâche d'élaboration d'articles du dictionnaire, en permettant la validation de la cohérence du lexique, ou bien la découverte de nouvelles connaissances.
- Au niveau des représentations d'énoncés, l'inférence permet d'envisager la participation à l'amélioration des applications de TALN.

Dans cette thèse, nous chercherons donc à rendre explicite et implémenter la sémantique logique du DEC. Nous nous concentrerons en particulier sur la sémantique lexicale, et nous étudierons les trois briques élémentaires de connaissances suivantes : les prédicats linguistiques, les représentations linguistiques, et les définitions lexicographiques.

Parallèlement à l'émergence de ces besoins spécifiques au TALN et à la lexicographie, nous assistons aujourd'hui à une explosion de la quantité de connaissances publiées, en particulier sur le Web des données [1]. La production de ces connaissances implique l'émergence de besoins spécifiques, auxquels l'ingénierie des connaissances cherche à répondre. Il s'agit de permettre par exemple de représenter, de manipuler, d'échanger, d'interroger, ou de raisonner avec les connaissances. Différents formalismes de représentation des connaissances ont été développés. Parfois, ils nécessitent d'être adaptés à un domaine spécifique, et parfois un formalisme existant peut être réutilisé pour un nouveau domaine. Comme toute communauté d'intérêt, les linguistes produisent des connaissances. Quelques écoles de linguistique ont ainsi naturellement embrassé le domaine de la représentation des connaissances, et des projets de standardisation des ressources lexicales ont vu le jour. Cependant, la théorie linguistique Sens-Texte (TST), avec sa finesse descriptive (et donc la complexité qui en découle), n'a pas encore franchi le cap. Pourtant, l'ingénierie des connaissances propose des solutions à certains besoins exprimés et latents. En particulier, les connaissances générées par différents groupes de lexicographes pourraient être rendues interopérables, et, du coup, réutilisables.

Objectifs de la thèse

Ainsi, la question de recherche principale à laquelle nous proposons de répondre dans cette thèse est la suivante :

Quel formalisme de représentation des connaissances serait adapté à la représentation des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques du Dictionnaire Explicatif et Combinatoire (DEC) ?

Dans cette thèse, nous mettons donc en œuvre une démarche d'ingénierie des connaissances appliquée aux connaissances linguistiques de la TST. Nous adopterons une méthodologie en trois étapes, qui vise à répondre séquentiellement aux sous-questions de recherche suivantes :

- Comment doit-on étendre la conceptualisation de la TST afin de faciliter la formalisation des connaissances du Dictionnaire Explicatif et Combinatoire ?
- Quel formalisme de représentation de connaissances serait naturellement approprié à la représentation des connaissances du DEC ?
  - Existe-t-il un formalisme standard de représentation des connaissances adéquat ?
  - Si non, comment peut-on construire un nouveau formalisme approprié ?
- Comment peut-on rendre ce formalisme opérationnel ? En particulier :
  - Quel type de raisonnement logique effectuer dans ce formalisme ?
  - Comment partager les connaissances linguistiques de la TST sur le Web des données ?

1. Le Web des données est une initiative du W3C, très active aujourd'hui. http://linkeddata.org

Organisation détaillée du mémoire

Ce mémoire est organisé de la manière suivante.

Partie I : Cadre et positionnement de l'étude : Théorie Sens-Texte et représentation des connaissances linguistiques. Cette partie spécifie le cadre de notre travail de thèse, introduit les notions indispensables à la compréhension de ce mémoire, et positionne notre travail dans le champ de la recherche actuelle. Nous y donnerons une idée des limites des pratiques actuelles, et y présentons les motivations et les principes généraux de notre travail.

Chapitre 1 - Introduction à la théorie linguistique Sens-Texte et définition du cadre d'étude. Ce chapitre introduit la théorie Sens-Texte que nous nous proposons de représenter, les frontières de notre travail, ainsi que les notions importantes du modèle Sens-Texte nécessaires à la compréhension de ce mémoire. Nous attachons une importance particulière à la description des prédicats sémantiques, des représentations linguistiques et des définitions lexicographiques, ainsi qu'au positionnement de nos travaux par rapport aux recherches passées et présentes.

Chapitre 2 - Propositions et standards de l'ingénierie des connaissances. Ce chapitre introduit la représentation des connaissances, et détaille la méthodologie de notre approche. Il justifie ensuite le choix des formalismes qui semblent les plus adaptés à la représentation des connaissances de la Théorie Sens-Texte, et en propose une vue d'ensemble. Enfin, nous présenterons différents travaux concernant la représentation de connaissances linguistiques.

Partie II : Extension de la conceptualisation de la Théorie Sens-Texte. Cette partie étudie dans le détail la conceptualisation actuelle des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques. Nous en précisons les aspects problématiques pour une formalisation ultérieure, et en étendons alors la conceptualisation.

Chapitre 3 - Conceptualisation des prédicats sémantiques : nécessité d'introduire un niveau sémantique profond. Ce chapitre étudie dans quelle mesure la conceptualisation actuelle des prédicats sémantiques doit être étendue pour que sa formalisation ultérieure soit facilitée. Nous justifions notamment l'introduction d'un niveau sémantique profond pour représenter les sens, et y définissons les prédicats linguistiques associés.

Chapitre 4 - Conceptualisation des représentations sémantiques et des définitions lexicographiques. Ce chapitre étudie dans quelle mesure la conceptualisation actuelle des représentations sémantiques et des définitions lexicographiques doit être étendue pour que sa formalisation ultérieure soit facilitée. Nous introduisons notamment les représentations linguistiques pour le niveau sémantique profond, et précisons la notion de définition lexicographique.

Chapitre 5 - Application de la conceptualisation étendue dans un projet de lexicographie explicative et combinatoire. Ce chapitre étudie comment l'extension de la conceptualisation proposée peut être utilisée pour l'édition de la zone sémantique dénotationnelle du DEC, dans le cadre d'un projet de lexicographie explicative et combinatoire. Nous présentons pour cela un prototype d'éditeur de définitions lexicographiques formalisées selon notre conceptualisation.

Partie III : Formalisation des prédicats sémantiques et des définitions lexicographiques. Cette partie concerne la formalisation de la conceptualisation étendue que nous avons proposée. Il s'agit donc de choisir un formalisme de représentation des connaissances adapté.

Chapitre 6 - Inadéquation des logiques de description et des Graphes Conceptuels. Ce chapitre étudie les candidatures des Logiques de Description et des Graphes Conceptuels pour représenter les connaissances de la TST, selon la conceptualisation que nous avons proposée. Nous justifions la construction d'un nouveau formalisme de représentation des connaissances à base de graphes, que nous nommons formalisme des Graphes d'Unités.

Chapitre 7 - Construction du formalisme des Graphes d'Unités pour la représentation des prédicats linguistiques. Ce chapitre initie la construction du formalisme des Graphes d'Unités, par l'introduction et la caractérisation d'une hiérarchie de types d'unité munis d'une structure actancielle, adaptée à la conceptualisation introduite au chapitre 3.

Chapitre 8 - Des graphes d'unités aux définitions de types d'unité pour représenter les définitions lexicographiques. Ce chapitre poursuit la construction du formalisme des Graphes d'Unités par la définition des graphes d'unités eux-mêmes, et leur utilisation pour formaliser les définitions lexicographiques.

Partie IV : Vers une opérationnalisation du formalisme des Graphes d'Unités. Cette partie détaille deux études concernant l'opérationnalisation du formalisme de représentation des connaissances que nous avons construit pour la TST.

Chapitre 9 - Raisonnement dans le formalisme des Graphes d'Unités. Ce chapitre définit une sémantique formelle pour le formalisme des Graphes d'Unités, et caractérise ensuite le raisonnement logique associé.

Chapitre 10 - Opérationnalisation sur le Web des Données. Ce chapitre introduit une modélisation du formalisme des Graphes d'Unités à l'aide des formalismes du Web Sémantique, afin de profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances sur le Web des données. Nous alignons ce modèle avec le modèle Ontolex de représentation des connaissances lexicales.

Chapitre 11 - Perspectives. Ce dernier chapitre ouvre progressivement le cadre d'étude restreint que nous nous sommes fixé, et passe en revue différentes perspectives de recherche qu'offrent nos travaux. Nous nous focalisons sur l'enrichissement du formalisme des Graphes d'Unités pour représenter plus de connaissances de la TST, et sur les applications en ingénierie des connaissances, en lexicographie, et en TALN.

Première partie

Cadre et positionnement de l'étude : Théorie Sens-Texte et définitions lexicographiques

CHAPITRE 1

Définition du cadre d'étude : le sens lexical dans le cadre de la théorie linguistique Sens-Texte

---

L'informatisation du [Dictionnaire Explicatif et Combinatoire (DEC)] complet, si elle pose de réels problèmes, n'est ni une tâche utopique, ni quelque chose de non naturel. Le DEC, DANS SA CONCEPTION, EST FAIT POUR ÊTRE INFORMATISÉ. *En ce sens, le DEC n'est pas différent de toute description relevant de la linguistique formelle. Le seul intérêt véritable de formaliser une description linguistique est de se donner les moyens de "faire tourner" un appareillage logico-déductif sur cette description afin (i) de vérifier sa validité, et (ii) d'en déduire plus de connaissances sur la langue.*
Mel'čuk et al. *(1995, p.208)*

Sommaire

Introduction

Ce chapitre définit le problème auquel cette thèse répond : formaliser la représentation des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques dans le cadre de la TST, de façon, comme d'écrit Mel'čuk, à pouvoir "“*faire tourner” un appareillage logico-déductif sur cette description*".

La section 1.1 adopte tout d'abord un point de vue plus large, afin de bien définir le positionnement de notre travail dans le champ des travaux en sémantique lexicale. Nous y justifions notre choix de travailler dans le cadre de la lexicologie explicative et combinatoire de la TST.

La suite de ce chapitre introduit les notions importantes du modèle Sens-Texte nécessaires à la compréhension de ce mémoire, et précise la portée de notre travail. La section 1.2 introduit la TST, et pose certaines frontières à notre travail. Enfin, nous détaillons dans la section 1.3 les notions de prédicats sémantiques et de définitions lexicographiques auxquelles nous nous intéressons plus particulièrement. Nous en verrons quelques limites et donnerons à cette occasion un aperçu de certains apports de notre travail.

1.1 Positionnement dans le champ de la sémantique lexicale

La sémantique lexicale est l'étude du sens des unités lexicales d'une langue. Nous présentons différentes approches actuelles de la conceptualisation du sens lexical (§1.1.1), puis nous précisons le cadre d'étude de cette thèse : la conceptualisation décompositionnelle du sens lexical dans le cadre de la théorie linguistique Sens-Texte (§1.1.2).

1.1.1 Conceptualisation du sens lexical

Il existe une grande variété de conceptualisations du sens lexical (cf., par ex. Geeraerts, 2010). Cette section présente trois courants toujours actuels de la conceptualisation du sens lexical : les approches décompositionnelles (§1.1.1.1), componentielles (§1.1.1.2), et relationnelles (§1.1.1.3).

1.1.1.1 Sémantique décompositionnelle

Selon l'approche décompositionnelle, le sens lexical doit être décrit par sa décomposition en unités de sens sémantiquement plus simples, afin de lui attribuer une définition explicite.

Dans son expression la plus extrême, l'approche décompositionnelle mène à la recherche de primitives sémantiques ou conceptuelles, à partir desquelles toute unité lexicale peut être décrite. Citons notamment les primitives de Shank (Schank et Abelson, 1977), ou bien la Métalangue Sémantique Naturelle (Wierzbicka, 1996), qui propose une soixantaine de primitives sémantiques pour définir toute unité lexicale.

D'un autre côté, l'approche lexicographique classique décrit le sens des unités lexicales par des définitions lexicographiques dites analytiques, qui représentent le sens des unités lexicales sous la forme aritotélicienne d'un texte paraphrastique avec genre proche et différences spécifiques. Dans ce contexte, l'une des approches les plus précises au niveau descriptif est développée dans le cadre de la TST. Comme nous le verrons, la complexité descriptive des définitions lexicographiques rend leur formalisation difficile. Leur utilisation dans les applications de TALN est donc aujourd'hui limitée. Cependant, il a été noté (Polguère, 1990; Wanner, 2003) que certaines applications bénéficieraient grandement d'une formalisation plus poussée des définitions lexicographiques.

1.1.1.2 Sémantique componentielle

Suivant l'approche componentielle, une définition prend la forme d'une conjonction de caractéristiques discrètes.

Dans le lexique utilisé par les grammaires de Montague (Dowty, 1979; Dowty *et al.*, 1980; Moot et Retoré, 2012), la définition d'une unité lexicale est représentée par deux lambda-expressions logiques : l'une représente sa compositionnalité syntaxique, et l'une autre représente sa compositionnalité sémantique. Les grammaires formelles décrivent alors la manière dont les unités lexicales peuvent se combiner pour former une phrase. Ainsi, chaque grammaire formelle impose une vision plus ou moins complexe du lexique, suivant les opérations de composition syntaxique et sémantique envisagées.

En parallèle au développement de grammaires formelles, différentes représentations du lexique ont donc été proposées. Le lexique de DATR (Evans et Gazdar, 1989, 1996; Keller, 1995), du formalisme LFG (Dalrymple, 1999), le Lexique Génératif de Pustejovsky (1991), et sa reformulation par Vanier *et al.* (2006), représentent ainsi la définition d'une unité lexicale sous la forme de structures de traits sémantiques et syntaxiques, un trait étant un couple (attribut, valeur), dans lequel la valeur est atomique ou complexe. Un des problèmes de l'intrication sémantique-syntaxe dans le lexique des approches componentielles est qu'il rend impossible la monotonie de l'héritage des caractéristiques dans le lexique (Bouma, 1992). Nous verrons qu'un formalisme monotone suffira pour ce qu'on cherche à représenter de la TST.

La Grammaire d'Unification Sens-Texte (Kahane, 2002; Kahane et Lareau, 2005), une instanciation de la Grammaire d'Unification Polarisée (Kahane, 2004), est particulièrement intéressante pour nous. En effet, son niveau sémantique contrôle la bonne combinaison des unités sémantiques pourvues de leurs structures actancielles. Cependant, aucune formalisation des définitions lexicographiques de la TST n'y est pour le moment utilisée.

1.1.1.3 Sémantique relationnelle

La conceptualisation relationnelle des sens lexicaux est à l'origine de la génération *Net* des bases de données lexicales. Le sens d'une unité lexicale y est implicitement défini par différentes relations que l'unité lexicale entretient avec les autres unités lexicales de la langue, dans ce qu'on appelle un *réseau lexical*.

On peut caractériser la richesse d'un réseau lexical par la variété de la nature des relations qui y lient les unités lexicales, et du degré d'axiomatisation de ces relations. Ainsi, le réseau lexical WordNet et ses différents dérivés multilingues EuroWordNet, MultiWordNet, et IndoWordNet (Miller, 1995; Fellbaum, 1998; Vossen, 1998; Pianta *et al.*, 2002; Sinha *et al.*, 2006), bien que très couvrants et probablement les plus utilisés en TALN, n'implémentent que des relations lexicales paradigmatiques simples (synonymie, hyperonymie, méronymie,...). La relation d'hyperonymie, ou relation *is-a*, est très étudiée, car elle permet de concevoir le réseau lexical comme une structure hiérarchisée. Implémenter la transitivité de cette relation revient à axiomatiser le réseau lexical. On parle alors d'*ontologie lexicale*. Ainsi, alors que WordNet possède une grande couverture mais une faible axiomatisation, d'autres ontologies lexicales sont moins couvrantes mais fortement axiomatisées. C'est le cas notamment de Multi-Net (Helbig, 2006) et de HowNet (Dong et Dong, 2006). Ces ontologies lexicales permettent alors un certain raisonnement avec le sens des phrases annotées sémantiquement.

Dans une perspective qui participe plus de la lexicographie que du TALN, les Fonctions Lexicales de la TST (Mel'čuk, 1996) représentent une des conceptualisations les plus complètes des différentes relations lexicales. Le réseau lexical DiCo (Dictionnaire de Cooccurrences) et son successeur le RLF (Réseau Lexical du Français) (Polguère, 2009; Lux-Pogodalla et Polguère, 2011) sont tissés de ces liens de fonctions lexicales. De la même manière, le réseau lexical construit de manière contributive par le biais de *jeux sérieux* dans le cadre des projets Papillon et JeuxDeMots (Mangeot *et al.*, 2003; Lafourcade, 2007) est également tissé de liens inspirés des fonctions lexicales.

Parmi ce qu'on peut nommer réseau lexical, FrameNet (Baker *et al.*, 1998), basée sur les cadres de (Fillmore, 1977; Fillmore *et al.*, 2003), a la particularité de se baser sur une approche cognitiviste de la sémantique. Les unités lexicales sont classées sous des "cadres lexicaux", qui possèdent un certain "cadre de valence" décrivant leur valence sémantique, et qui sont également liées entre elles par des relations. Bien que basée sur une théorie linguistique différente, FrameNet possède certaines similarités avec la TST, et des travaux ont étudié l'apport potentiel de chacune de ces théories à l'autre (Coyne et Rambow, 2009; Bouveret et Fillmore, 2008).

1.1.2 Choix du cadre d'étude : les définitions lexicographiques dans la théorie linguistique Sens-Texte

Ainsi, en tant que théorie linguistique extrêmement précise du point de vue descriptif, la TST tient une place centrale dans le champ de la sémantique lexicale. La richesse de description des fonctions lexicales et des définitions lexicographiques fait sa force, mais aussi sa faiblesse, en limitant : (i) sa représentation formelle et son axiomatisation ; (ii) la couverture de la langue ; et (iii) les applications en TALN.

En particulier, la représentation des définitions lexicographiques a été peu étudiée, et est attendue pour en faire bénéficier les applications en TALN. Nous proposons donc d'étudier ce point précis dans cette thèse, ce qui nous mène à placer en dehors de notre étude :

- l'intégration potentielle avec la Grammaire d'Unification Sens-Texte ;
- la représentation des fonctions lexicales ;
- le lien avec FrameNet.

1.2 Introduction à la Théorie Sens-Texte

La TST est basée sur un modèle fonctionnel particulier de la langue nommé modèle Sens-Texte (modèle ST), où le terme *modèle fonctionnel* est défini de la manière suivante (cf., Mel'čuk, 1997) :

Définition 1 (Modèle fonctionnel de la langue)**.** Un modèle fonctionnel de la langue est un système d'expressions symboliques créé par le chercheur dans le but de représenter le fonctionnement de la langue qu'il étudie.

La TST est inspirée des travaux de Lucien Tesnière (Tesnière, 1959) et a été initiée principalement par Igor Mel'čuk, Alexandre Žolkovskij et Jurij Apresjan, vers la fin des années 1960, à Moscou. Le travail précurseur habituellement cité dans la littérature est celui de Žolkovskij et Mel'čuk (1967).

Cette section introduit les principes fondateurs de la TST (§1.2.1), et les différents niveaux de représentation linguistiques auxquels nous nous intéressons (§1.2.2). Nous présentons enfin le DEC, qui est le lexique au cœur de la TST. Nous introduirons les différents projets d'informatisation du DEC, et plusieurs applications de la TST (§1.2.3).

1.2.1 Principes fondateurs de la TST

Cette section précise dans un premier temps le niveau de représentation linguistique que l'on se propose d'étudier dans le modèle de la langue sur lequel est basée la TST (§1.2.1.1). Les différents types d'unité linguistique que l'on y rencontre sont ensuite détaillés (§1.2.1.2), et nous présentons une première limite de la conceptualisation de la TST que nous nous proposons de lever dans cette thèse (§1.2.1.3).

1.2.1.1 Postulats de base de la TST

La TST est basée sur les trois postulats suivants (nous citons Mel'čuk, 1997) :

> **Postulat 1 : La langue comme correspondance "Sens-Texte"**
> La langue est un système fini de règles qui spécifie une correspondance multi-multivoque entre l'ensemble infini dénombrable des sens et l'ensemble infini dénombrable des textes.

Ainsi, en général, un sens peut correspondre à un très grand nombre d'énoncés synonymes, et un énoncé peut être très ambigu, c'est-à-dire correspondre à un très grand nombre de sens.

> **Postulat 2 : Les modèles Sens-Texte comme outil de description des langues**
> La correspondance doit être décrite par un dispositif logique, qui constitue un modèle fonctionnel de la langue de type Sens-Texte ; il doit être élaboré et présenté dans la direction Sens⇒Texte.

Le modèle ST privilégie donc le sens de la synthèse [1] de textes, plutôt que celui de l'analyse.

> **Postulat 3 : La phrase et le mot comme unités de base de la description linguistique**
> Dans la description de la correspondance, deux niveaux intermédiaires de représentation des énoncés sont nécessaires pour mettre en lumière les faits linguistiques pertinents :

1. La synthèse de texte consiste à générer du texte à partir d'une représentation sémantique. La génération de texte consiste à choisir une représentation sémantique, puis à la synthétiser.

la Représentation Syntaxique (RSyn), qui correspond aux régularités spécifiques à la phrase, et la Représentation Morphologique (RMorph), qui correspond aux régularités spécifiques au mot.

Un sens et un énoncé sont considérés comme des objets symboliques formels appelés respectivement Représentation Sémantique (RSém), et Représentation Phonologique (RPhon). Les RSém sont pensées spécifiques à chaque langue $\mathscr{L}$, et de par leur caractère profond, elles sont encore sujettes à discussion parmi les chercheurs de la théorie Sens-Texte.

Finalement, le modèle ST scinde certains niveaux de représentation en deux : un niveau profond, plus proche des sens, et un niveau de surface, plus proche des textes. Ainsi, sept niveaux de représentation linguistique sont postulés, et douze modules de correspondance sont envisagés, comme illustré par la figure 1.1.

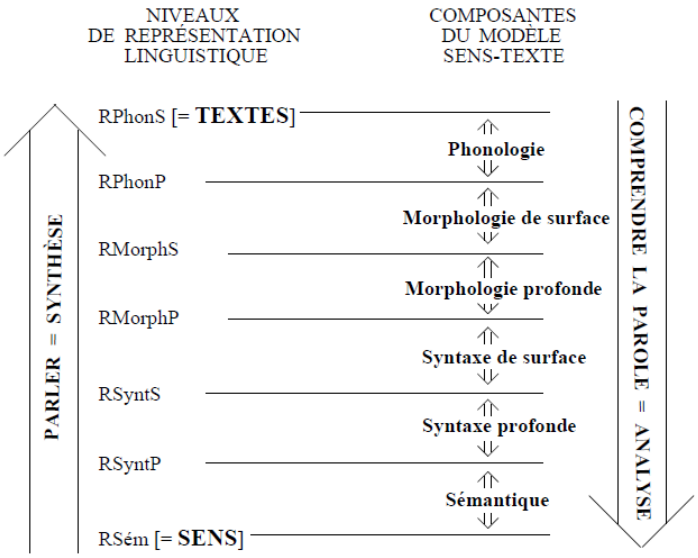


FIGURE 1.1 – Aperçu du modèle Sens-Texte. Les différents niveaux de représentation linguistiques postulés sont à gauche, et les différents modules de correspondance du modèle sont à droite. Illustration par Mel'čuk (1997).

Dans cette thèse, nous nous focaliserons sur l'étude et la représentation du niveau de représentation sémantique, nous aborderons un peu les niveaux de représentation syntaxique, et nous n'étudierons pas les niveaux de représentation les plus surfaciques. Par ailleurs, nous ne nous attarderons ni sur la représentation, ni sur le fonctionnement des modules de correspondance du modèle ST.

1.2.1.2 Unités lexicales, unités sémantiques, unités grammaticales

Nous utilisons les notations d'I. Mel'čuk. Une lexie, notée ⌈L⌉, est une unité lexicale de la langue, c'est-à-dire un mot ou groupe de mots pris dans un sens particulier.

Les lexies sont numérotées pour différencier l'acception choisie. Dans ce mémoire, nous utiliserons la numérotation du CNRTL [2] pour les lexies françaises, et la numérotation du Longman [3] pour les lexies anglaises. Par exemple, ⌈PEIGNE$_{\mathbf{A.1}}$⌉ désigne l'objet de toilette, et ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ désigne un outil du cardeur.

Une lexie peut être un lexème (un mot seulement), ou un phrasème (plusieurs mots, ex : ⌈DONNER LE SEIN⌉, ⌈CORDON BLEU⌉). La TST propose une classification précise des phrasèmes (Mel'čuk, 2003), que nous ne détaillerons pas ici. On y trouve différentes catégories de lexie, et la catégorie des semi-phrasèmes (=locutions semi-figées, =collocations). La TST introduit la notion importante de *fonction lexicale* pour permettre d'étudier certains semi-phrasèmes, mais nous ne l'aborderons pas dans ce mémoire. Nous verrons que la représentation des fonctions lexicales fait justement partie des travaux futurs de ce travail.

Les unités lexicales peuvent être sémantiquement pleines (i.e., chargées de sens), ou sémantiquement vides (par exemple, le verbe support ⌈FAIRE⌉ dans l'expression *faire attention*). Une unité lexicale sémantiquement pleine ⌈L⌉ est associée à une unité sémantique qui en représente le sens. Nous utilisons la notation (L) pour représenter l'unité sémantique associée à la lexie ⌈L⌉.

Une unité lexicale dans un texte peut être augmentée d'unités grammaticales (ou grammatèmes). Les grammatèmes peuvent être de deux types :

Un grammatème à valeur flexionnelle précise le sens de la lexie sans le modifier. Considérons par exemple la phrase *Allons admirer cette œuvre !*. La lexie ⌈ALLONS⌉ y est augmentée des grammatèmes *impératif présent, 1re, personne, pluriel*. La lexie ⌈ŒUVRE⌉ y est augmentée des grammatèmes *singulier* et *défini*. Un grammatème à valeur flexionnelle peut être sémantiquement plein, ou sémantiquement vide (s'il est imposé par une règle d'accord par exemple).

Un grammatème à valeur dérivationnelle change le sens de la lexie (ex : *dés*-œuvr-*é*). Nous n'étudierons pas les unités grammaticales à valeur dérivationnelle.

1.2.1.3 Distinction Unité linguistique - Type d'unité linguistique

Pour une lexie ⌈L⌉, Mel'čuk (2004a) distingue ⌈L⌉ dans la langue $\mathscr{L}$ (i.e., dans le dictionnaire), ou dans l'usage (i.e., dans les énoncés). Il s'agit d'une limite dans la conceptualisation de la TST et nous introduirons un nouveau terme pour lever cette ambiguïté dans la partie II. Nous nommerons alors *type de lexie* la lexie décrite dans le dictionnaire. Nous généraliserons cette distinction et la levée de l'ambiguïté pour les différentes autres unités linguistiques que nous étudions.

2. CNRTL - Centre National de Ressources Textuelles et Lexicales - http://cnrtl.fr/definition/
3. Longman English Dictionary Online - http://www.ldoceonline.com/

1.2.2 Les niveaux de représentation linguistique

Détaillons un peu plus les différentes représentations linguistiques manipulées, ainsi que la nature des objets qui s'y trouvent. A chaque niveau, précisons le cadre de cette thèse. Les illustrations sont reprises de Mel'čuk (1997).

1.2.2.1 Les représentations sémantiques

Une Représentation Sémantique (RSém) est constituée de plusieurs structures superposées. La principale est nommée structure sémantique, et représente le sens propositionnel de l'énoncé. Il s'agit d'un multigraphe étiqueté, orienté, et faiblement connexe, comme représenté sur la figure 1.2.

Dans les structures sémantiques, un nœud est étiqueté par l'unité sémantique qu'il représente (le sens associé à une lexie sémantiquement pleine). Les arcs représentent des relations de dépendance sémantique entre une unité sémantique et une autre, et sont étiquetés par des chiffres. Nous détaillerons leur signification dans la section 1.3. Par exemple, sur la figure 1.2, nous voyons que ⟨Orwell⟩ et ⟨causer$_{\mathbf{1}}$⟩ dépendent de ⟨certain$_{\mathbf{1.A.3}}$⟩.

Les RSém comportent d'autres structures. Par exemple, la structure communicative représente le sens communicatif/subjectif de l'énoncé, et la structure rhétorique représente l'intention pragmatique du locuteur. Dans cette thèse, nous nous concentrerons sur la représentation de la structure sémantique des RSém. Précisons simplement que le nœud communicativement dominant de la représentation, c'est-à-dire celui dont le sens résume le sens de l'énoncé, a son étiquette soulignée dans la représentation de la structure communicative. Sur la RSém de la figure 1.2, il s'agit du nœud étiqueté ⟨certain$_{\mathbf{1.A.3}}$⟩.

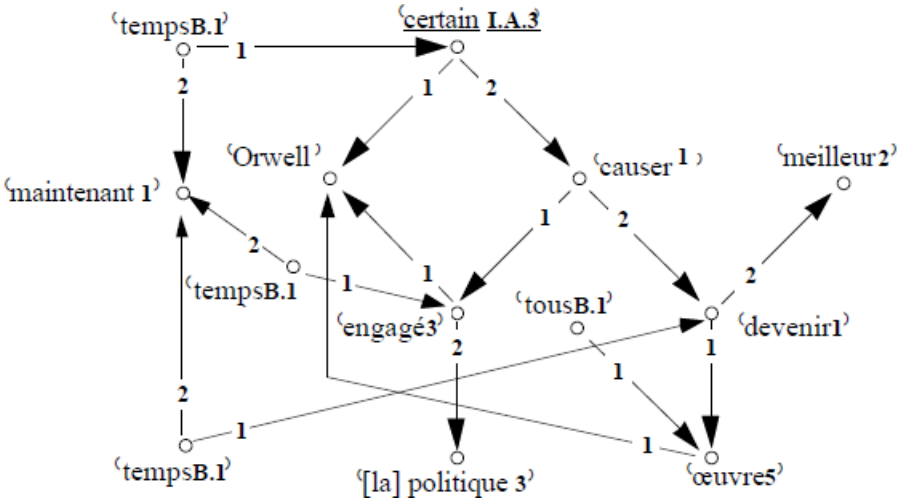


FIGURE 1.2 – Exemple d'une représentation sémantique (RSém). Cette RSém peut être exprimée par un grand nombre de phrases en français, comme par exemple : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997).

1.2.2.2 Les représentations syntaxiques profondes

De manière similaire, une Représentation Syntaxique Profonde (RSynP) est constituée de plusieurs structures superposées. La principale est nommée structure syntaxique profonde. Il s'agit d'un arbre de dépendance, comme illustré par la figure 1.3.

Un nœud est étiqueté soit par la lexie pleine qu'il représente, soit par un symbole de fonction lexicale (ici : **Oper$_1$**). Cette étiquette peut être augmentée de grammatèmes à valeurs flexionnelles pleines. Par exemple, sur la figure 1.3, le nœud étiqueté $\ulcorner$ŒUVRE$\urcorner_{déf,\ pl}$ représente *les œuvres*. Dans cette thèse, nous ne considérerons pas les étiquettes de type fonction lexicale.

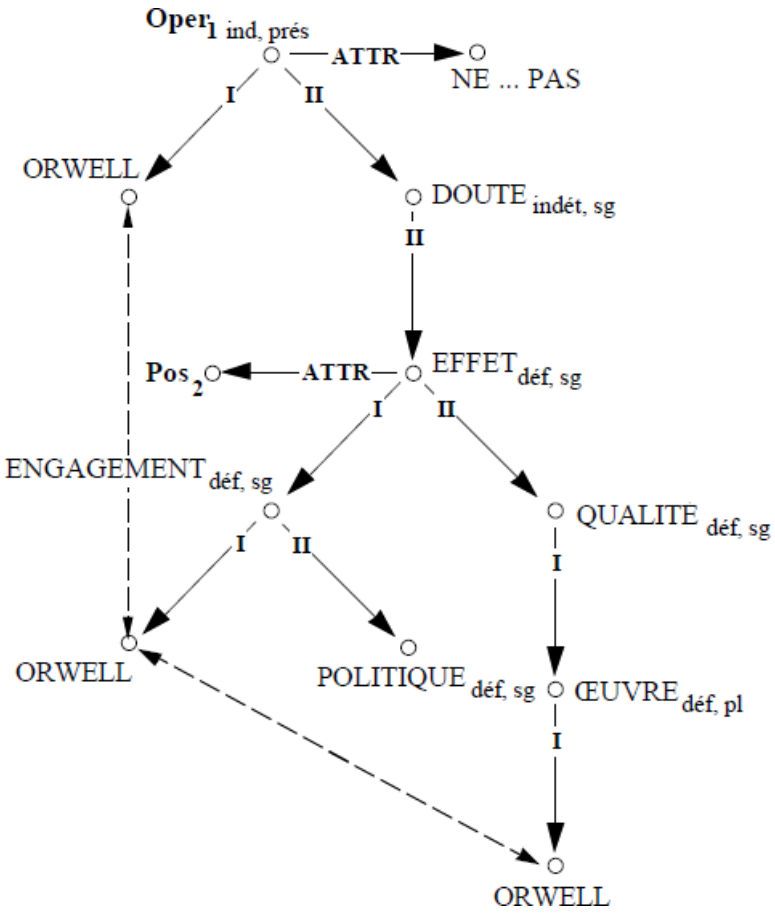


FIGURE 1.3 – Exemple d'une représentation syntaxique profonde (RSynP) de la phrase : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997).

Les arcs représentent des relations de dépendance syntaxique profonde. Ces dépendances peuvent être actancielles, ou circonstancielles. L'étiquette d'un arc est l'une des relations syntaxiques universelles parmi :

- six relations actancielles de **I** à **VI** ;
- la relation circonstancielle attributive **ATTR** pour les modifications ;
- la relation circonstancielle coordinative **COORD** pour les coordinations ;
- la relation circonstancielle appenditive **APPEND** pour les dépendances vagues comme celles qu'entretiennent les interjections ou les adverbes de phrase avec toute la phrase ou avec certains groupes de mots.

Parmi les structures superposées, nous ne considérons que la structure anaphorique, qui représente le lien entre les noms qui ont la même unité sémantique associée à l'aide des coréférences anaphoriques symétriques. Ces liens de coréférence sont représentés par une flèche en pointillé sur la figure 1.3 : les trois nœuds étiquetés ORWELL représentent les trois références à ORWELL dans la phrase : *Orwell*, *son [engagement]*, et *ses [œuvres]*.

1.2.2.3 Le niveau syntaxique de surface

La structure syntaxique de surface, composante principale d'une Représentation Syntaxique de Surface (RSynS), est également un arbre. Mentionnons simplement que ses nœuds sont étiquetés par toutes les lexies présentes dans la phrase, même les lexies vides, ainsi que par des marqueurs grammaticaux. De plus, ses arcs sont étiquetés par un jeu de relations de dépendance syntaxique propre à chaque langue. La figure 1.4 illustre une RSynS.

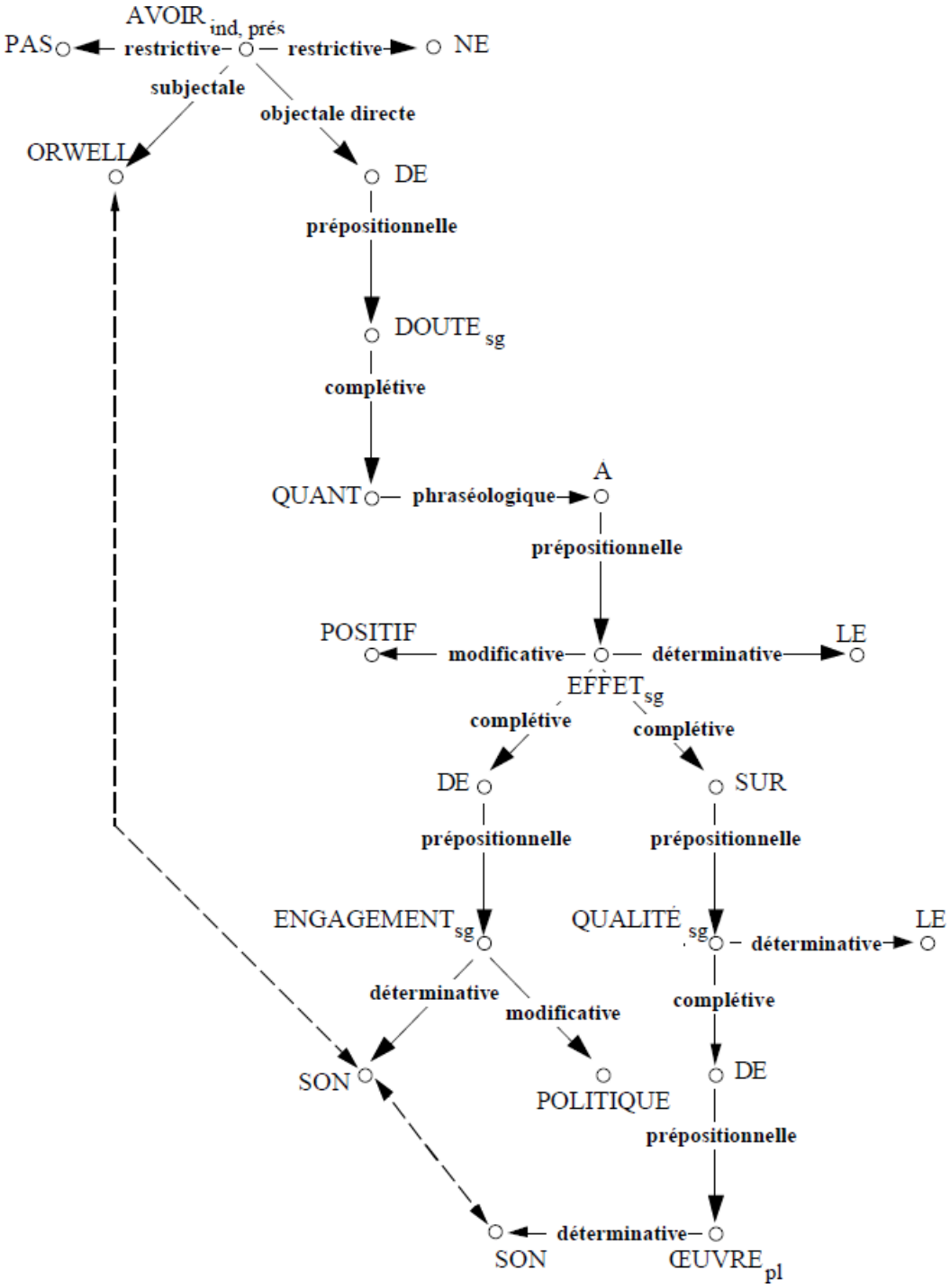


FIGURE 1.4 – Exemple d'une représentation syntaxique de surface (RSynS) de la phrase : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997).

1.2.3 Le Dictionnaire Explicatif et Combinatoire (DEC), informatisation et applications

Chaque module de correspondance peut contenir des règles grammaticales, qui décrivent le cas général, et des règles lexicales, qui décrivent le cas particulier. Une caractéristique importante de la TST est la place importante qu'elle accorde au lexique, nommé DEC (cf. par exemple Mel'čuk *et al.*, 1999; Mel'čuk, 2006).

1.2.3.1 Aperçu des zones du DEC

Intuitivement, nous pouvons imaginer que, plus la description est riche dans le lexique, plus nous pourrons générer de règles lexicales précises, et plus les modules de correspondance seront efficaces. En réalité, la précision descriptive exigée dans le DEC est tellement grande que la plupart des travaux scientifiques de la communauté Sens-Texte portent justement sur le DEC et ses applications directes, plutôt que sur des applications en TALN. Le domaine de recherche privilégié des chercheurs de la TST est donc la lexicologie (l'étude du dictionnaire) et son pendant applicatif, la lexicographie.

Le DEC comporte un article pour chaque lexie de la langue. Précisons qu'un article est organisé en différentes zones, dont les principales sont les suivantes.

- La zone phonologique (ex : prononciation) ;
- La zone morphologique (ex : partie du discours, formes réalisables ou irrégulières) ;
- La zone sémantique (définition dénotationnelle et connotationnelle) ;
- Les zones de combinatoire (comment la lexie peut ou ne peut pas être employée en combinaison avec d'autres dans les phrases) ;
- Les zones de stylistique, d'exemples, phraséologiques, et de nota bene.

Une série de quatre volumes papier du DEC (Mel'čuk *et al.*, 1984, 1988, 1992, 1999) a été publiée pour illustrer la lexicologie Sens-Texte. Il est important de noter que la précision descriptive exigée dans le DEC est telle que ces quatre volumes totalisent environ 500 entrées seulement. Par exemple, l'entrée $\lceil \text{CHEVEU}_{\textbf{I.B.}} \rceil$ est décrite sur 4 pages (cf., Mel'čuk *et al.*, 1999, p.192-196).

Nous concentrons notre travail sur la zone sémantique dénotationnelle du DEC, c'est-à-dire là où le sens linguistique de l'unité lexicale est décrit.

1.2.3.2 Projets d'informatisation du DEC, et applications

Cette section liste un ensemble de projets passés ou actuels qui ont visé à développer ou à utiliser une version informatisée du DEC. La plupart de ces travaux ont une visée de lexicologie ou de lexicographie ; nous recensons néanmoins deux travaux qui utilisent une variété du DEC pour des applications de TALN.

Le projet NADIA-DEC (Sérasset, 1997a,b) s'est focalisé sur l'informatisation du processus de production du DEC. L'éditeur DECID d'articles du DEC a été développé pour faciliter l'édition de chacune des zones du DEC, en particulier pour automatiser la mise en forme des fonctions lexicales. Cependant, les définitions lexicographiques sont représentées seulement sous forme de textes, cf. figure 1.5.

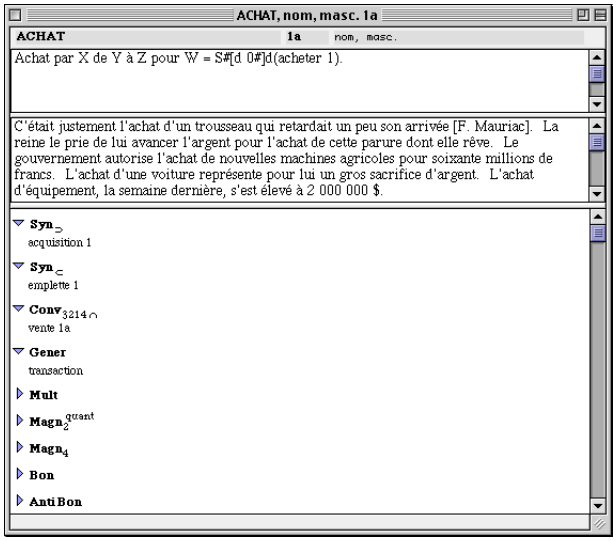


FIGURE 1.5 – Fenêtre de saisie pour la lexie $\lceil \text{ACHAT}_{\mathbf{1.A}} \rceil$ dans l'éditeur DECID. De haut en bas : la zone de définition, la zone d'exemple, et la zone de combinatoire. Copie écran reprise de Sérasset (1997b).

Le projet *Dictionnaire de cooccurrences* (DiCo) fut le premier projet d'informatisation du DEC (Mel'čuk *et al.*, 1995; Polguère, 2000b) à envisager des applications en TALN, et un creuset important pour les travaux de Lexicologie Explicative et Combinatoire. Le choix a été fait de représenter les données facilement formalisables du DEC. Comme nous le verrons, les définitions lexicographiques n'y ont donc été que partiellement représentées. Le DiCo est d'abord encodé dans un modèle objet UML (Steinlin *et al.*, 2004) représenté sur la figure 1.6, puis stocké dans une base de données SQL (Steinlin *et al.*, 2005). Un outil de navigation en ligne et d'accès à la base de données a été développé : le DiCouèbe [4]. Finalement, Lareau (2003) a développé un outil de synthèse de paraphrases à partir d'une RSém, en utilisant un encodage en Prolog de la base du DiCo : *Sentence Garden*. Cet outil a été utilisé par les linguistes pour vérifier la cohérence de la base DiCo.

Le Lexique Actif du Français (ou LAF, cf. Polguère, 2000a, 2007) est une application directe importante du DiCo. Il s'agissait de proposer une interface de consultation adaptée à un public d'enseignants et d'étudiants en linguistique. L'outil de navigation en ligne développé pour le LAF,

4. DiCouèbe - http://olst.ling.umontreal.ca/dicouebe/

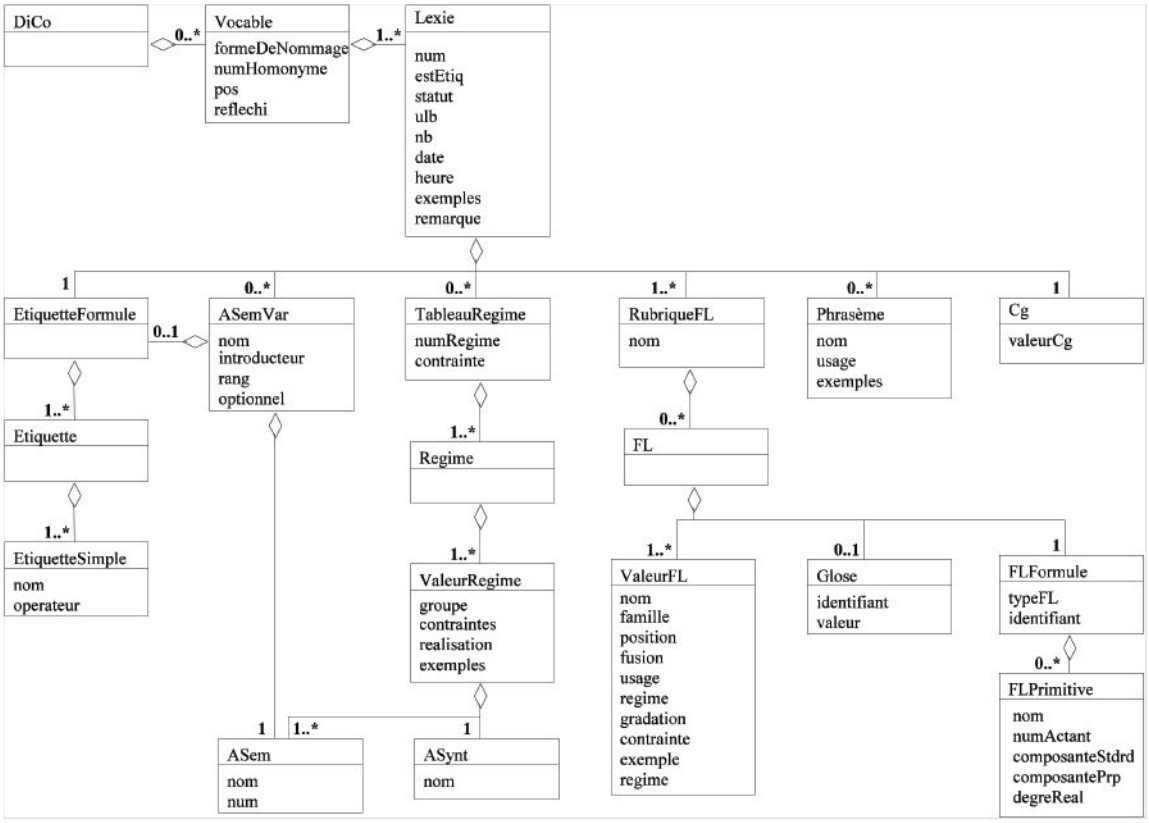


FIGURE 1.6 – Le modèle objet du DiCo. Extrait de Steinlin *et al.* (2004).

DiCoPop [5] (DiCo Populaire), permet ainsi une navigation dans le LAF plus intuitive que l'outil DiCouèbe ne le permet pour le DiCo.

Le projet DiCe est l'équivalent du DiCo pour l'espagnol (Alonso Ramos, 2003, 2004), l'accent est ici encore mis sur les collocations et non pas sur la zone sémantique dénotationnelle.

On peut également citer les dictionnaires terminologiques DicoInfo, DicoEnviro, et Juridico [6] (L'Homme, 2008). Les entrées y sont encodées en XML, et accessibles via l'interface de consultation commune Olster [7], comme le montre la figure 1.7.

Le système de TALN ETAP-3 (Apresian *et al.*, 2003; Boguslavsky *et al.*, 2004) utilise une implémentation d'une variété multilingue du DEC. Les connaissances linguistiques sont représentées dans un formalisme basé sur un calcul des prédicats avec trois valeurs de vérité (cf., Boguslavsky *et al.*, 2004, §1). Bien que certaines informations sémantiques soient représentées, les dictionnaires ne possèdent pas de zone de définition lexicographique.

5. DiCoPop - http://olst.ling.umontreal.ca/laf/
6. Projets de l'OLST - http://olst.ling.umontreal.ca
7. Olster - http://olst.ling.umontreal.ca/olster/olster.php

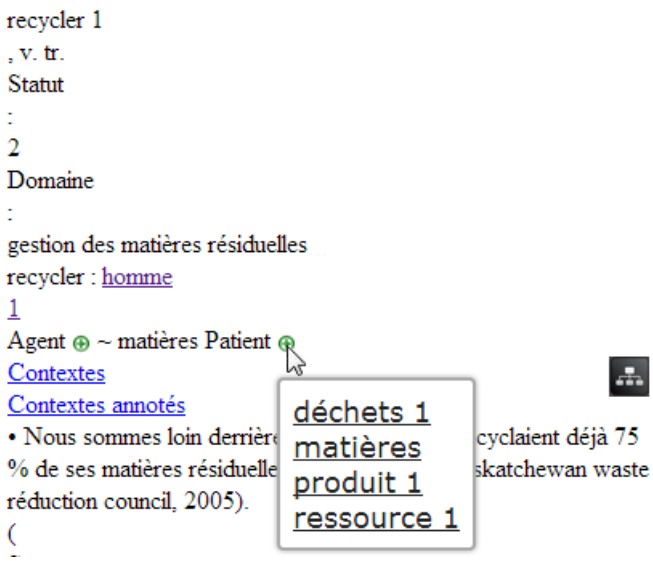


FIGURE 1.7 – Extrait de l'entrée ⌈RECYCLER⌉ du DicoEnviro.

Le système MATE (Bohnet et Wanner, 2010) est un environnement de développement pour la théorie Sens-Texte, utilisé pour des applications de génération automatique de textes de spécialité. L'utilisateur peut y construire des graphes, des règles, et une variété de DEC. Là non plus, les dictionnaires ne possèdent pas de zone de définition lexicographique.

Enfin, le projet RELIEF est un projet de lexicographie d'envergure du français (Lux-Pogodalla et Polguère, 2011), qui tire les enseignements du projet DiCo, et a pour but de construire un réseau lexical du français, RLF (Polguère, 2009) : un DEC aussi formalisé que possible. Nous reviendrons sur la zone sémantique dénotationnelle, qui est toujours en cours d'élaboration. La structure des données dans le projet RELIEF s'inspire de celle du projet DiCo et l'étend. Dans la suite de cette thèse, nous nous intéressons principalement au projet RELIEF, qui présente la formalisation de la zone sémantique dénotationnelle du DEC la plus aboutie.

1.3 Prédicats linguistiques et définitions lexicographiques

Nous nous intéressons maintenant plus particulièrement à la zone sémantique dénotationnelle du DEC. Cette section introduit donc les notions de prédicat sémantique (§1.3.1) et de définition lexicographique (§1.3.2) auxquelles nous nous intéressons plus particulièrement dans cette thèse. Nous en verrons quelques limites et justifierons à cette occasion certains apports de notre travail.

1.3.1 Les prédicats linguistiques dans la théorie Sens-Texte

Selon Mel'čuk (1997, p.9), les unités sémantiques sont divisées en prédicats et noms. Certaines unités sémantiques appellent à gouverner d'autres unités sémantiques, comme ⟨manger⟩ appelle à gouverner d'autres unités sémantiques, qui représentent celui qui mange et ce qui est mangé. D'autres unités sémantiques ne gouvernent pas, comme ⟨vache⟩, ou ⟨herbe⟩.

1.3.1.1 Structure actancielle des lexies

La théorie des actants (Tesnière, 1959; Mel'čuk, 2004a,b) précise ces concepts, et associe à chaque unité lexicale une structure actancielle, formée de trois types de positions actancielles (PosA) : sémantiques, syntaxiques profondes, et syntaxiques de surface. Puisque nous nous concentrons principalement sur la sémantique, nous nous intéressons exclusivement aux *positions actancielles sémantiques* (PosASém).

Situation linguistique dénotée par une lexie. La définition des PosASém implique la notion floue de situation linguistique dénotée par une lexie ⌈L⌉, et de ses participants.

La situation linguistique dénotée par une lexie ⌈L⌉ de la langue $\mathscr{L}$, notée SIT(L), n'a pas de définition précise. Disons qu'elle correspond grossièrement à la représentation mentale provoquée par l'évocation de ⌈L⌉ sous la seule influence de la connaissance de la langue $\mathscr{L}$, et des utilisations possibles de ⌈L⌉ dans la langue $\mathscr{L}$. En ce sens, SIT(L) est à opposer à la notion de situation extra-linguistique [8]. Les composants de SIT(L) sont nommés participants, peuvent être des faits ou des entités, et dépendent les uns des autres de manière à former un tout cohérent. Mel'čuk (2004a) admet que les participants de SIT(L) peuvent être obligatoires, ou optionnels.

Dans la partie II, nous étendrons la notion de SIT(L), sans chercher à en donner une définition plus précise. Nous nous baserons alors sur le principe de l'héritage des participants obligatoires, formulé de la manière suivante par Mel'čuk (2004a) :

Définition 2 (Principe de l'héritage des participants obligatoires)**.** Soit la définition lexicographique de ⌈L⌉ : $\langle L \rangle \equiv \langle L_1 \rangle \oplus \langle L_2 \rangle \oplus \ldots \oplus \langle L_n \rangle$
SIT(L) hérite de chacun des participants obligatoires de chacune des SIT(L_i) qui correspondent aux sens prédicatifs $\langle L_i \rangle$ qui composent $\langle L \rangle$.

Dans cette définition, $\oplus$ représente l'opérateur de combinaison sémantique. Nous pouvons d'ores et déjà entrevoir la question suivante : qu'advient-il des participants optionnels ? Pourquoi ne seraient-ils pas également hérités ? Nous justifierons qu'un participant optionnel peut effectivement être hérité, et nous montrerons l'importance d'introduire un nouveau type de participant : les participants interdits.

8. La situation extra-linguistique est la situation du monde, réel ou virtuel, dont le texte rend compte.

Positions actancielles. Sur la base de ces notions qui sollicitent l'intuition du lexicographe, la théorie des actants définit la notion de structure actancielle d'une lexie ⌈L⌉, qui est composée de PosASém obligatoires et optionnelles. Un ensemble de critères linguistiques est proposé pour discuter des PosASém d'une lexie ⌈L⌉. Ainsi, en substance, ⌈L⌉ possède une PosASém pour chaque participant qui est exprimé de manière privilégiée dans les phrases de la langue $\mathscr{L}$ où ⌈L⌉ apparaît. La PosASém sera obligatoire (resp. optionnelle), si le participant correspondant est obligatoire (resp. optionnel).

Une limite de cette définition est qu'elle attribue les PosASém à la lexie, plutôt qu'à son unité sémantique associée. Nous proposerons dans la partie II d'étendre simplement la conceptualisation de la TST en attribuant les PosASém d'une lexie ⌈L⌉ à son unité sémantique associée. Nous parlerons alors de PosA de ⟨L⟩.

Étiquettes sémantiques. Une étiquette sémantique est un nom de classe sémantique lexicale (Polguère, 2003, 2011). L'ensemble des étiquettes sémantiques est organisé en une hiérarchie de sens : si une étiquette `E1` est sous une étiquette de `E2`, alors toute lexie classée dans `E1` est également classée dans `E2`.

La structure actancielle d'une lexie ⌈L⌉ est complétée par des contraintes sémantiques sur chacune de ses PosASém, sous la forme d'étiquettes sémantiques. Par exemple, l'étiquette sémantique de la PosASém 1 de ⌈MANGER⌉ sera `animal`. Toute lexie ayant l'étiquette sémantique `animal` peut prendre cette PosASém dans les représentations.

Finalement, pour que le système de contrainte soit cohérent, chaque lexie doit elle-même être classée sous une étiquette sémantique directement. Comme nous le verrons, cette classification sémantique des lexies représente une première étape vers la formalisation des définitions lexicographiques. Il s'agit du degré de formalisation atteint par la plupart des projets d'informatisation du DEC (cf., par exemple Steinlin *et al.*, 2004).

1.3.1.2 Positionnement de notre travail

Applications possibles en lexicographie Dans ce travail, nous nous plaçons d'abord comme observateur des pratiques des lexicographes de la TST. Nous considérerons donc le processus du choix des PosASém comme une *boîte noire*. L'apport d'un formalisme de représentation des connaissances adapté à la théorie des actants sémantiques est transversal. Il s'agit par exemple de permettre aux lexicographes de formuler des contraintes sur la structure actancielle des lexies du DEC. Par exemple, nous savons que la plupart des lexies qui possèdent l'étiquette sémantique `action` possèdent une PosASém qui représente l'agent de l'action. Nous pourrions donc développer des outils de vérification de la cohérence du DEC, ou bien proposer au lexicographe une ébauche d'article qui prenne en compte cette contrainte lorsqu'il souhaite écrire l'article de ⌈MANGER⌉ par exemple.

Prédicats linguistiques et leurs structures actancielles Les lexies ont également un ensemble de Position Actancielle Syntaxique (PosASyn) profondes et de surface. À l'instar des prédicats sémantiques, les lexies pourraient donc bien être qualifiées de prédicats linguistiques. Par exemple, toute unité lexicale qui est un verbe transitif devrait avoir au moins deux positions actancielles profondes (PosASynP) **I** et **II**, qui correspondent grosso modo (pour le français) au sujet et au complément d'objet direct du verbe en question. Nous proposerons dans la partie II d'étendre la conceptualisation de la TST de sorte que toute unité linguistique ait sa propre structure actancielle, potentiellement vide.

Plus précisément, nous faisons le choix (arbitraire et non restrictif) d'étendre la conceptualisation de la notion de prédicat sémantique aux autres unités linguistiques. Nous postulons donc que chaque catégorie d'unité linguistique possède sa propre structure actancielle, qui peut être vide.

Une unité sémantique n'est pas un prédicat logique. Si nous considérons qu'une unité sémantique possède une structure actancielle qui correspond aux PosASém de sa lexie associée, alors nous comprenons que les théoriciens sens-texte ont eu la tentation de parler de *prédicat*, par analogie avec les prédicats du calcul des prédicats. Puisque notre travail se situe à la frontière de la représentation des connaissances et de la linguistique, nous désactivons l'ambiguïté terminologique, et utilisons deux termes :

Prédicat linguistique : une unité linguistique, associée à une structure actancielle.

Prédicat logique : un prédicat en calcul des prédicats.

Il y a plusieurs différences fondamentales entre les deux termes.

Tout d'abord, une PosA peut être optionnelle. Ce n'est pas le cas d'un argument pour un prédicat logique. Faudrait-il définir un prédicat logique différent pour chaque cas d'utilisation possible du prédicat linguistique ?

De plus, une PosA, même obligatoire, peut ne pas être exprimée dans le texte. Il existe une variété de cas d'utilisation possibles du prédicat sémantique dans les textes, normalement définis dans les zones de combinatoire du DEC. Faudrait-il définir autant de prédicats logiques que de cas d'utilisation possible du prédicat linguistique ?

Enfin, une unité sémantique peut remplir la PosA d'une autre unité sémantique. Il s'agit là d'une différence fondamentale. En logique du premier ordre, un prédicat logique prend des termes comme argument, et a une valeur de vérité. Un terme qui dépend d'autres termes est une fonction logique, et ne peut pas être un prédicat. Quel serait donc le statut d'une unité sémantique : prédicat logique, fonction logique, ou autre chose ?

Rapport entre les étiquettes sémantiques et les unités sémantiques. Une unité sémantique et une étiquette sémantique représentent *a priori* toutes les deux des sens. La question se pose donc du lien entre ces deux concepts. Peut-on hiérarchiser les unités sémantiques de la même manière que l'on hiérarchise les étiquettes sémantiques ? Peut-on fusionner les deux concepts ? Nous étudierons ce lien dans la section 3.2.

1.3.2 Les définitions lexicographiques

La définition lexicographique d'une lexie ⌜L⌝ présente le sens dénotationnel de ⌜L⌝. Cette section détaille dans un premier temps les travaux qui concernent la structure des définitions lexicographiques (§1.3.2.1). Nous en présentons ensuite une conceptualisation particulière sous la forme d'une RSém (§1.3.2.2), et présentons quelques limites de cette conceptualisation que nous nous proposons de lever (§1.3.2.3).

1.3.2.1 Structure des définitions lexicographiques

Dans le DEC, les définitions lexicographiques sont analytiques, c'est-à-dire de type genre proche et différences spécifiques. Par exemple, on trouve dans (Mel'čuk *et al.*, 1999) dans l'entrée ⌜PEIGNE$_{\mathbf{B.2.D}}$⌝ la définition lexicographique suivante :

peigne$_{\mathbf{B.2.d}}$ de personne X pour objet Y =
(Outil de tissage qu'une personne X utilise pour démêler$_{\mathbf{2}}$ les fibres d'un objet Y)

Le membre gauche de l'égalité représente le défini, ⌜PEIGNE$_{\mathbf{B.2.D}}$⌝, sous sa forme propositionnelle : une expression élémentaire dans laquelle apparaissent ⌜PEIGNE$_{\mathbf{B.2.D}}$⌝ et sa structure actancielle.

Le membre droit de l'égalité représente le définissant. Le genre proche de (peigne$_{\mathbf{B.2.d}}$) est (outil), et le reste de la définition précise les différences spécifiques de (peigne$_{\mathbf{B.2.d}}$) par rapport à (outil).

Les définitions lexicographiques sont de plus structurées sous forme de blocs. Mel'čuk *et al.* (1995) propose de hiérarchiser et d'ordonner les composantes sémantiques en les six blocs suivants :

1. **Les composantes ordinaires de la définition :** le sens de la lexie définie hérite des sens de ces composantes ;
2. **La composante générique** correspond au genre proche de la lexie définie.
3. **Les composantes faibles** sont neutralisées dans certains contextes bien définis. Par exemple, pour la lexie ⌜ÉTUDIANT⌝, la composante sémantique (de sexe masculin) est neutralisée lorsque la lexie est employée au pluriel (ex : *les étudiants de cette classe sont motivés*) ou de manière générique (ex : *l'étudiant d'aujourd'hui doit réapprendre à apprendre*).
4. **Les composantes optionnelles** peuvent être neutralisées par un contexte contradictoire explicite. Par exemple, pour la lexie ⌜PROFESSEUR⌝, la composante sémantique (employé par un établissement d'enseignement) est neutralisée dans le contexte suivant : *Le professeur de piano à domicile*.
5. **Les contraintes sur les variables** spécifient le type sémantique d'une PosASém.
6. **La partie présuppositionnelle** spécifie ce qui reste vrai lorsque la lexie est employée avec une négation. Par exemple, si X n'aide pas Y à faire Z, alors il reste vrai que Y fait ou essaie de faire Z.

Mel'čuk *et al.* (1995) identifie les différents blocs à l'aide de marqueurs typographiques, en particulier, les composantes faibles et optionnelles sont marquées entre parenthèses. La partie présuppositionnelle précède, devant le symbole 'ǁ', le reste de la définition. Voici un exemple inspiré de (Mel'čuk *et al.*, 1995, p.106) :

personne X aide$_{\mathbf{1}}$ personne Y à activité Z-er par ressources W = (personne Y faisant ou essayant de faire activité Z, ǁ personne X utilise ses ressources W dans le but de faciliter activité Z pour personne Y)

D'autres travaux en lexicologie explicative et combinatoire ont cherché à préciser la structure des définitions lexicographiques pour certaines catégories de lexies. Iordanskaja et Mel'čuk (1990) se sont par exemple focalisés sur la structure des définitions des lexies d'émotion. Ils justifient la nécessité d'utiliser des composantes plus précises, comme la caractérisation, la cause, ou l'effet de l'état émotionnel. Mel'čuk et Wanner (2001) se sont intéressés aux noms d'actes de langage. Ici encore des composantes plus précises sont proposées, telles que le but de l'acte de langage.

Wanner (2003) reprend et synthétise ces recherches, tout en justifiant de la nécessité pour les applications en TALN d'aboutir à une structuration aussi formelle que possible des définitions lexicographiques. L'auteur privilégie donc une visualisation sous forme de liste à plusieurs niveaux, chaque item représentant un type de composante ou de sous-composante sémantique.

Le projet BDéf (Altman et Polguère, 2003) avait pour but de compiler une petite base de données de définitions aussi formalisées que possibles, dans le cadre du projet DiCo. La composante générique d'une définition lexicographique est choisie dans un ensemble d'étiquettes sémantiques, chaque étiquette sémantique suggérant un ensemble de composantes habituellement utilisées pour les lexies ayant cette étiquette (Polguère, 2007).

Dans les dictionnaires terminologiques DicoInfo, DicoEnviro et Juridico, les définitions lexicographiques ont la particularité suivante : les PosASém sont annotées par des rôles sémantiques choisis dans un petit ensemble (ex : agent, patient, instrument). Ainsi, la méthodologie utilisée s'inspire de FrameNet (Fillmore, 1977; Fillmore *et al.*, 2003).

Finalement, le projet Definiens (Barque et Polguère, 2013; Barque *et al.*, 2010) utilise cette hiérarchie d'étiquettes sémantiques, et un petit ensemble de relations sémantiques pour étiqueter les composantes périphériques des définitions. La définition est représentée dans un format XML : un élément `CC` (pour composante centrale) ayant pour valeur de l'attribut `label` l'étiquette sémantique de la lexie, et un ou plusieurs éléments `PC` (pour composante périphérique) ayant pour valeur de l'attribut `label` une relation sémantique suggérée ou non par l'étiquette sémantique. Par exemple, une représentation possible de la définition de $^{(}\text{peigne}_{\mathbf{B.2.d}}^{)}$ serait la suivante :

```
<CC label="outil" >outil de tissage</CC>
<PC role="utilisation" >qu'une personne X utilise pour démêler<sub>2</sub> les fibres
d'un objet Y</PC>
```

On retrouve ce choix de formalisation dans le nouveau projet RELIEF (Lux-Pogodalla et Polguère, 2011).

1.3.2.2 Représentation sous la forme d'une RSém

Une représentation d'une définition en langue naturelle, aussi formalisée soit-elle, reste une linéarisation de la RSém qui encode le sens de la lexie définie. Mel'čuk *et al.* (1995) proposait donc de représenter les définitions lexicographiques directement sous forme de RSém. La figure 1.8 représente la définition de $^{\lceil}\text{PEIGNE}_{\mathbf{B.2.D}}^{\rceil}$:

- le nœud étiqueté $^{(}\text{outil}^{)}$ est marqué comme communicativement dominant. Il s'agit du genre proche.
- d'autres nœuds sont marqués X, Y,... et correspondent aux PosASém de $^{\lceil}\text{PEIGNE}_{\mathbf{B.2.D}}^{\rceil}$, donc les PosA de $^{(}\text{peigne}_{\mathbf{B.2.d}}^{)}$.

Jusqu'à présent, ce type de représentation n'est que marginalement utilisé pour des raisons pratiques. Citons Mel'čuk *et al.* (1995, p.73) :

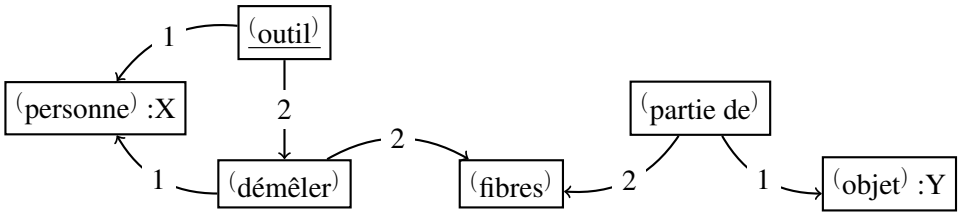


FIGURE 1.8 – Représentation de la définition lexicographique de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ sous la forme d'une représentation sémantique

> Même si un [dictionnaire avec les définitions lexicographiques représentées sous la forme de RSém] est, à notre avis, une réalisation indispensable, nous croyons qu'il reste encore, pour le moment, un idéal hors d'atteinte.

Une des raisons invoquées est la difficulté pratique de rédiger – à la main – un ensemble cohérent de telles représentations sémantiques. Cependant, Polguère (1990), lorsqu'il précise le système de règles dans la composante sémantique du modèle Sens-Texte dans une perspective de génération automatique de textes, justifie l'importance de représenter les définitions lexicographiques sous forme d'une équivalence entre :

- une RSém représentant la définition d'une lexie ;
- une RSém représentant (L) et sa structure actancielle.

Les deux règles de paraphrase ainsi élaborées devraient permettre de transformer une RSém en une RSém équivalente, en remplaçant une unité sémantique par sa paraphrase, et vice-versa. Ces opérations élémentaires au niveau sémantique sont fondamentales pour faciliter le choix d'une lexicalisation appropriée pour un ensemble de RSém équivalentes.

1.3.2.3 Positionnement de notre travail

Dans cette thèse, nous allons donc proposer une formalisation différente des définitions lexicographiques, aussi proche que possible des RSém. Nous justifierons dans la partie II qu'une RSém telle que celle illustrée par la figure 1.8 possède également quelques limites pour la représentation des définitions lexicographiques. Par exemple, nous ne pouvons pas y identifier les composantes faibles, optionnelles, ou présuppositionnelles de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉.

Nous justifierons d'une part que ces représentations sont une instanciation prototypique de la définition lexicographique de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ sous la forme d'une RSém, tout comme une définition formalisée en langue naturelle est une instanciation prototypique de la définition lexicographique de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ sous forme de texte.

Nous nous intéresserons particulièrement à la question de la représentation des composantes optionnelles dans les définitions lexicographiques, et aux questions de cohérence de la définition qui en découlent. Par exemple dans la représentation ci-dessus, si la PosA 2 de (outil) est obligatoire, et si la PosA 1 de (démêler) est obligatoire également, alors la PosA 1 de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ ne peut qu'être obligatoire. Par contre, nous n'aborderons pas la représentation des composantes présuppositionnelles, ni la représentation de la polysémie lexicale.

Conclusion

Ainsi, en tant que théorie linguistique hautement précise du point de vue descriptif, la TST tient une place centrale dans le champ de la sémantique lexicale. La richesse de description des fonctions lexicales et des définitions lexicographiques fait sa force, mais elle a aussi des limites : (i) l'absence de représentation formelle et d'axiomatisation ; (ii) la faible couverture de la langue ; et (iii) le peu d'applications en TALN.

En particulier, la représentation des définitions lexicographiques a été peu étudiée, et se fait désirer pour en faire bénéficier les applications en TALN. Nous proposons donc d'étudier ce point précis dans cette thèse. Récapitulons les frontières de notre travail :

- Nous nous focalisons sur l'étude et la représentation du niveau de représentation sémantique. Nous aborderons un peu le niveau de représentation syntaxique, et nous n'étudierons pas les niveaux de représentation plus surfaciques.
- Nous n'aborderons pas la représentation des fonctions lexicales, et le lien avec FrameNet.
- Nous n'étudierons pas les structures auxiliaires des représentations linguistiques (ex : les structures communicationnelles).
- Nous ne nous attarderons pas sur la représentation ni sur le fonctionnement des modules de correspondance du modèle sens-texte.
- Notre étude se focalise sur la zone sémantique dénotationnelle du DEC, c'est-à-dire là où le sens linguistique de l'unité lexicale est décrit.
- Nous n'aborderons pas la représentation des composantes faibles, de la partie présuppositionnelle, ni de la polysémie lexicale dans les définitions lexicographiques.

Chacun de ces points fait partie des perspectives de ce travail, et nous introduirons les travaux futurs les concernant dans le chapitre 11.

Nous avons ensuite mis en lumière quelques limites dans la conceptualisation de la TST. Nous nous proposons de lever ces limites dans la partie II de ce mémoire. En particulier :

- Nous proposerons de distinguer les unités linguistiques décrites dans le dictionnaire, et celles décrites dans les représentations d'énoncés.
- Nous proposerons une extension de la notion de situation linguistique. Nous nous intéresserons à la notion de participant optionnel, et nous justifierons l'introduction de la notion de participant interdit.
- Nous étendrons la conceptualisation des définitions lexicographiques, notamment puisque nous ne pouvons pas identifier les PosA optionnelles sur une RSém.

Le chapitre suivant précise la méthodologie que nous adopterons pour notre démarche d'ingénierie des connaissances. Nous verrons que l'extension de la conceptualisation de la TST est une première étape essentielle.

CHAPITRE 2

Propositions et standards de l'ingénierie des connaissances pour l'informatisation du DEC

__D__éfinir une ontologie pour la représentation des connaissances, c'est définir, pour un domaine et un problème donnés, la signature fonctionnelle et relationnelle d'un langage formel de représentation et la sémantique associée.
Bachimont (2000, p.3)

Sommaire

Introduction

En parallèle aux travaux de formalisation déjà menés sur le DEC, nous nous proposons de représenter les prédicats linguistiques, les représentations linguistiques, et les définitions lexicographiques au sens de l'Ingénierie des Connaissances (IC), en restant au plus proche des standards de la Représentation des Connaissances (RC). Le terme *représentation* ne signifie donc pas seulement *formaliser*, mais également *rendre non ambigu*, et *opérationnaliser*, i.e., *permettre d'effectuer des opérations logiques* (ex : manipulation, interrogation, déduction).

Ce chapitre illustre dans un premier temps ce que l'ingénierie des connaissances pourrait apporter aux linguistes de la TST (§2.1). Nous détaillons ensuite la méthodologie que nous adoptons pour notre étude (§2.2). Deux formalismes semblent les plus adaptés pour la représentation des prédicats linguistiques et des définitions lexicographiques : les Logiques de Description, et le formalisme des Graphes Conceptuels (GC). Nous les présenterons tour à tour (§2.3). Finalement, nous présenterons les derniers standards en représentation des connaissances lexicales, et préciserons les enjeux de leur réutilisation (§2.4).

2.1 Propositions pour une ingénierie des connaissances du DEC

Lorsque Mel'čuk *et al.* (1995, p.208) promeut l'informatisation du DEC (cf. la citation mise en exergue du chapitre 1), il met en évidence l'émergence de besoins spécifiques :

- la recherche intelligente d'information ;
- la mise à jour et la vérification automatique ;
- la déduction logique.

L'émergence de tels besoins existe en réalité lors de la production de connaissances par tout domaine de spécialité. L'IC vise justement à répondre à ces besoins. Il s'agit de permettre par exemple de représenter, de manipuler, d'échanger, d'interroger, ou de raisonner avec les connaissances. Les besoins envisagés par Mel'čuk seraient donc couverts.

Nous présentons dans cette section une liste de scénarios envisagés pour la représentation des prédicats linguistiques et des définitions lexicographiques. Chacun de ces scénarios implique Pierre, un lexicographe de la théorie Sens-Texte, qui doit renseigner la zone de définition de l'article $\lceil \text{PEIGNE}_{\mathbf{B.2.D}} \rceil$.

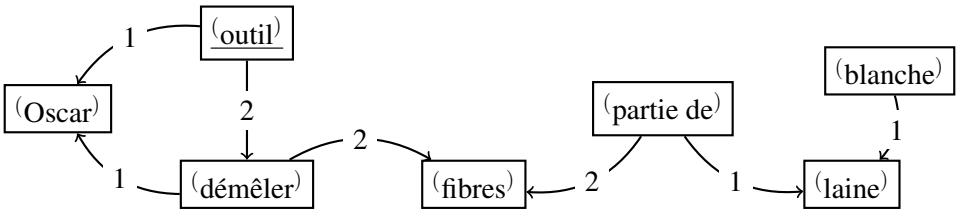
Scénario n°1 : recherche intelligente d'information. Selon les critères linguistiques, Pierre a déterminé que $\lceil \text{PEIGNE}_{\mathbf{B.2.D}} \rceil$ a une PosASém qui représente l'utilisateur (de type ⟨humain⟩), et une PosASém qui représente l'objet dont on veut démêler les fibres (de type ⟨objet⟩). Il souhaite obtenir une liste de lexies qui ont une structure actancielle similaire. Il peut interroger la base de connaissances en posant la question : "Quelles lexies sont des prédicats sémantiques qui impliquent un actant du type ⟨humain⟩, et un actant du type ⟨objet⟩ ?" [1]. Nous étudierons la possibilité d'interroger et de partager les connaissances de la TST dans le chapitre 10.

Scénario n°2 : élaboration d'une définition lexicographique par manipulation de RSém. Dans ce scénario, Pierre élabore de proche en proche la définition de ⟨peigne$_{\mathbf{B.2.d}}$⟩ sous la forme d'une RSém illustrée par la figure 1.8. Pierre choisit ⟨outil⟩ comme genre proche de ⟨peigne$_{\mathbf{B.2.d}}$⟩. Dans une fenêtre d'édition de graphe, Pierre visualise alors la RSém qui représente la structure actancielle de ⟨outil⟩ : un nœud de type ⟨outil⟩, qui gouverne trois autres nœuds, respectivement de type ⟨humain⟩,

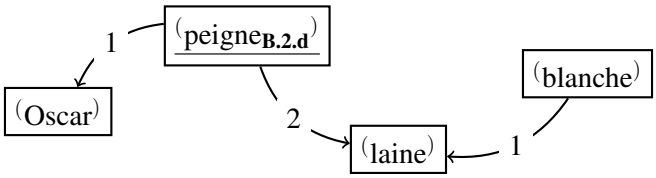
1. Cette requête est inspirée de celle proposée en exemple par Mel'čuk *et al.* (1995, p.208)

⟨activité⟩, et ⟨profession⟩. Pierre peut alors indiquer qu'un peigne est un outil particulier qui sert à démêler. En spécialisant le type du nœud correspondant, Pierre découvre que ⟨démêler⟩ a une structure actancielle : le nœud gouverne deux nouveaux nœuds, respectivement de type ⟨humain⟩, et ⟨fibres⟩. Pierre peut alors préciser que la première PosA de ⟨démêler⟩ correspond à la première PosA de ⟨outil⟩. Dans ce scénario, la formalisation du DEC permet à Pierre de s'assurer que les types des unités sémantiques manipulées sont effectivement décrits dans le DEC, et qu'ils respectent leur structure actancielle. Ce scénario sera détaillé dans le chapitre 5.

Scénario n°3 : raisonnement dans une RSém. Considérons les deux RSém illustrées sur la figure 2.1. A l'aide d'un outil de raisonnement, Pierre peut vérifier que les deux RSém sont en réalité équivalentes (i.e., représentent des paraphrases). Nous étudierons la possibilité de raisonner avec les connaissances de la TST dans le chapitre 9.



(a) Représentation sémantique dont voici une expression textuelle : *L'outil d'Oscar qu'il utilise pour démêler les fibres de la laine blanche*.



(b) Représentation sémantique dont voici une expression textuelle : *Le peigne d'Oscar pour la laine blanche*.

FIGURE 2.1 – Raisonnement dans une RSém : Illustration de deux représentations sémantiques équivalentes.

2.2 Méthodologie de l'ingénierie des connaissances

La tâche de l'ingénierie des connaissances, telle que Bachimont (2000) la définit, est de modéliser formellement un problème pour lequel les seules connaissances dont on dispose sont de nature linguistique ou cognitive. La sortie de ce processus de modélisation est un formalisme de représentation de connaissances, qui peut alors être mis en œuvre par des programmes.

Bachimont (2000) propose une méthode en trois étapes pour passer de l'expression linguistique des connaissances à une représentation formelle et calculable. Cette section est organisée selon ces trois étapes :

- définir la conceptualisation du domaine (§2.2.1) ;
- choisir un formalisme de représentation des connaissances (§2.2.2) ;
- opérationnaliser le formalisme choisi (§2.2.3).

Notons que d'autres méthodologies d'ingénierie des connaissances existent, certaines utilisant justement des techniques de TALN pour construire (semi-)automatiquement des ontologies à partir de textes (Aussenac-Gilles *et al.*, 2000, 2013). Nous souhaitons capturer autant de sémantique logique que possible de la TST, et préférons donc adopter une démarche manuelle.

2.2.1 Définir les primitives : l'engagement sémantique

Nous devons, à partir des connaissances de TST, définir les primitives du domaine. Le modèle Sens-Texte (modèle ST) étant déjà hautement conceptualisé, nous pourrions penser qu'il suffit de reprendre les primitives du modèle. Cependant, nous devons garder en tête qu'un formalisme logique devra être construit au dessus de cette conceptualisation. La difficulté est donc de choisir ou construire des primitives adaptées à une formalisation ultérieure.

En d'autres termes, nous devons identifier, dans la conceptualisation de la TST, ce qui serait problématique pour en représenter les connaissances. Nous proposerons alors d'en étendre la conceptualisation. En procédant de la sorte, nous créons un objet frontière de type forme standardisée, au sens de Star et Griesemer (1989), entre la TST et le domaine de la représentation des connaissances.

La partie II de ce mémoire consiste justement en l'extension de la conceptualisation de la TST. Les chapitres 3 et 4 étudient respectivement la conceptualisation des prédicats linguistiques et des définitions lexicographiques. Quant au chapitre 5, il évalue la perception de l'extension de la conceptualisation de la TST grâce à la participation d'une équipe de lexicographes à une expérience permettant de savoir dans quelle mesure ils accepteraient de manipuler directement les primitives que nous avons introduites pour leur tâche de définition des lexies dans le DEC.

Nous avons déjà vu un exemple d'un tel choix dans la section 1.2.1.3. Le terme *lexie* dans la TST cache en réalité deux sens : la lexie décrite dans le dictionnaire, et la lexie représentée dans une RSém en usage. Afin de faciliter la formalisation des connaissances de la TST, nous proposons de différencier les primitives en désambiguïsant les deux termes. Nous parlerons donc respectivement de *type de lexie*, et de *lexie*. Cet exemple sera repris et détaillé dans le chapitre 3.

2.2.2 Formaliser les connaissances : l'engagement ontologique

Une fois que la conceptualisation de la TST est étendue, nous passons à l'étape de formalisation des connaissances. Il s'agit de concevoir une ontologie du domaine. Citons la définition de Guarino (1998), qui reprend et enrichit celle de Gruber (1995) :

> An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e., its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.

Nous résumerons en français par l'expression suivante :

Une ontologie est la formalisation d'une conceptualisation d'un domaine.

2.2.2.1 Critères de choix d'un formalisme de RC

La conception d'une ontologie est intimement liée au formalisme de représentation des connaissances que l'on choisit. Pour évaluer le choix du formalisme de RC choisi, nous utiliserons les cinq critères de conception d'une ontologie proposés par Gruber (1995) [2].

Clarté. La définition d'un concept doit faire passer le sens voulu du terme, de manière aussi objective que possible (indépendante du contexte). Une définition doit de plus être complète (c'est-à-dire définie par des conditions à la fois nécessaires et suffisantes).

Par exemple, et pour faire suite à l'exemple de la section 1.3.1.2, nous préférerons qu'un prédicat sémantique soit représenté par un seul concept dans une ontologie, plutôt que par une multitude de prédicats logiques en logique du premier ordre.

De la même manière, nous préférerons que la définition lexicographique d'une lexie $\ulcorner \mathrm{L} \urcorner$ soit représentée par une seule règle claire, plutôt que par un ensemble abscons d'axiomes. Nous verrons que c'est une raison qui nous pousse à abandonner les logiques de description.

Cohérence. Rien qui ne puisse être inféré de l'ontologie ne doit entrer en contradiction avec les définitions des concepts (y compris celles qui sont exprimées en langue naturelle).

Par exemple, nous étudierons dans le chapitre 3 la possibilité d'organiser les prédicats sémantiques dans une hiérarchie au sein de laquelle les PosA sont héritées. Nous étendrons la conceptualisation de la TST de sorte que l'ontologie ait de bonnes chances d'être cohérente.

Extensibilité. Les extensions qui pourront être ajoutées à l'ontologie doivent être anticipées. Il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie.

Ce critère nous pousse par exemple à préférer un formalisme de RC qui soit monotone.

Minimalité de la déformation d'encodage. Une déformation d'encodage a lieu lorsque la spécification influe sur la conceptualisation (un concept donné peut être plus simple à définir d'une certaine façon dans un langage d'ontologie donné, bien que cette définition ne corresponde pas exactement au sens initial). Ces déformations doivent être évitées autant que possible.

2. Nous adaptons ici la traduction simplifiée de Wikipedia - fr.wikipedia.org/wiki/Ontologie_(informatique)

Minimalité de l'engagement ontologique. Le premier but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière complète. Contrairement aux bases de connaissances par exemple, nous n'attendons pas d'une ontologie qu'elle soit en mesure de fournir systématiquement une réponse à une question arbitraire sur le domaine. Selon Gruber, *l'engagement ontologique peut être minimisé en spécifiant la théorie la plus faible (celle permettant le plus de modèles) couvrant un domaine ; elle ne définit que les termes nécessaires pour partager les connaissances consistantes avec cette théorie*.

Ce dernier critère est étroitement lié avec le compromis bien connu entre expressivité et complexité. En effet, plus un formalisme de représentation des connaissances est expressif, et plus le raisonnement au sein de ce formalisme sera complexe. Par exemple, le scénario de recherche intelligente d'information cité dans la section 2.1 possède deux aspects : que veut-on pouvoir chercher (expressivité) ? et doit-on limiter l'intelligence du système pour qu'il réponde en un temps raisonnable (complexité) ?

2.2.2.2 Formalismes candidats pour la représentation des connaissances du DEC

Ainsi, la conception d'une ontologie est intimement liée au formalisme de représentation des connaissances que l'on choisit.

Pour que l'engagement ontologique soit correct, et pour avoir une possibilité de raisonner dans le formalisme, notre étude se limite à l'utilisation de fragments décidables de la logique du premier ordre.

Au premier abord, deux formalismes de RC existants semblent adaptés à la représentation des connaissances des prédicats linguistiques et des définitions lexicographiques :

- La famille des formalismes du Web Sémantique. Il s'agit du standard de facto de la représentation des connaissances sur lequel le Web des données [3] est basé.
- Le formalisme des Graphes Conceptuels. Ce formalisme a été introduit vers le début des années 1980 par J. Sowa, qui avait originellement pour but la représentation des connaissances linguistiques à des fins de traitement automatisé. Sowa s'est inspiré entre autres des mêmes travaux que les fondateurs de la TST, ceux de Tesnière (1959).

Nous étudierons chacun de ces formalismes dans la partie III. Aucun de ces formalismes ne s'avérera adapté, ce qui justifiera l'introduction d'un nouveau formalisme de représentation des connaissances nommé formalisme des Graphes d'Unités.

2.2.3 Utiliser la formalisation : l'engagement computationnel

La dernière étape de la méthodologie de l'IC est d'opérationnaliser le formalisme. Il s'agit de déterminer dans quelle mesure le formalisme choisi peut être utilisé pour les scénarios d'utilisation envisagés.

La partie IV définit l'engagement computationnel du formalisme des Graphes d'Unités. Nous nous concentrerons en particulier sur deux points :

- Le chapitre 9 étudie la déduction dans le formalisme des Graphes d'Unités. Nous lui attribuerons une sémantique logique, et déterminerons des conditions suffisantes pour que le raisonnement y reste décidable.
- Le chapitre 10 étudie le lien entre le formalisme des Graphes d'Unités et les formalismes du Web Sémantique, pour permettre l'interrogation et le partage des connaissances sur le Web des données.

3. Le Web des données est une initiative du W3C, très active aujourd'hui. http://linkeddata.org

2.3 Standards de représentation des connaissances

Nous nous intéressons donc plus particulièrement à deux formalismes existants pour la représentation des connaissances des prédicats linguistiques et des définitions lexicographiques. Nous proposons dans cette section une vue d'ensemble de ces deux formalismes :

- La famille des formalismes du Web Sémantique (§2.3.1) ;
- Le formalisme des Graphes Conceptuels (§2.3.2).

2.3.1 Les formalismes du Web Sémantique

Ces dernières années ont vu un engouement mondial pour le Web Sémantique, dont l'initiative Web des données est aujourd'hui une première vague de déploiement. Le World Wide Web Consortium (W3C) centralise les efforts de standardisation (on parle de *recommandations* du W3C) pour les formalismes sous-jacents à ces évolutions.

2.3.1.1 La pyramide des recommandations du Web Sémantique - Web des Données

La figure 2.2 représente la pyramide de ces recommandations telle qu'elle était pensée pour le Web Sémantique. La brique de base de cette pyramide spécifie que chaque ressource dont on veut parler doit être identifiée par une URI [4]. Ensuite, ces recommandations proposent :

- une structure de données unifiée (graphes RDF [5]) ;
- le protocole de requête et mise à jour correspondant (SPARQL [6]) ;
- des fragments de logique avec différentes expressivités pour capturer la sémantique formelle des schémas de données (RDFS [7], OWL [8]) ;
- un langage de règles qui offre une possibilité supplémentaire pour représenter les inférences sur les données (RIF [9]) ;

Dans sa note sur le Web des Données, Berners-Lee (2006) introduit quatre règles simples :

1. Utiliser des URI pour nommer les ressources ;
2. Utiliser des HTTP-URI pour que les gens puissent obtenir une représentation de ces ressources ;
3. Lorsque quelqu'un cherche à obtenir une représentation d'une ressource, lui renvoyer des informations utiles, en utilisant les standards du Web Sémantique ;
4. Inclure des liens vers d'autres URI, pour qu'il puisse découvrir plus de choses.

4. URI - Uniform Resource Identifier, RFC 3986
5. RDF - Resource Description Framework, cf., http://w3.org/RDF/
6. SPARQL, cf., http://www.w3.org/TR/sparql11-overview/
7. RDFS - RDF Schema, cf., http://www.w3.org/TR/rdf-schema/
8. OWL - Web Ontology Language, cf., http://www.w3.org/TR/owl2-overview/
9. RIF - Rule Interchange Format, cf., http://www.w3.org/TR/rif-overview/

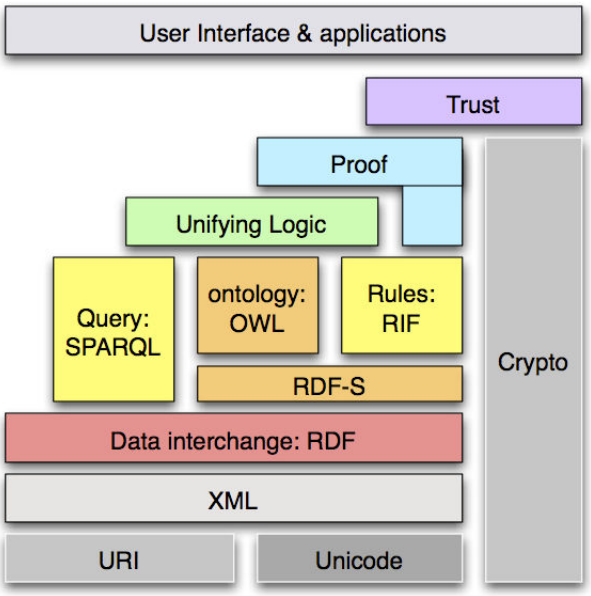


FIGURE 2.2 – La pile des recommandations du Web Sémantique (source : http://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/#(19)).

L'URI est donc la pierre angulaire du Web des données. A l'époque du Web communautaire (Web 2.0), les URL servaient à désigner des ressources du Web présentes sur le Web. Avec les Uniform Resource Identifier (URI), on commence à identifier, sur le Web, des ressources quelconques du monde. Une HTTP-URI est une URI qui identifie un emplacement sur le Web, où on peut obtenir une représentation Web de la ressource identifiée. Ainsi, les URI ne sont pas réservées à l'identification de documents sur le Web, mais peuvent être utilisées pour désigner n'importe quelle ressource, y compris les objets du monde réel, et pour nos besoins les lexies, et leur sens. Nous pouvons citer ici les travaux de Delaforge *et al.* (2012) et de Monnin (2013), qui étudient les notions de ressource sur le Web et d'URI d'un point de vue philosophique.

La tendance actuelle, incarnée par l'initiative Web des données, est d'encourager la publication de données structurées sur le Web, quelle que soit leur sémantique. Le Web Sémantique proposera alors un moyen standard pour échanger et lier différentes sémantiques. Entre temps, le reste de la pile continue à faire l'objet de travaux, comme la récente recommandation PROV [10] (pour *provenance*) qui s'insère dans la brique *trust*.

Ainsi, chaque application peut adopter une sémantique formelle qui lui est propre, tout en profitant des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des données. Nous proposons donc d'adopter les recommandations de base du Web Sémantique pour l'opérationnalisation de notre formalisme (cf., chapitre 10), et nous étudierons dans quelle mesure la sémantique logique de OWL est adaptée à nos besoins (cf., chapitre 6).

10. PROV - 2013 - http://www.w3.org/TR/prov-overview/

2.3.1.2 RDF et SPARQL

Le modèle des données Resource Description Framework (RDF) représente des multigraphes étiquetés et orientés. RDF permet la description et l'interconnexion de ressources qui peuvent être anonymes ou identifiées par une URI. En RDF, la brique de base de connaissance est le triplet, de la forme ⟨sujet, prédicat, objet⟩, le prédicat étant une Propriété RDF. Le modèle de données RDF admet plusieurs syntaxes concrètes (RDF/XML, turtle, RDFa, N-Triples, JSON-LD, etc.).

La seule sémantique logique apportée par RDF est la suivante : si dans deux graphes présents à deux bouts du Web se trouvent deux ressources RDF qui possèdent la même URI, alors ces ressources ont toutes les deux la même image dans l'union des deux graphes.

Finalement, SPARQL Protocol and RDF Query Language (SPARQL)[11] est le langage de requêtes et de mise à jour correspondant.

2.3.1.3 Plus de sémantique formelle avec RDFS et OWL

RDF Schema (RDFS) et Web Ontology Language (OWL) ajoutent de la sémantique formelle aux graphes RDF.

RDFS introduit quelques axiomes logiques, qui permettent de définir le squelette taxonomique d'une ontologie formelle légère :

- une hiérarchie de classes à l'aide de l'axiome `subClassOf` ;
- une hiérarchie de propriétés à l'aide de l'axiome `subPropertyOf` ;
- le domaine et l'image d'une propriété à l'aide des axiomes `domain` et `range`.

OWL, quant à lui, est un métalangage d'ontologie qui étend RDFS et permet de définir des ontologies sur la base des Logiques de Description (LD) (cf., Rudolph, 2011; Baader *et al.*, 2003). Les LD tiennent leur fondation des réseaux sémantiques (Quillian, 1968) et des systèmes à base de frames (Minsky, 1975). Elles proposent une décomposition de la logique du premier ordre en un ensemble de constructeurs de classes (ex : intersection, union, restriction de cardinalité), et d'axiomes (ex : classes équivalentes, disjointes). Plusieurs fragments de la logique du premier ordre sont alors définis, en fonction du sous-ensemble de ces axiomes qui est autorisé.

La figure 2.3 illustre une liste non exhaustive d'axiomes et de constructeurs de classes disponibles dans OWL2.

Ainsi l'idée générale est la suivante : l'ontologue choisit le sous-ensemble d'axiomes nécessaire à l'expressivité dont il a besoin. Les résultats sur les LD garantissent alors que la complexité calculatoire de n'importe quel combinaison de ces axiomes est bornée. On verra que seules quelques restrictions globales sont imposées. En particulier, la hiérarchie des propriétés doit être régulière (voir par exemple, Rudolph, 2011, p. 12).

11. SPARQL est un acronyme récursif.

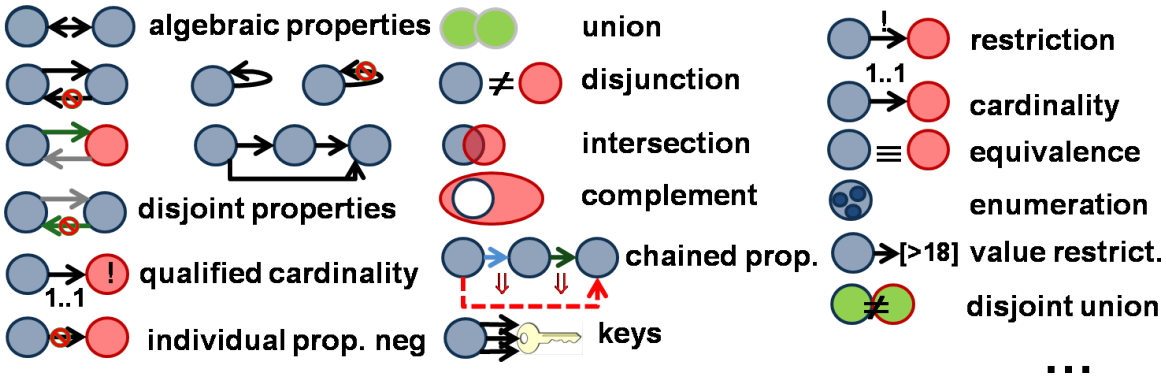


FIGURE 2.3 – Une liste non-exhaustive des axiomes et constructeurs de classes disponibles dans OWL2 (source : inspiré de Gandon, *OWL in one*, http://fabien-gandon.blogspot.fr/2009/05/owl-in-one.html).

2.3.2 Le formalisme des Graphes Conceptuels

Le formalisme des Graphes Conceptuels (GC), introduit par Sowa (1984), et formalisé par Chein et Mugnier (1992, 2009), tient ses origines des réseaux sémantiques (Quillian, 1968). Cependant, Sowa avait originellement pour but la représentation des connaissances lexicales à des fins de traitement automatisé, et il s'est inspiré entre autres des mêmes travaux que les fondateurs de la TST, à savoir ceux de Tesnière (1959) (cf., Sowa, 1989).

Les GC sont en outre présentés comme permettant de représenter spécifiquement la sémantique lexicale des lexies dans le dictionnaire (Sowa, 1992). Nous allons donc étudier dans quelle mesure les GC permettent de représenter les connaissances du DEC avec la finesse descriptive attendue.

2.3.2.1 Les GC basiques, et premiers défis

Dans leur version basique (Chein et Mugnier, 1992), les GC représentent des instances typées interconnectées par des relations *n*-aires également typées. Les types de concept et de relation sont décrits dans un *support*, et sont organisés en plusieurs hiérarchies : une hiérarchie pour les types de concept, et une hiérarchie pour chaque ensemble de type de relation de même arité. Chein et Mugnier (1995) ont introduit l'idée que les GC peuvent être vus comme des graphes bipartis, dont un ensemble de nœuds est l'ensemble des concepts, et dont l'autre ensemble de nœuds est l'ensemble des relations.

Voici donc la première difficulté à laquelle nous faisons face dans notre étude. Toute unité linguistique pouvant être dotée d'une structure actancielle, il n'y a pas de distinction claire entre les concepts et les relations dans la TST. La simple RSém de la figure 1.2 illustre bien ce problème : l'unité sémantique de type ⟨œuvre⟩ dépend d'une unité sémantique de type ⟨devenir⟩, qui dépend elle-même d'une unité sémantique de type ⟨causer⟩.

Nous étudierons donc la possibilité de travailler avec des relations réifiées, c'est-à-dire que chaque unité linguistique sera représentée à la fois par un concept et par une relation.

2.3.2.2 Intérêt des extensions du modèle de base

Plusieurs extensions du modèle originel ont été proposées. La plupart ont été envisagées par Sowa (1989), et formalisées ensuite. Voyons celles qui sont les plus intéressantes pour notre étude.

Tout d'abord, la version simple des GC ajoute deux notions liées mathématiquement, et intéressantes pour les représentations linguistiques : les liens de coréférence, et les types conjonctifs. Les liens de coréférence ont été introduits par Sowa (1984) pour représenter les coréférences anaphoriques. Les types conjonctifs ont ensuite été introduits par Cao *et al.* (1997), et Chein et Mugnier (2004) ajoutèrent la notion de type conjonctif banni, qui ne peuvent être instanciés. Par exemple, nous pourrions ainsi représenter le fait qu'une lexie ne peut être à la fois marquée par les grammatèmes *plur* et *sing*.

Ensuite, les notions de règles et de contraintes ont été introduites. Sowa (1989) avait introduit les λ-abstractions de GC. Une règle exprime une connaissance de la forme : "si A est vrai, alors B est vrai". Il s'agit d'un couple de λ-graphes qui partagent une frontière : une hypothèse, et une conclusion. Si l'hypothèse est vérifiée, alors on peut ajouter la conclusion dans la base de faits. Les contraintes, elles, peuvent être vues comme l'expression de connaissances de la forme : "si A est vrai, alors B doit être vrai" (contrainte positive), ou "si A est vrai, alors B ne doit pas être vrai" (contrainte négative). Citons notamment les travaux de Baget *et al.* (1999a,b). S'en sont suivis des travaux sur les algorithmes de raisonnement, et l'identification de différentes variantes de la famille des GC, suivant leur expressivité et leur complexité (cf., Baget, 2001; Baget et Mugnier, 2002). Dans notre étude, les règles et les contraintes semblent permettre une expression plus naturelle des connaissances des lexicographes.

Enfin, Sowa (1989) introduit la notion de définition de type de concept et de type de relation. Plus tard, Leclère (1998) travailla sur la formalisation et l'opérationnalisation de telles définitions. Ces définitions et la sémantique associée ressemblent fortement aux définitions lexicographiques du DEC. Nous verrons cependant que ces définitions ne peuvent pas être utilisées pour représenter les définitions lexicographiques.

Un dernier aspect intéressant des GC est que des transformations ont été proposées entre eux et les formalismes RDF/S du Web Sémantique (Corby *et al.*, 2000; Baget *et al.*, 2010). Ainsi, si les GC sont plus adaptés que OWL pour la représentation des connaissances du DEC, nous pourrions néanmoins utiliser ces transformations pour réécrire les connaissances en RDF à des fins de partage et d'interrogation des connaissances sur le Web des données.

2.4 Représentation des connaissances lexicales : standards et enjeux

Dans cette thèse, nous développerons un cadre pour un nouveau type de ressources lexicales, adapté à la représentation des définitions lexicographiques de la TST. Nous devrons donc nous poser la question de l'intégration éventuelle aux standards de représentation des ressources lexicales existants.

Cette section offre un aperçu des derniers standards pour la représentation des ressources lexicales (§2.4.1), et détaille les bénéfices que notre projet pourrait retirer de l'utilisation de ces standards (§2.4.2).

2.4.1 Standards pour la représentation des ressources lexicales sur le Web

De manière orthogonale aux études linguistiques et au développement d'applications de TALN, plusieurs projets ont travaillé sur la standardisation des ressources lexicales, comme GENELEX, MULTILEX, EDR, EAGLES, PAROLE, SIMPLE, ISLE. Le comité ISO TC37/SC4 [12] a synthétisé ces travaux dans les années 2000, et a publié le standard LMF (Lexical Markup Framework, ISO 24613:2008) (Francopoulo *et al.*, 2006, 2007). LMF définit un standard abstrait d'architecture pour les dictionnaires interprétables par ordinateur, sous la forme de schémas UML, et propose une DTD pour valider une sérialisation en XML. Le diagramme de classes UML au cœur de LMF est présenté dans la figure 2.4. Par ailleurs, LMF est basé sur un ensemble de standards de plus bas niveau, comme DCR (Data Category Registery, ISO 12620:2009) qui décrit les termes utilisés dans les ressources lexicales et qui a été implémenté par le projet ISOCat [13], ou encore les codes de pays (ISO 3166) et de langues (ISO 639).

Dans le cadre du projet MONNET, le modèle Lemon [14] a été développé sur la base de LMF pour modéliser le lexique avec les formalismes du Web Sémantique. Lemon utilise le vocabulaire LexInfo [15] (Cimiano *et al.*, 2011), qui est aligné avec ISOCat. L'ontologie OWL au cœur de Lemon est illustrée par la figure 2.5.

Deux efforts communautaires récents contribuent au développement du Web des données lexicales :

- Le groupe de travail Open Linguistics [16] (Chiarcos *et al.*, 2011), de l'Open Knowledge Foundation a été fondé en 2010 pour promouvoir l'idée des ressources linguistiques ouvertes et pour proposer des moyens pour les représenter. Un des projets de ce groupe est de développer le nuage des données linguistiques liées [17] ;
- Le groupe communautaire Ontology-Lexica du W3C [18] a été fondé en 2011. Ce groupe a pour objectif de développer des modèles pour la représentation d'informations linguistiques, en particulier les informations linguistiques en rapport avec des ontologies. L'ontologie OWL au cœur du modèle proposé par le groupe communautaire Ontology-Lexica est illustrée par la figure 2.6.

12. ISO TC 37/SC 4 - language resource management - http://www.tc37sc4.org/
13. ISOCat - http://www.isocat.org/
14. lemon - The Lexicon Model for Ontologies - http://lemon-model.net/
15. LexInfo - http://lexinfo.net/
16. OWLG - Open Linguistics Working Group - http://linguistics.okfn.org/
17. Linguistic Linked Open Data cloud - http://linguistics.okfn.org/llod
18. ontolex - Ontology-Lexical Community Group - http://www.w3.org/community/ontolex/

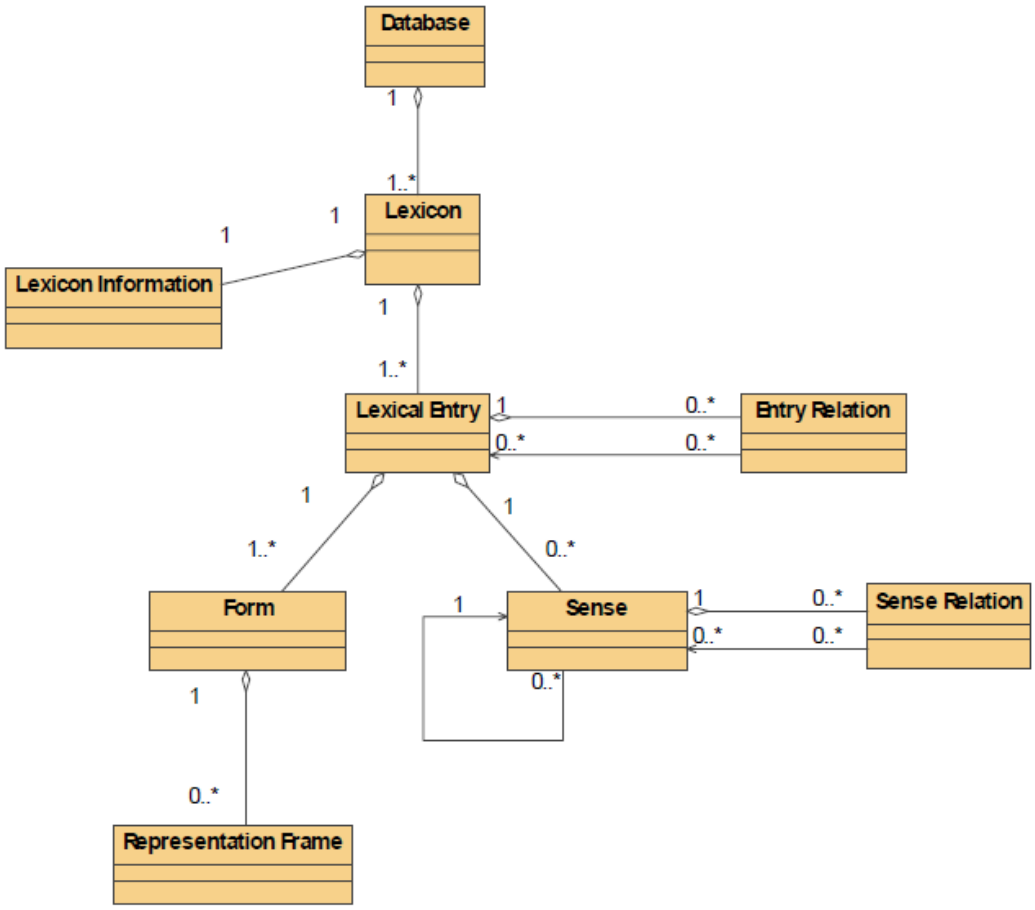


FIGURE 2.4 – Le diagramme de classes UML au cœur de LMF (source : (Francopoulo *et al.*, 2006)).

Enfin, notons la publication de la ressource lexico-sémantique unifiée à grande échelle [19] Uby. Uby est basée sur une implémentation complète de LMF, Uby-LMF (Eckle-Kohler et Gurevych, 2012), et contenait en 2013 10 lexiques dont les unités lexicales sont reliées par les sens : les lexiques anglais WordNet, Wiktionary, Wikipedia, FrameNet, VerbNet, les lexiques allemands Wiktionary, Wikipedia, GermaNet, IMSLex-Subcat, et le lexique multilingue OmegaWiki. Précisons qu'un sous-ensemble des données de Uby, LemonUby, a été exporté dans le modèle Lemon (Eckle-Kohler *et al.*, 2014).

19. Uby - Large-scale unified lexical-semantic resource - https://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/

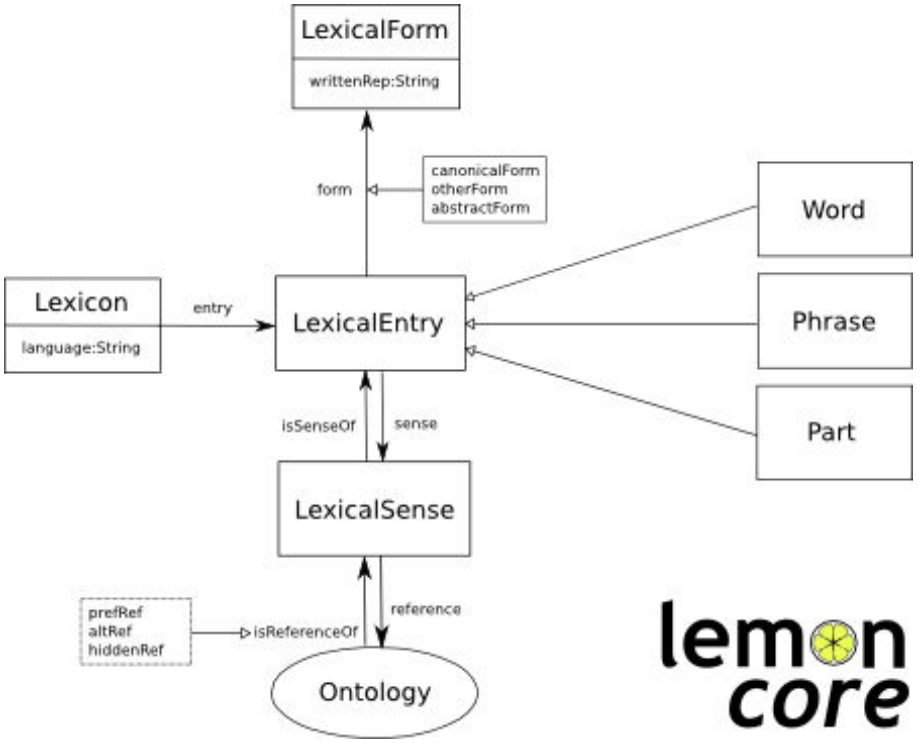


FIGURE 2.5 – L'ontologie OWL au cœur de Lemon (source : http://lemon-model.net).

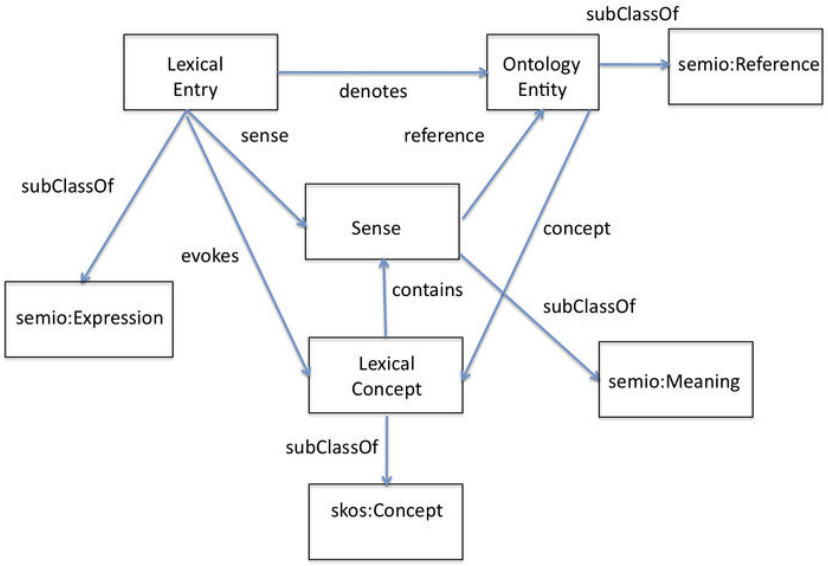


FIGURE 2.6 – L'ontologie OWL au cœur du modèle proposé par le groupe communautaire Ontology-Lexica (source : http://www.w3.org/community/ontolex/wiki/Specification_of_Core_Model).

2.4.2 Enjeux de la réutilisation des standards pour notre projet

Après avoir étendu la conceptualisation de la TST, nous choisirons un formalisme de RC pour représenter les prédicats linguistiques et les définitions lexicographiques. Quel que soit le formalisme choisi pour notre étude, nous proposerons deux transformations entre ce formalisme et le modèle RDF. Nous pourrons ainsi profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances.

Nous nous attacherons en particulier à développer le modèle comme une extension de celui proposé par le groupe communautaire Ontology-Lexica. Nous espérons retirer de cette approche les bénéfices suivants :

- cela faciliterait l'adoption de nos travaux par la communauté de la représentation des connaissances lexicales ;
- cela permettrait de réutiliser plus facilement des données existantes pour peupler notre formalisme ;
- cela accélérerait le développement d'une implémentation qui réutiliserait les outils et librairies de ces standards.

Conclusion

Nous avons donc détaillé notre proposition de formaliser les prédicats linguistiques et les définitions lexicographiques, au sens de l'IC, et en restant au plus proche des standards de la RC.

Nous avons illustré par différents scénarios qu'une telle formalisation serait bienvenue pour les projets de lexicographie Explicative et Combinatoire. Nous nous proposons d'apporter un début de réponse plus précise aux besoins suivants :

- l'élaboration d'une définition lexicographique par manipulation de graphes ;
- la vérification de la satisfaction de contraintes formelles dans le dictionnaire ;
- le raisonnement dans les représentations linguistiques ;
- la publication et la requête des connaissances sur le Web des données.

Pour y arriver, nous adopterons la méthodologie proposée par Bachimont (2000), qui consiste en trois étapes principales, qui définissent l'organisation générale de la suite de ce mémoire.

Partie II. Nous identifierons ce qui dans la conceptualisation de la TST serait problématique pour en formaliser les connaissances, et nous proposerons alors d'en étendre la conceptualisation :

- aux prédicats linguistiques (chapitre 3) ;
- aux représentations linguistiques et aux définitions lexicographiques (chapitre 4).

Partie III. Nous choisirons ensuite un formalisme de RC adapté à l'élaboration d'une ontologie du domaine. Nous nous fonderons sur les critères de Gruber (1995), qui nous permettront de valider ou d'invalider le choix d'un formalisme :

- clarté ;
- cohérence ;
- extensibilité ;
- minimalité de la déformation d'encodage ;
- minimalité de l'engagement ontologique.

Nous avons justifié la candidature des formalismes de RC existants, et nous en avons proposé une vue d'ensemble :

- Les formalismes du Web Sémantique sont le standard *de facto* de la RC sur le Web des données. Quel que soit le formalisme choisi pour notre étude, nous proposerons un alignement avec le modèle proposé par le groupe Ontology-Lexica. Nous étudierons tout de même dans quelle mesure la sémantique logique de OWL et des LD est adaptée.
- Le formalisme des GC présente de grandes similarités avec la TST : il s'inspire des travaux de Tesnière (1959), et on peut y définir des règles, des contraintes, et des définitions de types de concept et de types de relation. Nous étudierons dans quelle mesure ce formalisme est adéquat pour la représentation des prédicats linguistiques et des définitions lexicographiques.

Nous verrons que ni les logiques de description, ni le formalisme des GC, ne sont adaptés aux besoins de notre étude (chapitre 6), et nous justifierons donc la construction d'un nouveau formalisme de représentation des connaissances, dit des *Graphes d'Unités* (chapitres 7 et 8).

Partie IV. Finalement, nous opérationnaliserons le formalisme choisi pour répondre aux différents besoins formulés :

- nous étudierons la possibilité d'y effectuer un (ou des) raisonnement logique (chapitre 9) ;
- nous proposerons une transformation vers les standards de représentation des connaissances lexicales, afin de profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances (chapitre 10).

Deuxième partie

Extension de la conceptualisation de la Théorie Sens-Texte

CHAPITRE 3

Conceptualisation des prédicats sémantiques : nécessité d'introduire un niveau sémantique profond

---

*« **L**es actants sont les êtres ou les choses, qui, à un titre quelconque et de quelque façon que ce soit, même au titre de simples figurants et de la façon la plus passive, participent au procès.*
Tesnière (1959, p.102)

Sommaire

Introduction

Ce chapitre étudie la notion de prédicat sémantique dans la théorie des actants sémantiques de la TST. Nous avons mis en lumière un certain nombre de limites de la conceptualisation actuelle dans le chapitre 1, et nous répondons donc ici à la question de recherche suivante :

Dans quelle mesure la conceptualisation actuelle des prédicats linguistiques doit-elle être étendue pour que sa formalisation ultérieure soit facilitée ?

Par exemple et en premier lieu, nous avons vu que le terme lexie est ambigu, et peut représenter la lexie dans la langue, ou dans les textes (§1.2.1.3). La section 3.1 apporte donc des précisions terminologiques et notationnelles qui permettent de différencier ces deux sens en type et instance. Nous généraliserons ces précisions aux autres unités linguistiques que nous étudions : les unités grammaticales et les unités sémantiques.

Les étiquettes sémantiques, comme les unités sémantiques, représentent *a priori* toutes les deux des sens. La section 3.2 étudie dans quelle mesure les unités sémantiques peuvent être organisées en un hiérarchie de sens, comme le sont les étiquettes sémantiques. Nous étendrons cette étude à la hiérarchisation des autres unités linguistiques que nous étudions.

Nous justifierons finalement l'introduction d'un nouveau niveau de représentation linguistique pour représenter les sens : *le niveau sémantique profond*, et nous préciserons les unités sémantiques profondes dans la section 3.3. En particulier, le niveau sémantique profond est conçu pour unifier les notions d'étiquette sémantique et de situation linguistique. Nous associerons à chaque type d'unité sémantique profonde ($\equiv$ étiquette sémantique) une structure actancielle, que nous définirons.

3.1 Précisions terminologiques et notationnelles

Pour une lexie $\ulcorner$L$\urcorner$, Mel'čuk (2004a) distingue $\ulcorner$L$\urcorner$ dans la langue $\mathscr{L}$ (i.e., dans le dictionnaire), ou dans l'usage (i.e., dans les énoncés). Les formalismes de représentation des connaissances font cette distinction explicitement en utilisant les notions de *type* et d'*instance*. Nous proposons donc d'étendre la conceptualisation de la TST afin de supprimer cette ambiguïté. Ainsi pour une lexie $\ulcorner$L$\urcorner$, nous représenterons $\ulcorner$L$\urcorner$ comme un type dans le dictionnaire, et comme une instance de ce type dans les énoncés. Afin de compléter notre apport, nous proposons également de supprimer l'ambiguïté notationnelle en nous inspirant du langage de modélisation UML.

Nous généralisons cette distinction aux différentes autres unités linguistiques que nous étudions : les unités grammaticales, et les unités sémantiques.

Par ailleurs, et d'une manière générale, on associera des URI aux unités linguistiques que l'on manipule, afin de faciliter la représentation ultérieure à l'aide des formalismes du Web Sémantique, comme proposé dans la section 2.2.3.

3.1.1 Types, instances, identifiants, nœuds, et marqueurs

Dorénavant, le terme unité linguistique et ses dérivés réfèreront aux unités linguistiques vues dans les textes, c'est à dire aux instances d'unité linguistique. Pour référer aux unités linguistiques vues dans la langue, nous utiliserons explicitement le terme *type d'unité linguistique*.

Définition 3 (Unité linguistique, et type d'unité linguistique)**.** Nous différencions les types et les instances d'unité linguistique :
- une unité linguistique (ex : lexicale, grammaticale, sémantique) vue dans la langue est appelée *type d'unité linguistique* ;
- une unité linguistique vue dans les textes est appelée indifféremment *instance d'unité linguistique*, ou simplement *unité linguistique*.

Les (instances d')unités linguistiques seront décrites dans des représentations linguistiques. Afin de faciliter la représentation à l'aide des formalismes du Web Sémantique, chaque instance d'unité linguistique pourra être identifiée par une ou plusieurs URI, choisies dans un ensemble d'*identifiants d'unité linguistique* $\boldsymbol{M}$.

Définition 4 (Identifiants d'unité linguistique)**.** Nous introduisons un ensemble d'identifiants d'unité linguistique $\boldsymbol{M}$. Chaque élément de $\boldsymbol{M}$ identifie une unité linguistique, mais plusieurs éléments de $\boldsymbol{M}$ peuvent identifier la même unité linguistique. Tout identifiant d'unité linguistique $m \in \boldsymbol{M}$ est associé à une URI notée $\text{uri}(m)$.

Par exemple, $i01$ et $i471$ sont des identifiants d'unité linguistique avec $\text{uri}(i01)$ =<http://ns.inria.org/ug/v1/mtt/fr/example#personne1IA-i01> et $\text{uri}(i471)$ =<http://ns.inria.org/ug/v1/mtt/fr/example#sem-aiderIA1a-i471>.

Par contre, chaque représentation linguistique met en scène des instances d'unité linguistique par l'intermédiaire d'un concept intermédiaire nommé *nœud d'unité linguistique*. Cette indirection nous permet d'anticiper la possibilité que deux nœuds puissent représenter la même instance d'unité linguistique dans une représentation linguistique par exemple.

Définition 5 (Nœud d'unité linguistique)**.** Les nœuds d'unité linguistique sont interconnectés dans les représentations linguistiques. Chaque nœud d'unité linguistique représente une unité linguistique, mais plusieurs nœuds d'unité linguistique peuvent représenter la même unité linguistique.

Dans les représentations linguistiques, chaque nœud d'unité linguistique possède une *étiquette*, qui est un couple formé d'un *type* et d'un *marqueur*.

Définition 6 (Étiquettes, types, et marqueurs de nœuds d'unité linguistique)**.** Les étiquettes de nœuds d'unité linguistique spécifient :
- Le type du nœud d'unité linguistique : il s'agit d'un ensemble potentiellement vide de types d'unité linguistique qui sont autant de types de l'unité linguistique représentée par le nœud ;
- Le marqueur du nœud d'unité linguistique : il s'agit d'un ensemble potentiellement vide d'identifiants d'unité linguistique.

Par abus de langage, nous pourrons faire référence à une unité linguistique en utilisant un marqueur m, à la condition que m soit associé à une URI qui identifie également l'unité linguistique en question.

3.1.2 Types et instances des unités linguistiques

Cette section introduit les types et instances des unités lexicales et grammaticales (§3.1.2.1), puis les types et instances d'unité sémantique (§3.1.2.2).

3.1.2.1 Types et instances des unités lexicales et grammaticales

Définition 7 (Type de lexie)**.** Une lexie considérée dans la langue est appelée *type de lexie*. La notation $\ulcorner \mathrm{L} \urcorner$ est réservée pour dénoter les types de lexie. Tout type de lexie $\ulcorner \mathrm{L} \urcorner$ est associé à une URI notée $\mathrm{uri}(\ulcorner \mathrm{L} \urcorner)$.

Par exemple, $\ulcorner \mathrm{PERSONNE}_{\mathbf{1.I.A}} \urcorner$ et $\ulcorner \mathrm{MANGER}_{\mathbf{1.A}} \urcorner$ sont deux types de lexie française, ayant par exemple pour URI <http://ns.inria.org/ug/v1/mtt/fr/ecd#personne1IA> et <http://ns.inria.org/ug/v1/mtt/fr/ecd#manger1a>.

Définition 8 (Type d'unité grammaticale)**.** Une unité grammaticale considérée dans la langue est appelée *type d'unité grammaticale*. La notation g est réservée pour dénoter les types d'unité grammaticale. Tout type d'unité grammaticale g est associé à une URI notée $\mathrm{uri}(g)$.

Par exemple, *singulier*, *pluriel*, $defini$ et $indefini$ sont des types d'unité grammaticale qui s'appliquent à la plupart des noms, ayant par exemple pour URI <http://ns.inria.org/ug/v1/mtt/fr#singulier>, <http://ns.inria.org/ug/v1/mtt/fr#pluriel>, <http://ns.inria.org/ug/v1/mtt/fr#defini> et <http://ns.inria.org/ug/v1/mtt/fr#indefini>.

Définition 9 (Unité lexicale)**.** Une lexie considérée dans les textes est appelée *instance de lexie*, ou simplement *lexie*.
Une lexie est *typée* par un unique type de lexie, plus éventuellement un ensemble de types d'unité grammaticale.
Une instance de lexie peut être *identifiée par* une ou plusieurs URI.
La notation $\{\ulcorner \mathrm{L} \urcorner, g_1, g_2, \ldots\} : \{m_1, m_2, \ldots\}$ dénote l'*étiquette* d'un nœud de lexie :

- $\ulcorner \mathrm{L} \urcorner$ est le type lexical du nœud de lexie, il s'agit également du type de l'instance de lexie représentée ;
- g_1 et g_2 sont des types grammaticaux du nœud de lexie, ce sont donc également des types grammaticaux de l'instance de lexie représentée ;
- $\{m_1, m_2, \ldots\}$ est le *marqueur* du nœud de lexie. S'il est vide, alors le nœud est dit *anonyme*.

Par exemple, $\{\ulcorner \mathrm{PERSONNE}_{\mathbf{1.I.A}} \urcorner, singulier, defini\} : i428$, avec $\mathrm{uri}(i428) =$ <http://ns.inria.org/ug/v1/mtt/fr/example#personne1IA-i428> est une étiquette de nœud de lexie, qui a le type lexical $\ulcorner \mathrm{PERSONNE}_{\mathbf{1.I.A}} \urcorner$, les deux types grammaticaux *singulier* et $defini$, et un marqueur singleton $i428$.

Cette lexie est également instance des types d'unité grammaticale *singulier* et $defini$.

Définition 10 (Unité grammaticale)**.** Une *instance d'unité grammaticale*, ou simplement *unité grammaticale*, est une lexie qui possède au moins un type grammatical.

Visualisation Nous proposons une visualisation graphique inspirée d'UML pour les types et instances d'unité lexicale et grammaticale, comme l'illustre la figure 3.1. Les rectangles du haut représentent des types de lexie et d'unités grammaticales, et possèdent une zone rectangulaire inférieure pour y visualiser leurs structures actancielles (syntaxique profonde et de surface) éventuelles. Les rectangles du bas représentent deux nœuds de lexie.

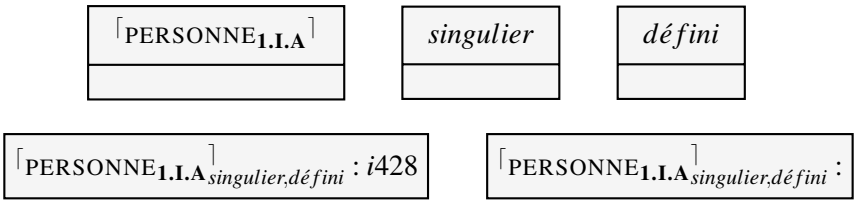


FIGURE 3.1 – Visualisation des types et instances d'unité lexicale et grammaticale.

3.1.2.2 Types et instances des unités sémantiques

Parallèlement aux lexies, nous distinguons les unités sémantiques dans la langue ou dans les RSém. Une unité sémantique dans la langue est appelée *type d'unité sémantique* et est décrite dans le lexique. Ses instances se lient dans les RSém.

Définition 11 (Type d'unité sémantique)**.** À tout type de lexie pleine ⌈L⌉ est associé un *type d'unité sémantique* noté ⟨L⟩. La notation ⟨L⟩ est utilisée pour dénoter un type d'unité sémantique. Tout type d'unité sémantique ⟨L⟩ est associé à une URI notée uri(⟨L⟩).

Par exemple, ⟨personne$_{\mathbf{1.I.A}}$⟩ est un type d'unité sémantique avec l'URI <http://ns.inria.org/ug/v1/mtt/fr/ecd#sem-personne1IA>, qui a pour type de lexie associé ⌈PERSONNE$_{\mathbf{1.I.A}}$⌉.

Selon Mel'čuk (1997), ⟨one⟩, ⟨several⟩, ⟨definite⟩ ⟨indefinite⟩ sont des types d'unité sémantique qui peuvent être lexicalisés par les types d'unité grammaticale suivants pour les noms anglais : *singular*, *plural*, *definite* et *indefinite*.

Définition 12 (Unité sémantique)**.** Une unité sémantique considérée dans les textes est appelée *instance d'unité sémantique*, ou simplement *unité sémantique*.
Une unité sémantique est *typée* par un ou plusieurs types d'unité sémantique.
Une instance d'unité sémantique peut être *identifiée par* une ou plusieurs URI.
La notation $\{⟨L_1⟩,⟨L_2⟩,\ldots\} : \{m_1,m_2,\ldots\}$ dénote l'*étiquette* d'une nœud d'unité sémantique :

- $\{⟨L_1⟩,⟨L_2⟩,\ldots\}$ est le *type* du nœud d'unité sémantique ;
- $⟨L_1⟩,⟨L_2⟩,\ldots$ sont les types de l'unité sémantique représentée ;
- $\{m_1,m_2,\ldots\}$ est le *marqueur* du nœud d'unité sémantique. S'il est vide, alors le nœud est dit *anonyme*.

Par exemple, ⟨personne$_{\mathbf{1.I.A}}$⟩ : $i21$ avec :
uri($i21$) =<http://ns.inria.org/ug/v1/mtt/fr/ecd#sem-personne1IA-i21> est une étiquette de nœud d'unité sémantique. L'unité sémantique représentée a au moins le type d'unité sémantique ⟨personne$_{\mathbf{1.I.A}}$⟩, et est identifiée par <http://ns.inria.org/ug/v1/mtt/fr/ecd#sem-personne1IA-i21>.

Visualisation Nous proposons une visualisation graphique inspirée d'UML pour les types et instances d'unité sémantique, comme l'illustre la figure 3.2. Le nœud supérieur représente un type d'unité sémantique, et possède une zone rectangulaire inférieure pour y visualiser sa structure actancielle éventuelle. Les nœuds inférieurs représentent des instances d'unité sémantique.

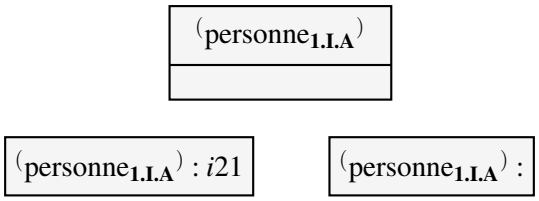


FIGURE 3.2 – Visualisation des types et instances d'unité sémantique.

3.1.3 Association de la structure actancielle sémantique aux types d'unité sémantique

Dans la théorie des actants (Mel'čuk, 2004a,b), chaque type de lexie possède une structure actancielle composite munie de trois types de PosA : sémantiques, syntaxiques profondes, et syntaxiques de surface. Puisque nous nous concentrons sur les définitions lexicographiques dans cette thèse, nous nous intéressons exclusivement aux PosASém. D'un autre côté, la méthodologie que l'on applique peut être réutilisée pour étendre la conceptualisation des structures actancielles syntaxiques des unités lexicales.

Puisqu'on décrit séparément les types de lexie et les types d'unité sémantique, nous formulons le choix de conceptualisation suivant : chaque type d'unité sémantique possède une structure actancielle qui correspond à la structure actancielle sémantique de son type de lexie associé.

Définition 13 (Positions actancielles d'un type d'unité sémantique)**.** À tout type de lexie pleine ⌈L⌉ est associé un *type d'unité sémantique* noté ⟨L⟩. Les PosA obligatoires et optionnelles de ⟨L⟩ correspondent aux PosASém obligatoires et optionnelles de ⌈L⌉, comme défini dans (Mel'čuk, 2004a, p.39), et sont numérotées.

Par exemple, le type de lexie anglaise ⌈TO EAT⌉ est associé à un type d'unité sémantique ⟨to eat⟩ qui possède les PosA suivantes :
- une PosA obligatoire 1 qui représente l'animal qui mange ;
- une PosA obligatoire 2 qui représente ce qui est mangé ;
- une PosA optionnelle 3 qui représente le contenant dans lequel l'animal mange.

Suivant les critères linguistiques donnés dans (Mel'čuk, 2004a), la troisième PosASém du type de lexie anglaise ⌈TO EAT⌉ n'existe pas pour le type de lexie française ⌈MANGER⌉. Il peut être exprimé dans les représentations sémantiques seulement sous la forme d'un circonstant.

Ainsi, la précision descriptive dans la conceptualisation des unités sémantiques dans la TST encourage la prudence lors du lien de différents lexiques, comme par exemple dans le projet Papillon (Mangeot *et al.*, 2003). En effet, les types d'unité sémantique associés à ces deux types de lexie ne peuvent pas être directement équivalents. Pour aller plus loin, on peut même supposer que le sens de ⟨to eat⟩ est plus précis que celui de ⟨manger⟩.

Visualisation La structure actancielle des types d'unité sémantique est visualisée dans la partie inférieure de la visualisation, comme l'illustre la figure 3.3. La nature des PosA est représentée à l'aide des symboles '!' pour obligatoire, et '?' pour optionnel.

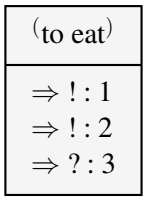


FIGURE 3.3 – Visualisation de la structure actancielle des types d'unité sémantique.

3.2 Hiérarchisation des types d'unité linguistique

Maintenant que nous avons précisé la terminologie et les notations pour notre étude, nous nous intéressons à la possibilité d'organiser les types d'unité linguistique en hiérarchies.

La capacité de raisonnement première offerte par les formalismes de représentation des connaissances est l'inférence de type. Si un type A est déclaré sous-type d'un type B, et que si a est une instance du type A, alors on peut inférer que a est également instance du type B. Les taxonomies sont des ontologies très légères qui utilisent presque exclusivement cet axiome. La question à laquelle cette section répond peut alors être formulée de la manière suivante :

Dans quelle mesure l'inférence de type est-elle intéressante pour la représentation des types d'unité linguistique ?

Nous étudierons la hiérarchie des types de lexie et des types d'unité grammaticale (§3.2.1), puis la hiérarchie des types d'unité sémantique (§3.2.2).

3.2.1 Hiérarchie des types de lexie et des types d'unité grammaticale

Supposons que le type de lexie ⌈PERSONNE$_{\mathbf{1.I.A}}$⌉ soit déclaré sous-type d'un type de lexie ⌈ANIMAL$_{\mathbf{1.A}}$⌉. Alors toute instance de ⌈PERSONNE$_{\mathbf{1.I.A}}$⌉ dans une représentation d'énoncé serait également une instance de ⌈ANIMAL$_{\mathbf{1.A}}$⌉. Or, un mot dans une phrase ne peut pas être à la fois *personne* et *animal*. En d'autres termes, une lexie devrait n'avoir qu'un seul type lexical dans une représentation linguistique. L'inférence de type n'est donc pas pertinente pour les types de lexie.

Par contre, il devrait être possible d'inférer que l'unité lexicale représentée par ⌈PERSONNE$_{\mathbf{1.I.A}}$⌉ : dans un texte est une instance du type grammatical NOM. Cette contrainte peut être représentée si l'on déclare le type ⌈PERSONNE$_{\mathbf{1.I.A}}$⌉ comme sous-type du type d'unité grammaticale NOM.

Pour aller plus loin, considérons le problème de la mise au pluriel des instances de lexie de type ⌈FOOTBALL⌉, ⌈LUNETTES⌉, et ⌈CHEVAL⌉. ⌈FOOTBALL⌉ est ce qu'on appelle un *singulare tantum* et ses instances ne peuvent pas être mises au pluriel. Au contraire, ⌈LUNETTES⌉ est un *plurale tantum* et ses instances ne peuvent pas être mises au singulier. Par contre, les instances de ⌈CHEVAL⌉ peuvent être indifféremment mises au pluriel ou au singulier.

Supposons que l'on hiérarchise les types d'unité grammaticale, et introduisons un type *pluralisable* comme supertype commun à *singulier* et *pluriel*. On peut dès lors déclarer les types de lexie ⌈FOOTBALL⌉, ⌈LUNETTES⌉, et ⌈CHEVAL⌉, comme respectivement sous-types des types d'unité grammaticale *singulier*, *pluriel*, et *pluralisable*. Plus généralement, toute une hiérarchie peut être construite sous un type d'unité grammaticale *nombre*, et chaque type de lexie qui possède le trait grammatical *nombre* pourra être placé sous l'un des types d'unité grammaticale de cette hiérarchie. On appellera ainsi *singulare tantum* les types de lexie qui sont déclarés sous-types de *singulier*.

De cette manière, si un nœud de lexie étiqueté ⌜FOOTBALL⌝ : *i*07 est présent dans une représentation linguistique, alors l'inférence de type permet automatiquement de déterminer que la lexie *i*07 est également instance de *singulier* [1]. Cependant, l'inférence de type ne suffit pas pour empêcher *i*07 d'être également déclarée de type *pluriel*, par exemple par le biais d'un nœud étiqueté {⌜FOOTBALL⌝, *pluriel*} : *i*07. Le formalisme que nous choisirons pour représenter les connaissances du DEC devra donc comporter un mécanisme de validation de types. C'est par exemple le cas pour les logiques de description avec les axiomes `disjointClasses` et `disjointProperties`, et pour les Graphes Conceptuels avec les types conjonctifs bannis ou les règles de contraintes.

Visualisation Nous représentons la relation de spécialisation comme en UML. Par exemple, la figure 3.4 illustre la hiérarchie des types utilisés dans l'exemple ci-dessus.

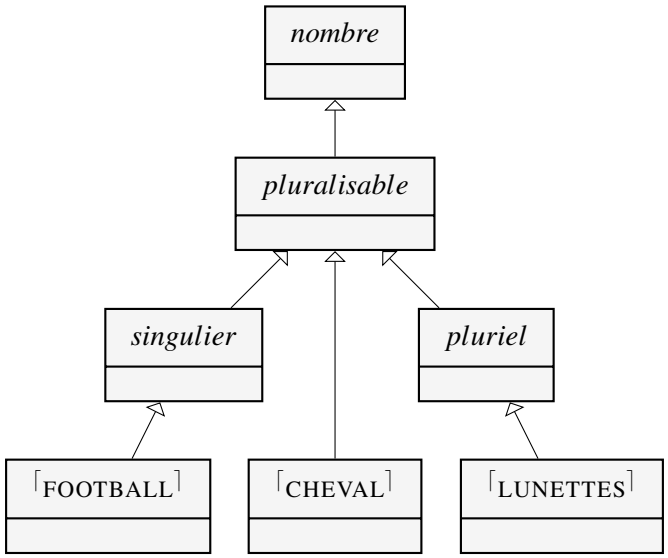


FIGURE 3.4 – Visualisation de la hiérarchie des types d'unité lexicale et grammaticale.

1. Nous utilisons ici l'abus de langage déclaré dans la section 3.1.1. La lexie *i*07 est en réalité la lexie représentée par le nœud de lexie étiqueté ⌜FOOTBALL⌝ : *i*07.

3.2.2 Hiérarchie des types d'unité sémantique

Chaque type d'unité sémantique est associé à une structure actancielle, et est censé représenter le sens du type de lexie associé. Si l'on organise les types d'unité sémantique dans une hiérarchie, les deux propriétés suivantes sont souhaitables :

1. la hiérarchie des types d'unité sémantique doit représenter une hiérarchisation des sens ;
2. la structure actancielle des types d'unité sémantique doit être héritée et spécialisée au sein de la hiérarchie.

La définition suivante précise ce que nous entendons par héritage et spécialisation de la structure actancielle des types d'unité sémantique.

Définition 14 (Contrainte d'héritage et de spécialisation de la structure actancielle)**.** Soit $^{(}L_1^{)}$ un sous-type de $^{(}L_2^{)}$:

1. $^{(}L_1^{)}$ hérite et éventuellement spécialise chacune des PosA de $^{(}L_2^{)}$;
2. $^{(}L_1^{)}$ peut introduire de nouvelles PosA ;

Cette contrainte peut s'appliquer à la plupart des formalismes de représentation des connaissances, à savoir à ceux qui ne permettent pas la gestion d'exceptions.

Les deux propriétés de la conceptualisation des types d'unité sémantique énumérées ci-dessus jouent un rôle différent dans la perspective d'une formalisation ultérieure. La première permet d'envisager l'inférence de type comme une inférence de la sémantique lexicale dans les RSém, quand la deuxième permet d'envisager l'inférence de type comme une inférence des propriétés de combinatoire sémantico-syntaxique.

Nous allons montrer que ces deux propriétés sont incompatibles à l'aide d'un contre-exemple.

Le type d'unité sémantique française $^{(}$outil$^{)}$ a une PosA 1 qui correspond à la personne X qui utilise l'outil, et une PosA 2 scindée qui correspond à l'activité Y_1 ou à la profession Y_2 pour laquelle l'outil a été conçu[2]. D'autre part, $^{(}$ciseaux$^{)}$ possède un sens plus précis que $^{(}$outil$^{)}$, et sa position actancielle 2 correspond à l'objet Y que l'outil est censé couper. Déclarer $^{(}$ciseaux$^{)}$ sous-type de $^{(}$outil$^{)}$ dans la hiérarchie des types d'unité sémantique reviendrait alors à imposer qu'un objet est un type de profession ou d'activité.

Considérons la solution suivante pour rétablir la satisfaction des deux propriétés dans la hiérarchie des types d'unité sémantique. Il serait possible d'imposer une scission supplémentaire pour la PosA 2 de $^{(}$outil$^{)}$: Y_3 correspondrait à l'objet dont l'état est censé être altéré par l'outil. Nous identifions deux aspects problématiques dans cette solution.

- **Problème méthodologique.** L'application stricte des critères linguistiques de choix de la structure actancielle ne justifie pas cette scission de la PosA 2 de $^{(}$outil$^{)}$. Ainsi, la recherche de formalisation de notre approche d'ingénierie des connaissances présenterait un effet de bord indésirable : le non-respect des connaissances du domaine de spécialité. En d'autres termes, alors que nous nous plaçons dans une démarche de capture des connaissances des linguistes, la solution proposée impose de devenir prescriptifs en modifiant les méthodes des linguistes.
- **Problème de formalisation.** Nous souhaitons proposer une conceptualisation de la théorie Sens-Texte la moins ambiguë possible. Or, si l'on extrapole cette solution :
  - d'autres spécialisations de $^{(}$outil$^{)}$ peuvent lui imposer la surcharge de la PosA 2. Par exemple, $^{(}$arrosoir$^{)}$ lui impose Y_4 : le liquide transporté. Ainsi, la PosA Y de $^{(}$outil$^{)}$ serait

2. cf., (Mel'čuk, 2004a, p.43) pour une définition de la notion de position actancielle sémantique scindée.

surchargée de différentes PosA optionnelles, et nous devrions faire davantage d'inférences pour identifier la relation sémantique qui existe réellement entre une occurrence de ⟨outil⟩ et l'une de ses PosA dans les textes.
– la PosA 2 ainsi complexifiée serait héritée par chacune des spécialisations de ⟨outil⟩. Or, la composante Y_4 (liquide transporté) n'a aucune légitimité dans la définition de ⟨ciseaux⟩. Nous devrions donc spécifier que la plupart sont interdites, provoquant ainsi une surcharge d'information dans notre modèle.

Ainsi, l'organisation hiérarchique des types d'unité sémantique utilisés pour dénoter des concepts ontologiques ne correspond pas à l'organisation hiérarchique de ces concepts ontologiques. Bien que la RSém soit la plus *proche* des sens, la hiérarchie des types d'unité sémantique ne correspond pas à une hiérarchie des sens. Nous dissocions donc les deux notions de types d'unité sémantique et de sens, mais nous proposerons une articulation simple entre ces niveaux. Différencier le sens des unités lexicales et les concepts ontologiques est également l'approche choisie par le groupe communautaire Ontology-Lexica.

Nous verrons dans la section suivante qu'en repoussant la description des sens à un niveau de représentation plus profond, le *niveau sémantique profond*, nous nous affranchissons des limites identifiées. La figure 3.11 représente les types d'unité sémantique profonde /outil\ et /ciseaux\, /ciseaux\ héritant à la fois du sens de /outil\ et de sa structure actancielle.

Visualisation La hiérarchie des types d'unité sémantique ne représente donc que la spécialisation des structures actancielles, comme l'illustre la figure 3.5. Sur cette figure, et à des fins d'illustration, nous introduisons un ensemble de types d'unité sémantique auxiliaires au nom composé de symboles ' ?' et ' !'. Le type d'unité sémantique ⟨ ! ?⟩ représente le plus générique des types d'unité sémantique dont la PosA 1 est obligatoire, et dont la PosA 2 est optionnelle. Le type d'unité sémantique ⟨ ! ! ?⟩ représente le plus générique des types d'unité sémantique dont les PosA 1 et 2 sont obligatoires, et dont la PosA 3 est optionnelle.

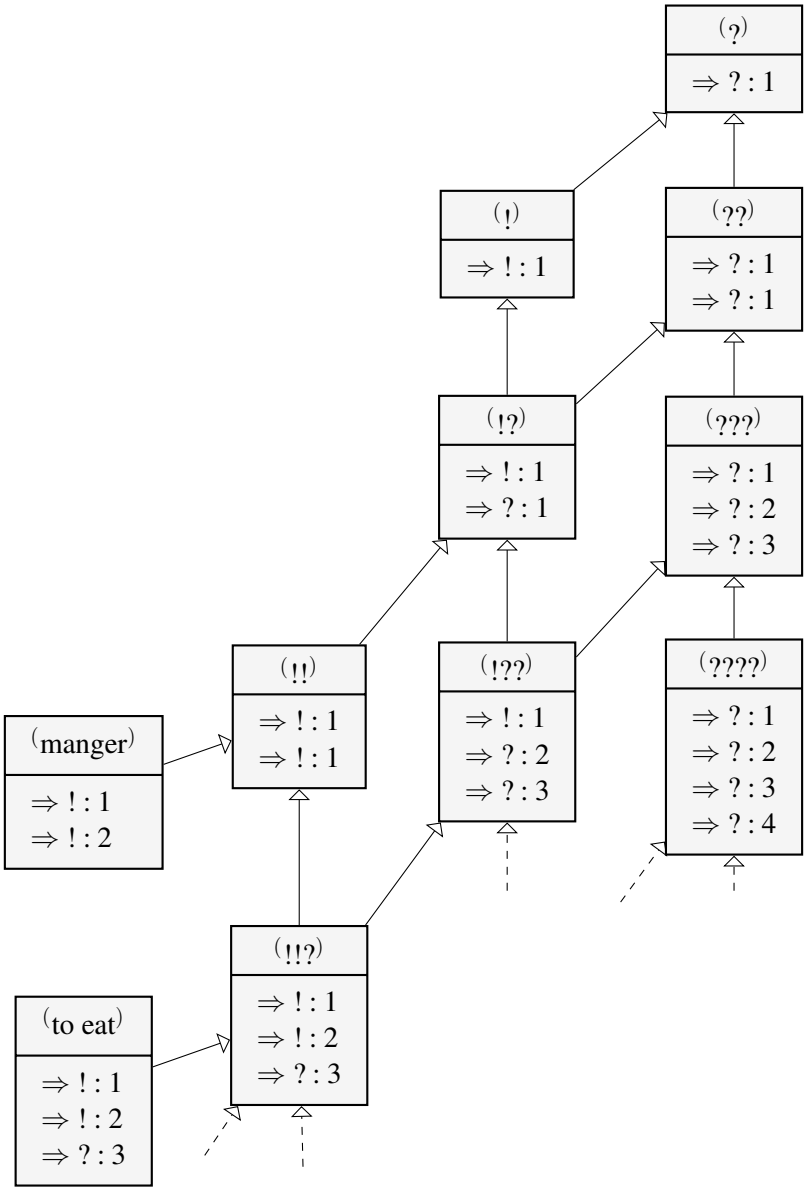


FIGURE 3.5 – Visualisation de la hiérarchie des types d'unité sémantique, dirigée par leur structure actancielle.

3.3 Des unités sémantiques profondes pour porter les sens

Ainsi nous avons justifié qu'une hiérarchie de types d'unité sémantique ne peut pas représenter à la fois une hiérarchie des sens, et une hiérarchie des structures actancielles.

Nous proposons donc d'étendre la conceptualisation de la TST en définissant un nouveau niveau de représentation linguistique, plus profond que le niveau sémantique de la TST, et qui serait dédié à la représentation des sens. Nous nommons ce niveau le *niveau sémantique profond*, et renommons à cette occasion le niveau sémantique de la TST en *niveau sémantique de surface*.

Cette section précise donc la conceptualisation des unités sémantiques profondes en types et instances, ainsi que la structure actancielle associée aux types. Nous développons cette étude en suivant les trois objectifs principaux suivants :

1. la hiérarchie des types d'unité sémantique profonde représente une hiérarchisation des sens ;
2. la structure actancielle des types d'unité sémantique profonde est héritée et éventuellement spécialisée au sein de la hiérarchie ;
3. il existe une correspondance simple entre la structure actancielle des types d'unité sémantique profonde, et celle des types d'unité sémantique de surface.

La hiérarchie des types d'unité sémantique profonde peut alors être identifiée à la hiérarchie des étiquettes sémantiques, auxquelles on ajouterait une structure actancielle. Nous introduirons les types et instances des unités sémantiques profondes (§3.3.1), nous choisirons un ensemble de symboles pour les relations sémantiques profondes (§3.3.2), puis nous conceptualiserons la structure actancielle de ces nouveaux types d'unité linguistique (§3.3.3).

3.3.1 Types et instances des unités sémantiques profondes (USemP)

A l'instar des unités sémantiques *de surface* précédemment caractérisées, nous distinguons les unités sémantiques profondes dans la langue et dans les textes. Les unités sémantiques profondes dans la langue sont nommées types d'unité sémantique profonde, elles sont décrites dans le dictionnaire, et leurs instances sont liées dans les *représentations sémantiques profondes* (RSémP).

Définition 15 (Type d'unité sémantique profonde)**.** À tout type de lexie pleine $\ulcorner \mathrm{L} \urcorner$ est associée un *Type d'Unité Sémantique Profonde (TUSemP)* noté ${}^{/}\mathrm{L}^{\backslash}$. La notation ${}^{/}\mathrm{L}^{\backslash}$ est réservée pour dénoter un TUSemP. Tout TUSemP ${}^{/}\mathrm{L}^{\backslash}$ est associé à une URI noté uri$({}^{/}\mathrm{L}^{\backslash})$.

Par exemple, ${}^{/}\mathrm{personne}_{\mathbf{1.I.A}}{}^{\backslash}$ est un type d'unité sémantique profonde avec l'URI `<http://ns.inria.org/ug/v1/mtt/fr/ecd#dsem-personne1IA>`, qui a pour type de lexie associé $\ulcorner \mathrm{PERSONNE}_{\mathbf{1.I.A}} \urcorner$.

Définition 16 (Unité sémantique profonde)**.** Une unité sémantique profonde considérée dans les textes est appelée *Instance d'Unité Sémantique Profonde (IUSemP)*, ou simplement *Unité Sémantique Profonde (USemP)*.
Une USemP est *typée* par un ou plusieurs TUSemP.
Une USemP peut être *identifiée par* une ou plusieurs URI.
La notation $\{{}^{/}\mathrm{L}_1{}^{\backslash}, {}^{/}\mathrm{L}_2{}^{\backslash}, \ldots\} : \{m_1, m_2, \ldots\}$ dénote l'*étiquette* d'un nœud d'unité sémantique profonde :

- $\{{}^{/}\mathrm{L}_1{}^{\backslash}, {}^{/}\mathrm{L}_2{}^{\backslash}, \ldots\}$ est le *type* du nœud d'unité sémantique profonde ;
- ${}^{/}\mathrm{L}_1{}^{\backslash}, {}^{/}\mathrm{L}_2{}^{\backslash}, \ldots$ sont des types de l'USemP représentée ;
- $\{m_1, m_2, \ldots\}$ est le *marqueur* du nœud d'unité sémantique profonde. S'il est vide, alors le nœud est dit *anonyme*.

Par exemple, /personne$_{\textbf{1.I.A}}$\ : *i*748 avec
uri(*i*748) =<http://ns.inria.org/ug/v1/mtt/fr/ecd#dsem-personne1IA-i748> est une étiquette de nœud d'USemP. L'USemP représentée a au moins le TUSemP /personne$_{\textbf{1.I.A}}$\, et est identifiée par <http://ns.inria.org/ug/v1/mtt/fr/ecd#sem-personne1IA-i748>.

Visualisation Nous proposons une visualisation graphique inspirée d'UML pour les types et instances d'USemP, comme l'illustre la figure 3.6. Le rectangle du haut représente un type d'unité sémantique profonde, et possède une zone rectangulaire inférieure pour y visualiser sa structure actancielle éventuelle. Les rectangles du bas représentent des instances d'unité sémantique profonde.

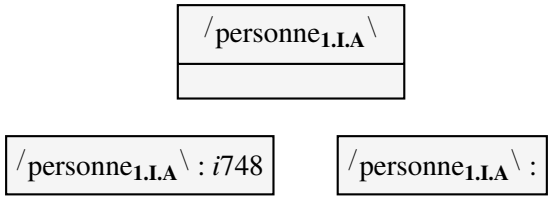


FIGURE 3.6 – Visualisation des types et instances d'unité sémantique profonde.

3.3.2 Choix des relations sémantiques profondes

Nous devons proposer un ensemble de relations qui étiquetteront les PosA des TUSemP, et les relations actancielles entre USemP dans les représentations sémantiques profondes. Comme ces relations vont être porteuses de sens, une première proposition est d'utiliser un ensemble de rôles sémantiques universels (ex : *agent*, *patient*, *objet*). Cependant, nous notons trois problèmes avec cette approche :

- il n'existe pas d'accord collectif sur un tel ensemble universel ;
- même si un accord était atteint dans une communauté donnée (ex : dans la communauté Universal Networking Language (UNL)), l'expérience montre que l'étiquetage reste parfois problématique et que des désaccords émergent ;
- une nouvelle relation sémantique profonde devrait être choisie à chaque fois qu'une PosASém qui porte un nouveau sens serait introduite.

Ainsi, nous choisissons de nous inspirer de l'approche de Fillmore pour FrameNet, et d'utiliser un ensemble de relations sémantiques lexicalisées, potentiellement infini. Par exemple, puisque le type de lexie ⌈TO EAT⌉ a trois PosASém, le TUSemP associé /to eat\ pourrait avoir au moins les trois PosA étiquetées de la manière suivante :

- *eater* : l'animal qui mange ;
- *eaten* : ce qui est mangé ;
- *container* : le contenant dans lequel l'animal mange.

Visualisation La structure actancielle des TUSemP est visualisée dans la partie inférieure du rectangle, comme l'illustre la figure 3.7. La nature des PosA est représentée à l'aide des symboles '!' pour obligatoire, et '?' pour optionnel.

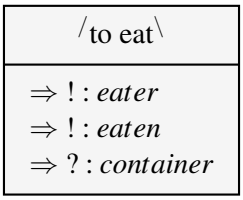


FIGURE 3.7 – Visualisation de la structure actancielle d'un type d'unité sémantique profonde.

3.3.3 Conceptualisation de la structure actancielle des types d'USemP

Cette section met au point un ensemble de principes et de critères pour définir la structure actancielle des TUSemP, de sorte que la hiérarchie des TUSemP satisfasse les trois propriétés précisées en préambule de cette section (cf., p.63).

Nous repartons de la notion de situation linguistique (§3.3.3.1), précisons ensuite les différentes composantes de la structure actancielle des TUSemP (§3.3.3.2), et concluons sur la structure générale de la hiérarchie des TUSemP (§3.3.3.3).

3.3.3.1 Principe d'héritage et de spécialisation éventuelle

Précisons dans un premier temps ce que nous entendons par *héritage et spécialisation de la structure actancielle des TUSemP*.

Définition 17 (Principe d'héritage et de spécialisation éventuelle de la structure actancielle des TUSemP)**.** Soit $/L_1\backslash$ un sous-type de $/L_2\backslash$:

1. $/L_1\backslash$ hérite et éventuellement spécialise chacune des PosA de $/L_2\backslash$;
2. $/L_1\backslash$ peut introduire de nouvelles PosA.

Ce principe n'est que partiellement défini pour le moment, et l'héritage et l'éventuelle spécialisation d'une PosA sera précisée dans la section 3.3.3.3. En attendant, notons que ce principe est très similaire au principe d'héritage des participants obligatoires de la théorie des actants sémantiques (Mel'čuk, 2004a, p.13) :

> Let there be the lexicographic definition of $\ulcorner L \urcorner$. [The Linguistic Situation denoted by $\ulcorner L \urcorner$] SIT($\ulcorner L \urcorner$) inherits all obligatory participants of all SIT($\ulcorner L_1 \urcorner$) that correspond to the predicative meanings [$/L_i\backslash$] which compose [$/L\backslash$].

Note. Puisqu'on a montré que les sens sont portés par les TUSemP, et pas par les TUSemS, nous avons remplacé la notation (L) par $/L\backslash$ dans la citation ci-dessus.

L'hypothèse principale sous-jacente de notre étude dans cette section est que nous identifions le niveau sémantique profond au niveau des situations linguistiques (cf., §1.3.1.1). Ainsi, les PosA d'un TUSemP $/L\backslash$ représentent des participants de la situation linguistique dénotée par $\ulcorner L \urcorner$ SIT($\ulcorner L \urcorner$).

Avant de construire la structure actancielle des TUSemP, rappelons tout d'abord deux connaissances importantes de la théorie des actants sémantiques :

- toute PosASém obligatoire de $\lceil L \rceil$ correspond à un ensemble non vide de participants de SIT($\lceil L \rceil$) : au moins un participant obligatoire, et éventuellement d'autres participants (obligatoires ou optionnels) si la PosASém est scindée ;
- toute PosASém optionnelle de $\lceil L \rceil$ correspond à au moins un participant optionnel de SIT($\lceil L \rceil$), et àplus d'un si la PosASém est scindée.

3.3.3.2 Nature et signature des positions actancielles (PosA)

Positions actancielles obligatoires et optionnelles. Puisque c'est au niveau sémantique profond que les sens sont représentés, et afin de limiter la complexité de la conceptualisation des PosASém, nous imposons que chaque PosA obligatoire ou optionnelle d'un TUSemP $/L\backslash$ représente exactement un participant de SIT($\lceil L \rceil$).

Critère de représentation des participants par les PosA, partie 1/2.

- toute PosA obligatoire de $/L\backslash$ représente exactement un participant obligatoire de SIT($\lceil L \rceil$) ;
- toute PosA optionnel de $/L\backslash$ représente exactement un participant optionnel de SIT($\lceil L \rceil$).

Par ailleurs, nous savons que toute PosA obligatoire (resp. optionnelle) de (L) correspond à une PosASém obligatoire (resp. optionelle) de $\lceil L \rceil$. Nous proposons le critère suivant qui assure une correspondance simple entre la structure actancielle d'un TUSemP $/L\backslash$ et la structure actancielle du TUSemS (L) associé.

Critère de correspondance simple entre les structures actancielles des TUSemP et des TUSemS.

- chaque PosA obligatoire de (L) correspond à un ensemble non vide de PosA de $/L\backslash$: au moins une PosA obligatoire, et éventuellement d'autres PosA (obligatoires ou optionnelles) si la PosASém correspondante de $\lceil L \rceil$ est scindée ;
- chaque PosA optionnelle de (L) correspond à au moins une PosA optionnelle de $/L\backslash$, et plus d'une si la PosASém correspondante de $\lceil L \rceil$ est scindée.

La correspondance entre les PosA des TUSemP et des TUSemS peut alors être visualisée, comme sur la figure 3.8.

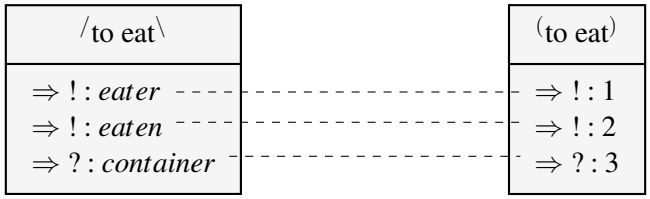


FIGURE 3.8 – Visualisation de la correspondance entre les structure actancielle des types d'unité sémantique de surface et profonde.

Positions actancielles interdites Nous souhaitons que chaque sous-type de $^{/}\mathrm{L}^{\backslash}$ en spécialise le sens, et hérite de chaque PosA. Nous pouvons donc justifier de l'importance d'introduire un nouveau type de position actancielle pour les TUSemP : les PosA interdites. Considérons par exemple les types de lexie anglaise $^{\lceil}$TO EAT$^{\rceil}$ (*manger*) et $^{\lceil}$TO GRAZE$^{\rceil}$ (*brouter*). Selon les critères linguistiques de la théorie des actants sémantiques, le premier possède une PosASém optionnelle qui correspond au contenant dans lequel l'animal mange. $^{/}$to eat$^{\backslash}$ possède donc une PosA optionnelle que l'on nommera par exemple *container*. D'autre part, le sens de $^{\lceil}$TO GRAZE$^{\rceil}$ est plus spécifique que celui de $^{\lceil}$TO EAT$^{\rceil}$, donc $^{/}$to graze$^{\backslash}$ est un sous-type de $^{/}$to eat$^{\backslash}$ et hérite de la PosA *container*. Cependant, il n'existe pas de participant dans SIT($^{\lceil}$TO GRAZE$^{\rceil}$) qui représente ce contenant. Ce paradoxe peut être résolu en introduisant une nouvelle nature pour les participants et les PosA : la nature *interdite*. $^{/}$to graze$^{\backslash}$ hérite ainsi de la PosA *container*, mais la spécialise en une PosA interdite qui représente un participant interdit de SIT($^{\lceil}$TO GRAZE$^{\rceil}$). Cet exemple se généralise et se formalise de la manière suivante :

Critère de représentation des participants par les PosA, partie 2/2.
Chaque PosA interdite de $^{/}\mathrm{L}^{\backslash}$ représente un *participant interdit* de SIT($^{\lceil}\mathrm{L}^{\rceil}$).

Ce critère implique qu'aucune correspondance ne peut exister entre les PosA interdites de $^{/}\mathrm{L}^{\backslash}$ et les PosA de $^{(}\mathrm{L}^{)}$. Cependant, dans un souci de minimalité, il est important d'imposer qu'aucune PosA ne soit introduite si ce n'est pas nécessaire.

Critère de limite du nombre de PosA des TUSemP, partie 1/2.
$^{/}\mathrm{L}^{\backslash}$ n'introduit une PosA interdite que si c'est nécessaire.

Nous pouvons maintenant proposer une définition partielle du principe d'héritage et de spécialisation éventuelle d'une PosA :

- En accord avec le principe d'héritage des participants obligatoires de la théorie des actants (Mel'čuk, 2004a, p.13), si un TUSemP $^{/}\mathrm{L}_1^{\backslash}$ est un sous-type de $^{/}\mathrm{L}_2^{\backslash}$, alors chaque PosA obligatoire de $^{/}\mathrm{L}_2^{\backslash}$ est héritée en tant que PosA obligatoire par $^{/}\mathrm{L}_1^{\backslash}$.
- De même, chaque PosA interdite de $^{/}\mathrm{L}_2^{\backslash}$ est héritée en tant que PosA interdite par $^{/}\mathrm{L}_1^{\backslash}$.
- Enfin, une PosA optionnelle de $^{/}\mathrm{L}_2^{\backslash}$ peut être héritée en tant qu'optionnelle par $^{/}\mathrm{L}_1^{\backslash}$, mais nous avons vu qu'elle peut également être spécialisée en PosA obligatoire ou interdite.

On identifie une PosA interdite par le symbole '0', comme l'illustre la figure 3.9 ci-dessous.

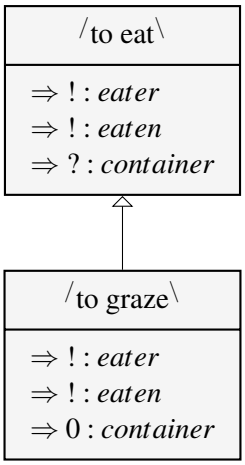


FIGURE 3.9 – Visualisation de la structure actancielle des types d'unité sémantique profonde.

Signatures. Parmi les composantes de la définition lexicographique, on trouve la composante de *contraintes sur les variables*. Cette composante spécifie le sens d'une lexie qui prend une PosASém d'une autre lexie. Puisque le niveau sémantique profond est celui qui porte les sens, nous proposons d'associer à chaque TUSemP une *signature*, qui spécifie le type que peut prendre un actant d'une USemP.

Définition 18 (Signature d'un TUSemP)**.** Un TUSemP $/\mathrm{L}\backslash$ est associé à une *signature*, qui encode la composante de contraintes sur les variables de la définition lexicographique de $\lceil \mathrm{L} \rceil$.

Par exemple, la signature de /to eat\ peut être :
– /living animal\ pour la PosA *eater* ;
– /thing\ pour la PosA *eaten*.

Nous pouvons maintenant étendre la définition du principe d'héritage et de spécialisation éventuelle d'une PosA : si un TUSemP $/\mathrm{L}_1\backslash$ est un sous-type de $/\mathrm{L}_2\backslash$, alors la signature de $/\mathrm{L}_2\backslash$ pour une PosA est héritée et peut être spécialisée pour $/\mathrm{L}_1\backslash$. Par exemple, la signature de /to graze\ pour *eaten* peut être spécialisée à /vegetal\.

Les signatures peuvent être visualisées de deux façons : dans la partie inférieure du nœud, ou comme une association illustrée par un trait double. Cette deuxième visualisation permet d'illustrer la spécialisation éventuelle d'une signature. La figure 3.10 illustre ces deux visualisations équivalentes.

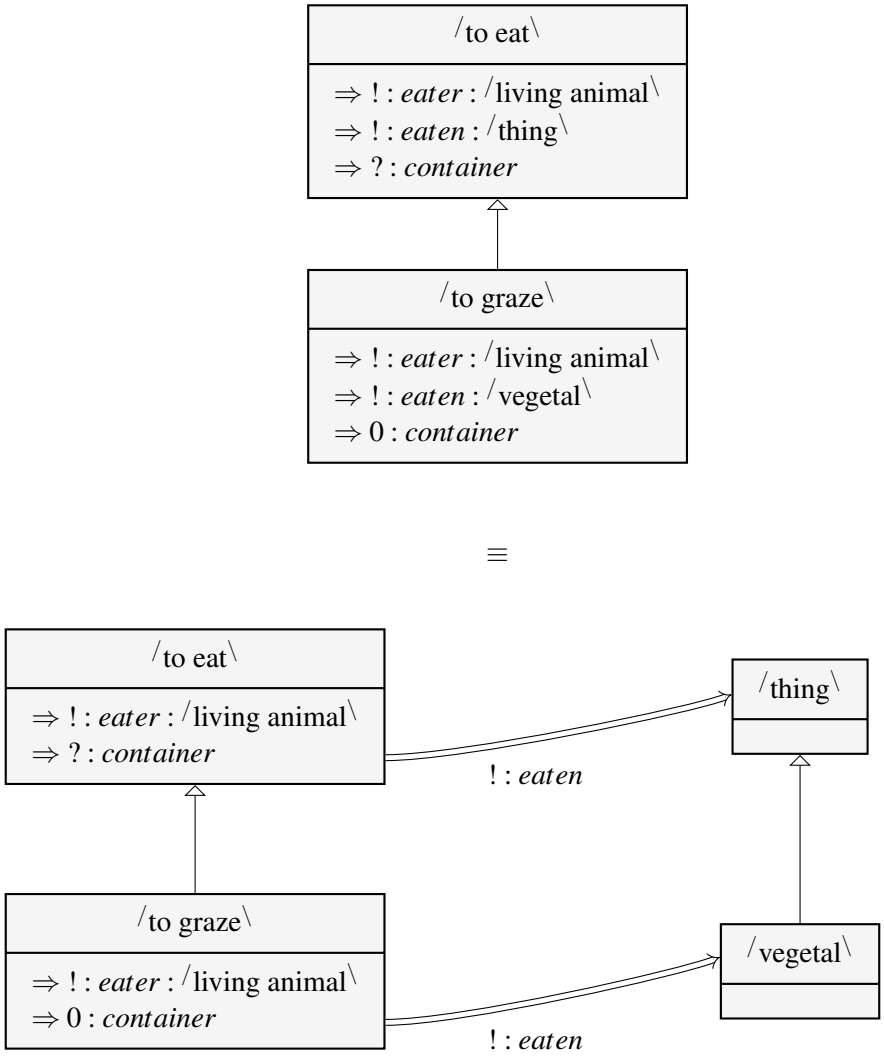


FIGURE 3.10 – Visualisations équivalentes de la signature des types d'unité sémantique profonde.

3.3.3.3 Hiérarchie des types d'unité sémantique profonde

Héritage et spécialisation d'une PosA. Maintenant que nous avons introduit les PosA interdites et les signatures, nous pouvons préciser le principe d'héritage et de spécialisation d'une PosA particulière d'un TUSemP :

Définition 19 (Principe d'héritage et de spécialisation éventuelle d'une PosA)**.** Soit $^{/}L_1{}^{\backslash}$ un sous-type de $^{/}L_2{}^{\backslash}$, et s une PosA de $^{/}L_2{}^{\backslash}$. $^{/}L_1{}^{\backslash}$ hérite de la PosA s et peut éventuellement la spécialiser de trois façons :

1. si s est optionnelle pour $^{/}L_2{}^{\backslash}$, elle peut être spécialisée à *obligatoire* pour $^{/}L_1{}^{\backslash}$;
2. si s est optionnelle pour $^{/}L_2{}^{\backslash}$, elle peut être spécialisée à *interdite* pour $^{/}L_1{}^{\backslash}$;
3. la signature de de $^{/}L_2{}^{\backslash}$ pour s peut être spécialisée pour $^{/}L_1{}^{\backslash}$.

Ainsi le critère d'héritage et de spécialisation éventuelle de la structure actancielle des TUSemP est similaire au principe d'héritage des participants obligatoires de la théorie des actants (Mel'čuk, 2004a, p.13), avec les différences suivantes :
i) il se concentre sur chaque PosA plutôt que sur l'ensemble des participants, et
ii) il est étendu aux participants optionnels et interdits, et aux signatures.

TUSemP absurde. Soit $\{^{/}L_i{}^{\backslash}\}$ l'ensemble des supertypes de $^{/}L^{\backslash}$. Le principe d'héritage et de spécialisation éventuelle impose que :

- si l'un des $^{/}L_i{}^{\backslash}$ possède une PosA s, alors $^{/}L^{\backslash}$ hérite de la PosA s ;
- si l'un des $^{/}L_i{}^{\backslash}$ possède une PosA obligatoire s, alors s est obligatoire pour $^{/}L^{\backslash}$, et représente un participant obligatoire de SIT($^{\lceil}L^{\rceil}$) ;
- si l'un des $^{/}L_i{}^{\backslash}$ possède une PosA interdite s, alors s est interdite pour $^{/}L^{\backslash}$, et représente un participant interdit pour SIT($^{\lceil}L^{\rceil}$) ;
- si l'un des $^{/}L_i{}^{\backslash}$ possède une PosA optionnelle s, et si aucun autre $^{/}L_j{}^{\backslash}$ n'a une PosA s obligatoire ou interdite, alors s peut être optionnelle, obligatoire, ou interdite pour $^{/}L^{\backslash}$;
- dans tous les autres cas, $^{/}L^{\backslash}$ est absurde et ne peut accepter d'instance.

limite du nombre de PosA. Bien que les critères précédemment formulés permettent de contraindre la définition de la structure actancielle d'un TUSemP, l'ensemble des PosA reste sous-spécifié. En effet, il est toujours possible pour un TUSemP $^{/}L^{\backslash}$ d'introduire de nouvelles PosA obligatoires ou optionnelles, tant que ces PosA représentent des participants de SIT($^{\lceil}L^{\rceil}$).

Par exemple, un TUSemP $^{/}L^{\backslash}$ pourrait posséder une, deux, ou même mille PosA pour représenter chacun des participants de SIT($^{\lceil}L^{\rceil}$). Le critère suivant a donc pour objectif de limiter le nombre de PosA.

Critère de limite du nombre de PosA des TUSemP, partie 2/2. Un TUSemP $^{/}L^{\backslash}$ ne peut introduire une nouvelle PosA représentant un participant ψ de SIT($^{\lceil}L^{\rceil}$) que si les deux conditions suivantes sont réunies :

- $^{\lceil}L^{\rceil}$ possède une PosASém correspondant à ψ ;
- aucune PosA d'aucun supertype de $^{/}L^{\backslash}$ ne représente ψ.

Conclusion sur l'exemple /outil\ /ciseaux\. La figure suivante illustre notre approche sur l'exemple problématique de la section 3.2.2 : le type d'unité sémantique profonde /ciseaux\ hérite à la fois du sens et de la structure actancielle de /outil\, et la structure actancielle de /outil\ est en correspondance simple avec la structure actancielle de (outil). Notons que (ciseaux) ne spécialise pas (outil).

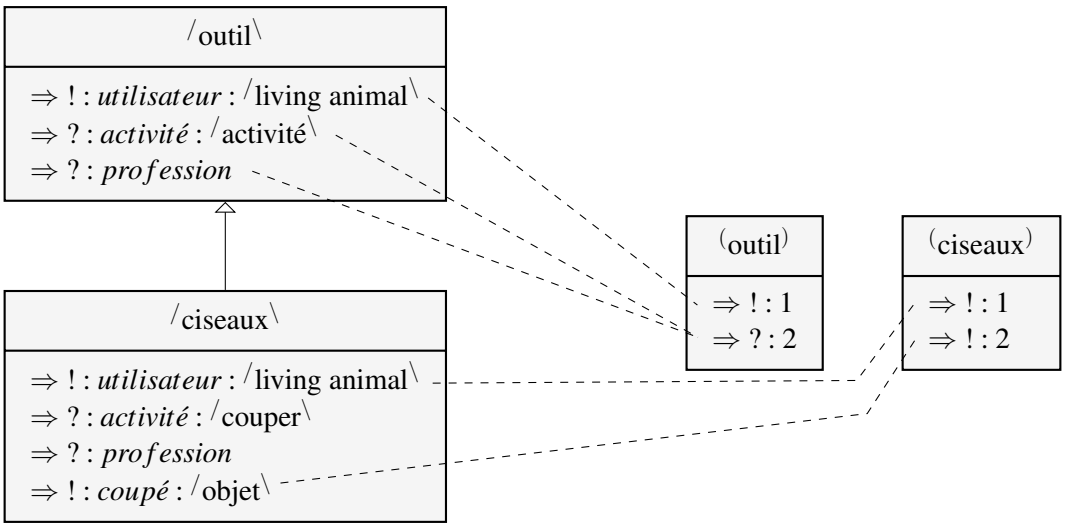


FIGURE 3.11 – Visualisations des types d'unité sémantique profonde /outil\ et /ciseaux\.

Conclusion

Nous pouvons maintenant répondre à la question de recherche posée en introduction de ce chapitre, à savoir : *Dans quelle mesure la conceptualisation actuelle des prédicats linguistiques doit-elle être étendue pour que sa formalisation ultérieure soit facilitée ?*

Nous avons apporté des précisions terminologiques et notationnelles à la TST, en différenciant type d'unité (dans la langue) et instance d'unité (dans l'usage). De plus, nous avons associé à chaque type d'unité sémantique *de surface* une structure actancielle qui correspond à la structure actancielle sémantique de son type de lexie associé.

Nous avons montré que l'inférence de type est intéressante si l'on organise les types de lexie et les types d'unité grammaticale dans une même hiérarchie. Par contre, nous avons justifié qu'une organisation en hiérarchie des types d'unité sémantique de surface (TUSemS) ne pouvait à la fois représenter une hiérarchie des sens, et permettre l'héritage et la spécialisation de la structure actancielle des types d'unité sémantique.

Nous avons donc repoussé la représentation des sens à un nouveau niveau sémantique profond, confondu avec le niveau des situations linguistiques. Les types d'unité sémantique profonde (TUSemP) portent le sens des types de lexie, et peuvent être vus comme des étiquettes sémantiques auxquelles on ajouterait une structure actancielle. Nous avons défini à l'aide de principes et de critères la structure actancielle des TUSemP, et avons introduit les deux notions fondamentales nécessaires suivantes :

- dans une situation linguistique, un participant optionnel hérité peut être spécialisé en participant interdit, et ainsi être neutralisé ;
- la notion correspondante dans la structure actancielle des TUSemP est la notion de Position Actancielle (PosA) interdite ;
- nous avons introduit la notion de signature dans la structure actancielle des TUSemP pour représenter la composante de contrainte des arguments des définitions lexicographiques.

Ainsi, les TUSemP peuvent être organisés en une hiérarchie qui satisfait les propriétés suivantes :

1. la hiérarchie des types d'unité sémantique profonde représente une hiérarchisation des sens ;
2. la structure actancielle des types d'unité sémantique profonde est héritée et éventuellement spécialisée au sein de la hiérarchie ;
3. il existe une correspondance simple entre la structure actancielle des types d'unité sémantique profonde, et celle des types d'unité sémantique de surface.

Dans le chapitre suivant, nous étudierons dans quelle mesure la conceptualisation actuelle des définitions lexicographiques doit être étendue pour que sa formalisation ultérieure soit facilitée. Puisque le niveau sémantique profond est celui qui porte les sens, il semble d'ores et déjà adapté d'y conceptualiser les définitions lexicographiques.

CHAPITRE 4

Conceptualisation des représentations sémantiques et des définitions lexicographiques

*— **L**a définition fait connaître ce qu'est la chose.*
Aristote

Sommaire

Introduction

Nous avons identifié un certain nombre de limites de la conceptualisation actuelle des définitions lexicographiques sous la forme d'une Représentation Sémantique (RSém) dans le chapitre 1, et le chapitre 3 a permis de préciser la conceptualisation des unités linguistiques dans la TST. Ce chapitre propose maintenant de préciser la conceptualisation des représentations linguistiques et des définitions lexicographiques dans la TST.

La question de recherche à laquelle ce chapitre répond peut donc être formulée de la manière suivante :

Dans quelle mesure la conceptualisation actuelle des représentations linguistiques et des définitions lexicographiques doit-elle être étendue afin de faciliter la formalisation ultérieure de la TST ?

Nous avons introduit une visualisation des instances d'unité linguistique dans le chapitre précédent. Nous l'étendrons aux représentations linguistiques dans la section 4.1.

Ensuite, nous savons qu'une définition lexicographique représentée sous la forme d'une RSém dans la TST représente une instanciation prototypique de la situation linguistique dénotée par une lexie $\ulcorner L \urcorner$. Or, une définition lexicographique doit définir un type d'unité linguistique, et donc se situer au niveau du dictionnaire. Par ailleurs, le chapitre précédent a justifié l'introduction du niveau sémantique profond pour représenter les sens, qui se confond avec la représentation des situations linguistiques. Nous proposerons donc dans la section 4.2 de porter les définitions lexicographiques au niveau du dictionnaire, et au niveau sémantique profond.

Finalement, nous avons illustré dans la section 1.3.2.3 que, dans les définitions lexicographiques, certains nœuds sont obligatoires et d'autres peuvent être optionnels. La section 4.2.3 étudiera donc la possibilité de faire de l'inférence et de la validation dans les définitions lexicographiques.

4.1 Conceptualisation des représentations linguistiques

Nous avons proposé une visualisation des instances d'unité linguistique dans les sections 3.1.2 et 3.3.1 du chapitre précédent. Ce sont ces unités linguistiques qui seront interconnectées pour former les différentes représentations linguistiques.

Avant d'illustrer nos propositions de visualisation sur les différentes représentations de la section 1.2.2, nous précisons dans la section suivante différents choix complémentaires de visualisation.

4.1.1 Choix complémentaires de visualisation

Cette section introduit les choix complémentaires de visualisation pour alléger la représentation graphique des étiquettes (§4.1.1.1), puis des différents types de relation que l'on peut rencontrer dans les représentations linguistiques (§4.1.1.2).

4.1.1.1 Visualisation des étiquettes de nœuds d'unité linguistique

Nous explicitons tout d'abord un ensemble de règles de visualisation qui permettent de simplifier la lecture d'une étiquette de nœud d'unité linguistique :

- si le type ou le marqueur d'un nœud est un singleton, les accolades peuvent être omises ;
- si le marqueur d'un nœud est vide (le nœud est donc anonyme), il peut être omis ainsi que le symbole ':' ;
- si le type d'un nœud est vide, il peut être omis, mais le symbole ':' doit être conservé ;
- si le type et le marqueur sont tous deux vides, alors l'étiquette peut être vide.

La figure 4.1 ci-dessous illustre ces règles. Sur une ligne donnée, les nœuds d'unité linguistique possèdent la même étiquette.

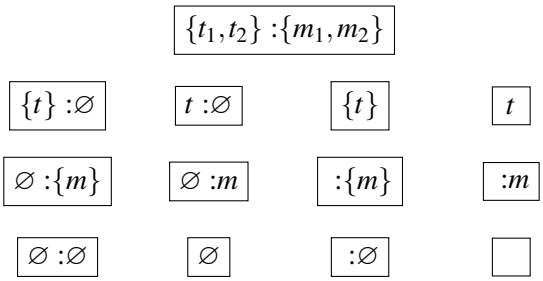


FIGURE 4.1 – Visualisation des nœuds d'unité linguistique et de leurs étiquettes. Sur une ligne donnée, les nœuds d'unité linguistique possèdent la même étiquette.

4.1.1.2 Relations actancielles, circonstancielles, et de coréférence anaphorique

La TST, en héritière des travaux de Tesnière (1959), scinde le jeu de relations syntaxiques en deux : les relations actancielles d'une part, et les relations circonstancielles d'autre part. Seules les relations actancielles ont leur place dans le dictionnaire, dans ce qu'on appelle la zone de combinatoire du DEC, et peuvent faire partie de la structure actancielle syntaxique profonde des unités lexicales (Mel'čuk, 2004b).

Nous proposons donc de différencier la visualisation des relations actancielles et des relations circonstancielles dans les RSynP de la manière suivante :

- les relations actancielles sont représentées par un trait double : $\Rightarrow$;
- les relations circonstancielles sont représentées par un trait simple : $\rightarrow$.

Par ailleurs, nous conservons la visualisation des relations de coréférence anaphoriques symétriques proposée par Mel'čuk (1997), c'est-à-dire sous la forme d'un trait discontinu à double flèche.

La figure 4.2 illustre les trois types d'arc que l'on peut rencontrer dans les représentations linguistiques.

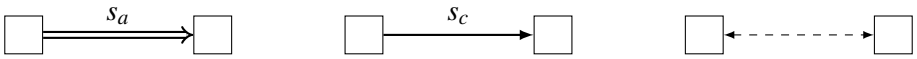


FIGURE 4.2 – Visualisation des arcs et de leurs étiquettes. De gauche à droite : une relation actancielle, une relation circonstancielle, une relation de coréférence anaphorique.

4.1.2 Visualisation des représentations linguistiques

Cette section introduit successivement les représentations syntaxiques profondes (§4.1.2.1), sémantiques de surface (§4.1.2.2), et sémantiques profondes (§4.1.2.3).

4.1.2.1 Représentations syntaxiques profondes

Les représentations syntaxiques profondes introduites dans la section 1.2.2.2 mettent en jeu des instances de lexie, un jeu de neuf relations syntaxiques universelles, et des relations de coréférence anaphoriques symétriques. Les relations syntaxiques universelles sont scindées entre les relations actancielles **I** à **VI** d'une part, et les relations circonstancielles **ATTR**(ibutive),**COORD**(inative) et **APPEND**(itive) d'autre part. La figure 4.3 reprend l'exemple de la figure 1.3, et l'augmente de nos précisions de visualisation.

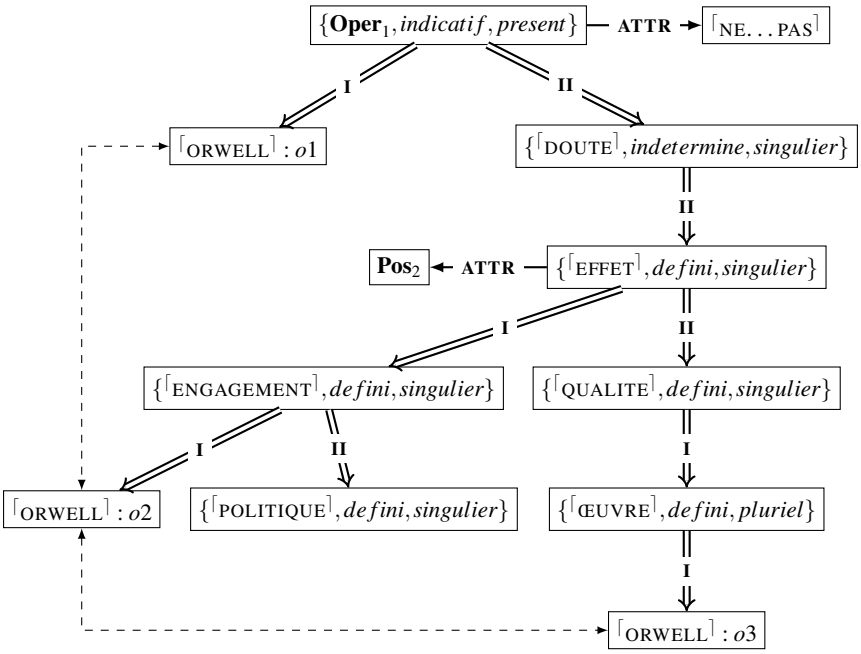


FIGURE 4.3 – Exemple augmenté d'une représentation syntaxique profonde (RSynP) de la phrase : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997), et augmenté de nos précisions de visualisation.

Remarquons dans la RSynP ci-dessus que les trois nœuds co-référents possèdent un marqueur différent. Nous pouvons entrevoir un premier type d'inférence souhaitable avec les représentations linguistiques lorsqu'elles seront formalisées :

- les nœuds marqués $o1$ et $o3$ sont également co-référents ;
- les trois identifiants d'unité linguistique $o1$, $o2$, et $o3$, identifient la même unité linguistique (quelles que soient leurs URI). Nous pouvons remplacer le marqueur de ces trois nœuds par $\{o1, o2, o3\}$.

4.1.2.2 Représentations sémantiques de surface

Les représentations sémantiques de surface introduites dans la section 1.2.2.1 mettent en jeu des instances d'unité sémantique de surface, ainsi qu'un ensemble de relations actancielles sémantiques numérotées. La figure 4.4 reprend l'exemple de la figure 1.2, et l'augmente de nos précisions de visualisation.

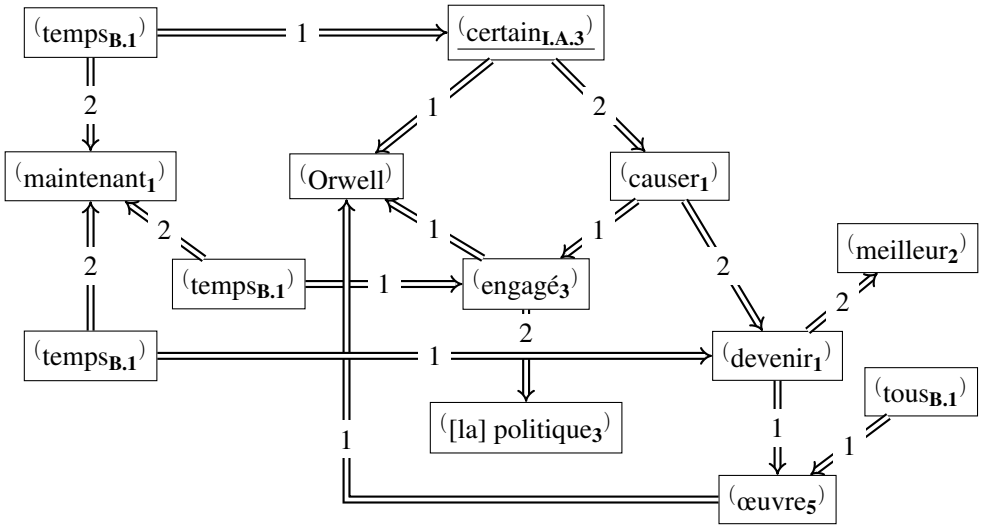


FIGURE 4.4 – Exemple augmenté d'une représentation sémantique de surface (RSémS). Cette RSémS peut être exprimée par un nombre élevé de phrases en français, comme par exemple : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997), et augmenté de nos précisions de visualisation.

4.1.2.3 Représentations sémantiques profondes

Les représentations sémantiques profondes font partie de l'extension de la conceptualisation de la TST que nous proposons. Elles mettent en jeu des instances d'unité sémantique profonde, ainsi qu'un ensemble potentiellement infini de rôles sémantiques lexicalisés. La figure 4.5 illustre une RSémP correspondant à la Représentation Sémantique de Surface (RSémS) de la section précédente. La représentation sémantique profonde est obtenue en choisissant pour chaque TUSemP une structure actancielle selon les principes et critères formulés dans la section 3.3.3.

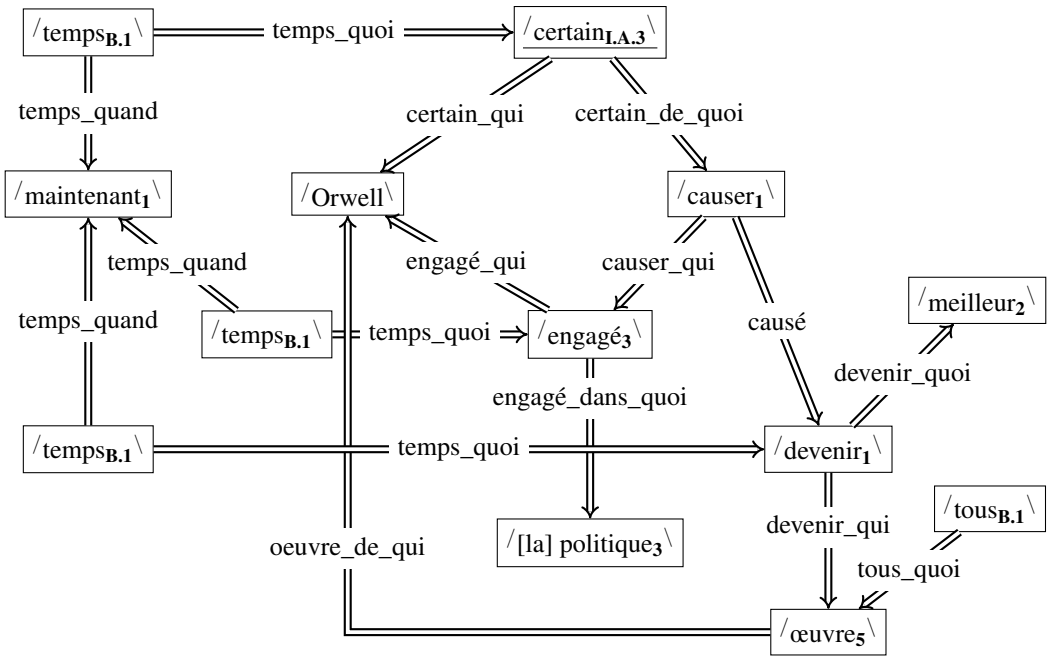


FIGURE 4.5 – Exemple augmenté d'une représentation sémantique profonde (RSémP). Cette RSém peut être exprimée par un nombre élevé de phrases en français, comme par exemple : *Orwell n'a pas de doute quant à l'effet positif de son engagement politique sur la qualité de ses œuvres.* Exemple repris de Mel'čuk (1997), et augmenté de nos précisions de visualisation.

4.2 Conceptualisation des définitions lexicographiques

Une représentation d'une définition en langue naturelle, aussi formalisée soit-elle, reste une linéarisation qui encode le sens de la lexie définie. D'un autre côté, une définition lexicographique représentée sous la forme d'une RSémS met en scène des (instances d')unités sémantiques. Il s'agit donc d'une instanciation prototypique de la situation linguistique dénotée par une lexie $\ulcorner \mathrm{L} \urcorner$. Nous avons par ailleurs montré dans la section 1.3.2.3 qu'il est actuellement impossible de représenter ainsi les composantes de la définition qui sont optionnelles, que le sens d'un type de lexie est porté par un TUSemP, et que le niveau sémantique profond encode les situations linguistiques. Nous proposons donc dans la section 4.2.1 de reconceptualiser les définitions lexicographiques :

- au niveau sémantique profond,
- au niveau du dictionnaire.

Nous préciserons ensuite la notion de définition lexicographique formelle (§4.2.2), et étudierons la possibilité d'y faire de l'inférence et de la validation (§4.2.3).

4.2.1 Repositionnement dans le modèle Sens-Texte

Nous partons donc de la conceptualisation proposée par Mel'čuk *et al.* (1995) qui prend la forme d'une RSémS.

Soit $\ulcorner \mathrm{L} \urcorner$ le type de lexie défini. Notre point de départ est une RSémS telle que décrite dans la section 1.3.2.2 :

- le nœud qui représente le genre proche est marqué comme communicativement dominant (son étiquette est soulignée) ;
- certains nœuds sont marqués par une lettre X, Y,..., et représentent les PosA du TUSemS associées au type de lexie défini.

Cette RSémS est une instanciation prototypique de SIT($\ulcorner \mathrm{L} \urcorner$) au niveau sémantique de surface.

4.2.1.1 Repositionnement au niveau sémantique profond

Repositionnons dans un premier temps les définitions lexicographiques au niveau sémantique profond.

Pour chaque TUSemS (L_i) présent dans la RSémS, nous supposons connus :

- le TUSemP correspondant $/\mathrm{L}_i\backslash$ et sa structure actancielle ;
- la correspondance entre la structure actancielle de (L_i) et celle de $/\mathrm{L}_i\backslash$.

La RSém doit d'abord être repositionnée au niveau sémantique profond de la manière suivante :

1. Pour tout nœud de la RSémS, remplacer l'étiquette du nœud $\{(\mathrm{L}_1),(\mathrm{L}_2),\ldots\} : \{m_1, m_2, \ldots\}$, par $\{/\mathrm{L}_1\backslash, /\mathrm{L}_2\backslash, \ldots\} : \{m_1, m_2, \ldots\}$.
2. Pour tout arc a étiqueté i entre les nœuds u_1 et u_2 :
   (a) pour tout TUSemS (L_i) dans le type de u_1, ajouter un arc entre u_1 et u_2, avec chacun des symboles de PosA de $/\mathrm{L}_i\backslash$ qui correspondent à la PosA i de (L_i).
   (b) supprimer a.

Pour illustrer ce processus, nous repartons de la définition lexicographique du type de lexie $\ulcorner \textsc{peigne}_{\mathbf{B.2.D}} \urcorner$, illustrée en figure 1.8. La figure 4.6 en illustre le repositionnement au niveau sémantique profond.

La représentation sémantique profonde ainsi obtenue peut être vue comme une instanciation prototypique de SIT($\ulcorner \mathrm{L} \urcorner$).

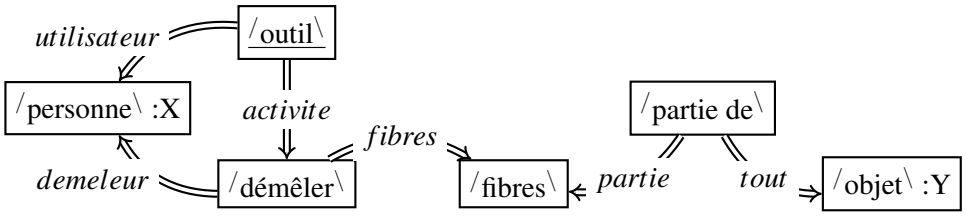


FIGURE 4.6 – Repositionnement de la définition lexicographique de $\lceil$PEIGNE$_{\mathbf{B.2.D}}\rceil$ au niveau sémantique profond.

4.2.1.2 Repositionnement au niveau du dictionnaire

Afin de repositionner les définitions lexicographiques au niveau du dictionnaire, chacun des nœuds ne doit plus représenter une instance, mais un type d'unité sémantique profonde, qui identifie un participant de SIT($\lceil$L$\rceil$). Nous parlerons donc en termes de *nœud de participant* dans cette dernière représentation.

Nous supposons connue la structure actancielle du TUSemP associé à $\lceil$L$\rceil$. En particulier, et à des fins d'illustration, nous supposons que /outil\ possède une PosA optionelle *profession* de signature /profession\, dont /peigne$_{\mathbf{B.2.d}}$\ hérite et qu'ilspécialise en PosA obligatoire de signature /tisserand\. On doit donc imposer la contrainte de validation suivante dans la définition lexicographique de $\lceil$L$\rceil$.

Définition 20 (Contrainte partielle de validation des définitions lexicographiques : correspondance entre les structures actancielles)**.** Soit *a* un arc symbolisé par *s* partant du nœud communicativement dominant u_0 d'étiquette $\{/L_1\backslash, /L_2\backslash, \ldots\}$, et menant vers un nœud d'étiquette u_2 $\{/L'_1\backslash, /L'_2\backslash, \ldots\}$: *m* qui représente une PosASém de $\lceil$L$\rceil$. Soit *i* le numéro associé à la PosA *m* dans la structure actancielle de (L). Alors la PosA *i* de (L) doit correspondre à la PosA *s* de /L\.

Par exemple, les structure actancielles de $(\text{peigne}_{\mathbf{B.2.d}})$ et $(\text{peigne}_{\mathbf{B.2.d}})$ doivent au moins posséder la correspondance illustrée par la figure 4.7.

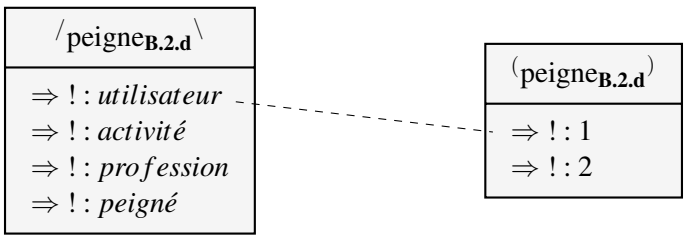


FIGURE 4.7 – Correspondance partielle entre les structures actancielles des types d'unité sémantique de surface et profonde imposée par la contrainte de validation des définitions lexicographiques.

Chaque nœud représentant une IUSemP est donc remplacé par un nœud de participant, illustré par un rectangle à deux parties :

- La partie du dessus contient une étiquette $\{/L_1\backslash, /L_2\backslash, \ldots\} : \{s\}$:
  - $\{/L_1\backslash, /L_2\backslash, \ldots\}$ est le type du nœud remplacé ;
  - si le nœud remplacé possède un marqueur *m* (*m* représente un symbole X,Y,...), remplacer *m* par le symbole *s* correspondant dans la structure actancielle de /L\ ;
- La partie du dessous contient l'union des structures actancielles des TUSemP dans $\{/L_1\backslash, /L_2\backslash, \ldots\}$, hormis celles qui sont déjà illustrées dans la représentation.

La figure 4.8 illustre la sortie de ce processus appliqué au type de lexie ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉.

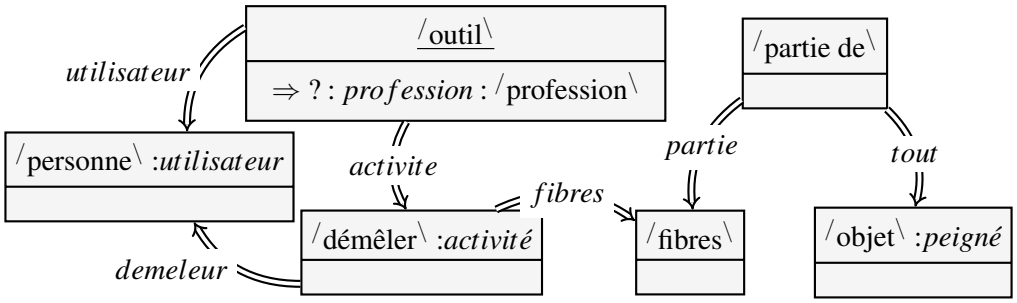


FIGURE 4.8 – Repositionnement de la définition lexicographique de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ au niveau sémantique profond et au niveau du dictionnaire.

Nous imposons finalement une dernière contrainte de validation aux définitions lexicographiques, qui concerne la signature du TUSemP associé à un type de lexie défini.

Définition 21 (Contrainte de validation des définitions lexicographiques : signature)**.** Soit u un nœud marqué s. La signature de /L\ pour s doit être le type du nœud u.

Par exemple la structure actancielles de (peigne$_{\mathbf{B.2.d}}$) est illustrée par la figure 4.9.

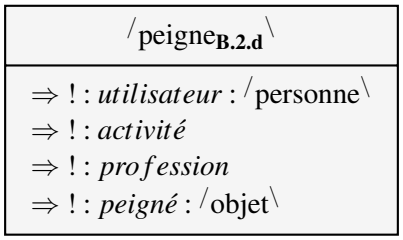


FIGURE 4.9 – Signature imposée par la contrainte de validation des définitions lexicographiques.

4.2.1.3 Explicitation de la nature des Positions Actancielles (PosA)

Ce qui reste absent de ces représentations dans la TST est la nature des PosA en jeu. En effet, nous avons montré dans la section 3.3.3.2 que chaque PosA d'un TUSemP peut être obligatoire, optionnelle, ou interdite.

Nous proposons de visualiser cette information sur la définition lexicographique de la manière suivante. Soit a un arc étiqueté par s, partant d'un nœud de participant u_1, et menant à un nœud de participant u_2. Soit /L\ un TUSemP dans le type de u_1 pour lequel s représente une PosA. On préfixe l'étiquette de a par le symbole '!', '?', ou '0', suivant que la PosA est obligatoire, optionnelle, ou interdite. La figure 4.10 ci-dessous représente ces trois préfixes.

De plus, nous connaissons la nature d'au moins quelques participants dans SIT(⌈L⌉) :

- le participant central est obligatoire ;
- un participant qui correspond à une PosA obligatoire de TUSemP est obligatoire ;
- un participant qui correspond à une PosA optionnelle de TUSemP est optionnel ;
- un participant qui correspond à une PosA interdite de TUSemP est interdit.

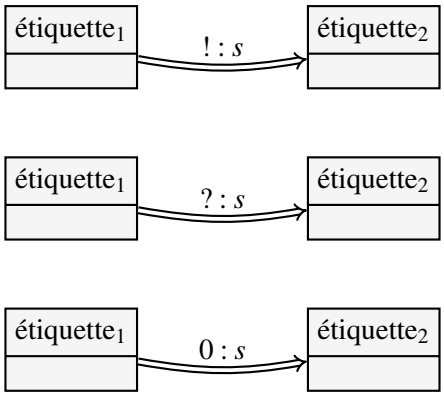


FIGURE 4.10 – Les symboles '!', '?', ou '0' préfixent les étiquettes d'arc dans les définitions lexicographiques.

Nous proposons de visualiser la nature des participants de SIT($\lceil$L$\rceil$) dans la définition lexicographique en préfixant l'étiquette de chaque nœud de participant par le symbole '!', '?', ou '0', suivant que le participant est obligatoire, optionnel, ou interdit. Pour le moment, tout participant dont on ne connait pas la nature est supposé être optionnel. La figure 4.11 ci-dessous représente ces trois préfixes.

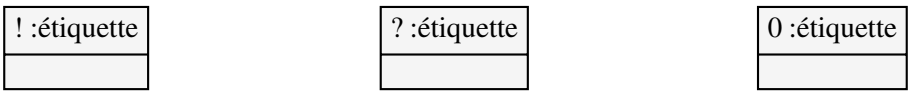


FIGURE 4.11 – Les symboles '!', '?', ou '0' préfixent les étiquettes des nœuds de participant dans les définitions lexicographiques.

La figure 4.12 illustre la sortie de ce processus appliqué au type de lexie $\lceil$PEIGNE$_{\mathbf{B.2.D}}\rceil$.

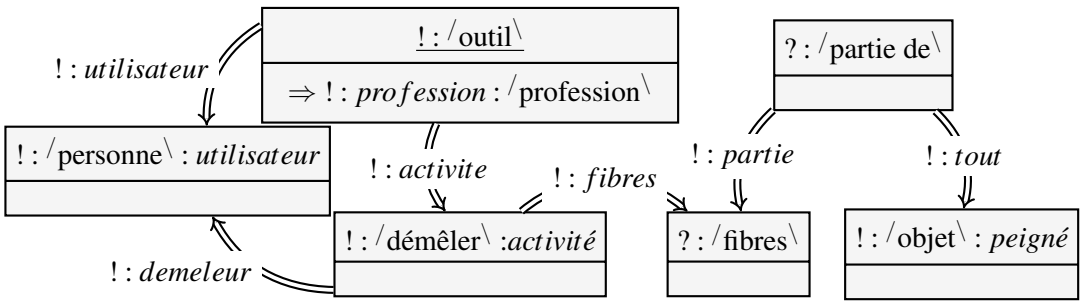


FIGURE 4.12 – Précision de la nature des PosA et des participants dans la définition lexicographique.

4.2.2 Définition des définitions lexicographiques formelles

Maintenant que nous avons repositionné les définitions lexicographiques au niveau du dictionnaire et au niveau sémantique profond, et que nous y avons précisé la nature des PosA et des participants, nous pouvons formuler une définition formelle des définitions lexicographiques formelles.

Définition 22 (Définition lexicographique formelle)**.** Une définition lexicographique formelle d'un type de lexie $\ulcorner \mathrm{L} \urcorner$ est un n-uplet $D(/\mathrm{L}\backslash) = \langle p_0, P, A, type, nature, marker \rangle$, où :

- P est un ensemble de nœuds de participant. Chacun représente un participant de SIT($\ulcorner \mathrm{L} \urcorner$) ;
- $p_0 \in P$ est le participant central de SIT($\ulcorner \mathrm{L} \urcorner$) ;
- A est un ensemble de triplets de PosA. $\langle p_1, s, p_2 \rangle \in A$ signifie que le participant p_2 remplit une PosA s du participant p_1.
- $type$ est une fonction qui associe à tout $p \in P$ un type $type(p)$, qui est un ensemble (potentiellement vide) de TUSemP. $type(p_0) = genus(/\mathrm{L}\backslash)$ dénote le genre proche de $/\mathrm{L}\backslash$.
- $nature$ est une fonction qui associe à chaque arc de PosA de A, et à chaque nœud de participant de P, une nature choisie dans l'ensemble {'!','?','0'}.
- $marker$ est une fonction qui associe à chaque $p \in P$ un ensemble (potentiellement vide) de symboles de PosA. $marker(p) \neq \varnothing$ signifie que p est une PosA de $/\mathrm{L}\backslash$.

Une définition lexicographique doit satisfaire les conditions de validité suivantes :

- $nature(p_0) = !$;
- Pour tout $\langle p_1, s, p_2 \rangle \in A$, s est une PosA de $type(p_1)$, dont la nature est $nature(\langle p_1, s, p_2 \rangle)$.
- $\langle p_0, s, p \rangle \in A \Rightarrow s \in marker(p)$.

Ainsi, la section précédente 4.2.1 a introduit un processus pour générer la définition lexicographique formelle d'un type de lexie $\ulcorner \mathrm{L} \urcorner$ à partir des connaissances suivantes :

1. la structure actancielle sémantique de $\ulcorner \mathrm{L} \urcorner$;
2. une RSém qui représente l'instanciation prototypique de la décomposition sémantique de $\ulcorner \mathrm{L} \urcorner$;
3. la structure actancielle de $/\mathrm{L}\backslash$;

4.2.3 Validation et inférence dans les définitions lexicographiques

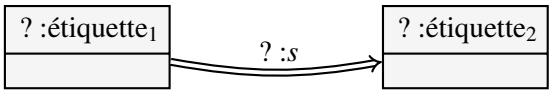
Soit la définition lexicographique formelle d'un type de lexie $\ulcorner \mathrm{L} \urcorner$: $D(/\mathrm{L}\backslash) = \langle p_0, P, A, type, nature, marker \rangle$. Soit l'arc $\langle p_1, s, p_2 \rangle \in A$. Chacun des deux nœuds de participant p_1 et p_2 peuvent être obligatoires, optionnels, ou interdits. De plus, la PosA s du type de p_1 peut également être obligatoire, optionnelle, ou interdite.

Les tableaux de la figure 4.13 décrit chacune des $3^3 = 27$ configurations possibles pour les arcs d'une définition lexicographique formelle.

La plupart de ces configurations sont stables et ne permettent pas de déduire plus d'information sur la définition lexicographique formelle. Concentrons-nous maintenant sur les six configurations les plus intéressantes pour notre étude :

Deux configurations mènent à des inconsistances dans la définition lexicographique formelle, ce qui signifie que le type $/\mathrm{L}\backslash$ est absurde et ne peut pas être instancié.

1. Un participant obligatoire possède une PosA obligatoire qui est remplie par un participant interdit (cf., figure 4.14, haut).
2. Un participant obligatoire possède une PosA interdite qui est remplie par un participant obligatoire (cf., figure 4.14, bas).

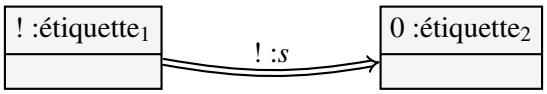


!:*s*	!:étiquette$_2$	?:étiquette$_2$	0:étiquette$_2$
!:étiquette$_1$	✓	⚠	✗
?:étiquette$_1$	✓	✓	⚠
0:étiquette$_1$	✓	✓	✓

?:*s*	!:étiquette$_2$	?:étiquette$_2$	0:étiquette$_2$
!:étiquette$_1$	✓	✓	✓
?:étiquette$_1$	✓	✓	✓
0:étiquette$_1$	✓	✓	✓

0:*s*	!:étiquette$_2$	?:étiquette$_2$	0:étiquette$_2$
!:étiquette$_1$	✗	⚠	✓
?:étiquette$_1$	⚠	✓	✓
0:étiquette$_1$	✓	✓	✓

FIGURE 4.13 – Description des 27 configurations possibles pour les arcs d'une définition lexicographique formelle. Les configurations stables sont marquées ✓ ; celles qui permettent l'inférence sont marquées ⚠ ; et celles qui sont inconsistantes sont marquées ✗.



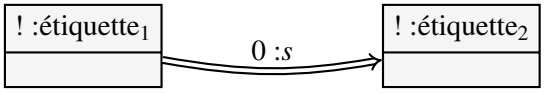


FIGURE 4.14 – Les deux configurations qui rendent une définition lexicographique formelle inconsistante.

Ces deux configurations inconsistantes impliquent quatre autres configurations intéressantes, qui permettent d'inférer de nouvelles connaissances dans la définition lexicographique formelle. Sur la figure 4.15 et de haut en bas, on trouve les configurations suivantes :

1. Un participant obligatoire ! : étiquette$_1$ possède une PosA obligatoire ! : s qui est remplie par un participant optionnel ? : étiquette$_2$. Puisque le participant ? : étiquette$_2$ ne peut pas être interdit (cas absurde 1), il peut être spécialisé à obligatoire.
2. Un participant obligatoire ! : étiquette$_1$ a une PosA interdite 0 : s qui est remplie par un participant optionnel ? : étiquette$_2$. Puisque le participant ? : étiquette$_2$ ne peut pas être obligatoire (cas absurde 2), il peut être spécialisé à interdit.
3. Un participant optionnel ? : étiquette$_1$ a une PosA obligatoire ! : s qui est remplie par un participant interdit 0 : étiquette$_2$. Puisque le participant ? : étiquette$_1$ ne peut pas être obligatoire (cas absurde 1), il peut être spécialisé à interdit.
4. Un participant optionnel ? : étiquette$_1$ a une PosA interdite 0 : s qui est remplie par un participant obligatoire ! : étiquette$_2$. Puisque le participant ? : étiquette$_1$ ne peut pas être obligatoire (cas absurde 2), il peut être spécialisé à interdit.

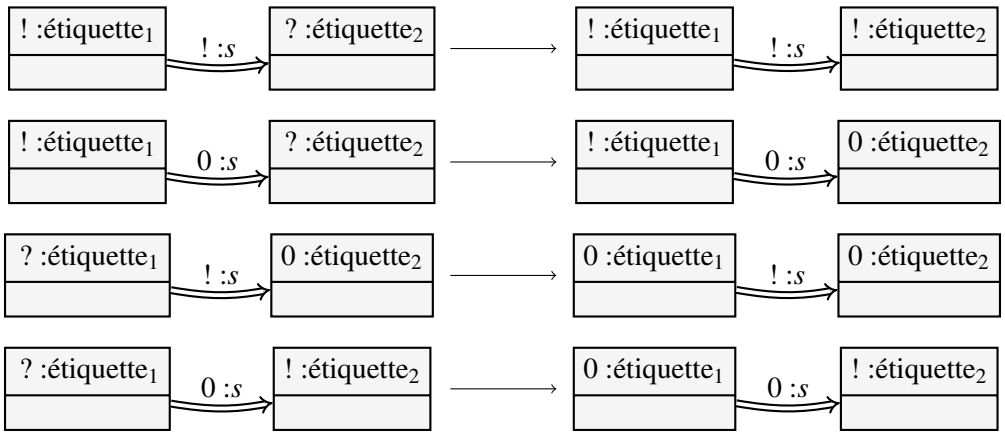


FIGURE 4.15 – Les quatre configurations qui permettent de faire de l'inférence dans une définition lexicographique formelle.

Lorsque l'on détecte une possibilité d'inférence dans une définition lexicographique formelle, et après avoir effectué le changement, de nouveaux tests peuvent être menés pour vérifier si un autre cas d'inférence ou d'inconsistance est détecté. Vérifier la consistance d'une définition lexicographique formelle consiste donc à appliquer les transformations possibles jusqu'à ce qu'aucune ne puisse être effectuée. Si aucun arc de PosA n'est problématique dans la définition lexicographique formelle après cela, alors elle est consistante.

Conclusion

Nous pouvons maintenant répondre à la question de recherche posée en introduction de ce chapitre, à savoir : *Dans quelle mesure la conceptualisation actuelle des représentations linguistiques et des définitions lexicographiques doit-elle être étendue afin de faciliter la formalisation ultérieure de la TST ?*

Nous avons proposé dans la section 4.1 des choix de visualisation pour les représentations linguistiques, et nous avons illustré notre approche sur les exemples de la section 1.2.2.

Nous avons ensuite détaillé une nouvelle conceptualisation pour les définitions lexicographiques : nous avons repositionné ces définitions au niveau sémantique profond, qui porte les sens (§4.2.1.1) ; puis au niveau du dictionnaire (§4.2.1.2), pour aboutir à une représentation formalisée de la situation linguistique dénotée par le type de lexie à définir SIT($\ulcorner$L$\urcorner$). Enfin, nous avons explicité la nature des participants de SIT($\ulcorner$L$\urcorner$), et des différentes positions actancielles (PosA) que ces participants possèdent (§4.2.1.3).

Après avoir proposé une formalisation des définitions lexicographiques formelles (§4.2.2), nous avons finalement montré dans la section 4.2.3 qu'il est possible d'y mener une étude de validation et d'y inférer de nouvelles connaissances. Nous avons pour cela étudié les 27 configurations que peuvent prendre chaque arc de PosA. Nous avons montré que 2 d'entre elles impliquent l'inconsistance de la définition lexicographique formelle, et que de ces deux configurations problématiques découlent quatre configurations sources d'inférence.

Maintenant que nous avons complété une extension de la conceptualisation de la TST en ce qui concerne les prédicats linguistiques, les représentations linguistiques, et les définitions lexicographiques, nous souhaitons la valider auprès des lexicographes de la TST. C'est le sujet du chapitre suivant de ce mémoire.

CHAPITRE 5

Application de la conceptualisation étendue dans un projet de lexicographie explicative et combinatoire

---

*_____ **C**e n'est pas grand-chose d'avoir des idées, le tout est de les appliquer, c'est-à dire de penser par elles les dernières différences.*
Emile-Auguste Chartier, dit Alain, Propos sur l'éducation (1932)

Sommaire

Introduction

Après avoir étendu la conceptualisation des prédicats linguistiques et des définitions lexicographiques, nous avons souhaité l'utiliser dans un projet de lexicographie explicative et combinatoire. Nous nous attacherons donc dans ce chapitre à répondre à la question de recherche suivante :

Comment l'extension de la conceptualisation proposée peut-elle être utilisée pour l'édition de la zone sémantique dénotationnelle du DEC ?

Pour répondre à cette question, nous nous sommes tourné vers les lexicographes du projet RELIEF (Lux-Pogodalla et Polguère, 2011), décrit dans la section 1.2.3.2, puisqu'il s'agit du projet de lexicographie qui présente la conceptualisation de la zone sémantique dénotationnelle du DEC la plus aboutie.

Nous présentons dans ce chapitre la conception, le développement, et la validation d'un premier prototype d'éditeur de définitions lexicographiques formalisées selon notre conceptualisation. Le développement et l'évaluation de l'éditeur a fait l'objet du stage orienté IHM de Master 2 de Romain GUGERT pendant l'été 2013 (Gugert, 2013), coencadré avec Alain GIBOIN. Nous présentons donc la substantifique moëlle de ces travaux, ainsi que notre travail de conception fait en amont.

La section 5.1 détaille dans un premier temps le scénario actuel de rédaction d'une définition lexicographique dans le projet RELIEF. Nous proposons ensuite dans la section 5.2 un nouveau scénario adapté à notre conceptualisation étendue. Finalement, la section 5.3 présente le prototype d'éditeur de définitions lexicographiques formelles, ainsi que les premiers retours des lexicographes du projet RELIEF sur ce prototype.

5.1 La rédaction d'une définition lexicographique : scénario actuel

Le logiciel d'édition développé dans le projet RELIEF se nomme MVSDicet. La zone sémantique dénotationnelle y est toujours en cours de développement, et les définitions lexicographiques y sont représentées à l'aide des trois éléments principaux suivants :

- une hiérarchie d'étiquettes sémantiques (Polguère, 2011) ;
- le type de balisage développé pour le projet Definiens (Barque, 2008; Barque *et al.*, 2010) ;
- la désambiguïsation des mots pleins dans le texte.

Présentons un scénario dans lequel Alain, lexicographe en chef, demande à Sophie, lexicographe, de définir le type de lexie ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉. Rappelons que ce type de lexie sert d'illustration à la notion de définition lexicographique introduit dans la section 1.3.2, et possède la définition suivante dans Mel'čuk *et al.* (1999) :

peigne$_{\mathbf{B.2.d}}$ *de personne X pour objet Y* =
⟨Outil de tissage qu'une personne X utilise pour démêler$_{\mathbf{2}}$ les fibres d'un objet Y⟩

Suite à des entretiens avec des lexicographes, nous avons déterminé que Sophie réaliserait aujourd'hui cette tâche en cinq étapes :

1. Sophie cherche d'abord une étiquette sémantique dans la hiérarchie des étiquettes sémantiques développée par Alain (Polguère, 2011). Elle choisit `outil` ;
2. Sophie détermine ensuite la structure actancielle de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉ à l'aide de différents critères linguistiques : une personne X et un objet Y. Elle cherche ensuite une forme propositionnelle qui lui convient dans une hiérarchie développée par Alain. Elle choisit : ∼ de X pour Y.
3. Sophie rédige alors la définition et balise en XML le genre proche et les différences spécifiques à l'aide du type de balisage développé pour le projet Definiens (Barque, 2008; Barque *et al.*, 2010).
4. Finalement, Sophie spécifie le type de lexie auquel chaque mot significatif réfère.

Nous supposons que la sortie de ce processus est la suivante :

```
<CC label="outil" >outil de tissage</CC>
<PC role="utilisation" >qu'une personne X utilise pour démêler<sub>2</sub> les fibres
d'un objet Y</PC>
```

Cette représentation ne permet de déterminer ni la structure actancielle du TUSemP /peigne$_{\mathbf{B.2.d}}$\, ni la définition lexicographique formelle de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉. Nous souhaitons donc proposer une nouvelle tâche, centrée sur la conceptualisation étendue que nous avons définie.

5.2 Proposition pour un nouveau scénario

Dans le chapitre précédent, la formalisation d'une définition lexicographique d'un type de lexie ⌈L⌉ prenait en entrée l'instanciation prototypique de la situation linguistique dénotée par ⌈L⌉, c'est-à-dire une RSém. Ici, au contraire, aucune RSém n'est *a priori* disponible.

Nous proposons donc un processus formé de quatre tâches principales (cf., figure 5.1, à droite) :

1. définition de la structure actancielle du TUSemP (§5.2.1) ;
2. élaboration de la définition lexicographique formelle (§5.2.2) ;
3. sélection de la structure actancielle du TUSemS (§5.2.3) ;
4. définition des correspondances entre les structures actancielles du TUSemP et du TUSemS (§5.2.4).

Processus suivi dans le projet RELIEF	Processus à suivre avec le prototype d'éditeur
1. Sélection d'une étiquette sémantique dans une hiérarchie (Polguère, 2011) 2. Sélection de la structure actancielle sémantique de surface (Mel'čuk, 2004a) 3. Rédaction de la définition lexicographique en XML (Barque *et al.*, 2010)	1. Sélection de la structure actancielle de la sémantique profonde 2. Édition de la définition formelle 3. Sélection de la structure actancielle sémantique de surface 4. Mise en correspondance des structures actancielles

TABLE 5.1 – à gauche, processus des lexicographes du projet RELIEF ; à droite, processus du prototype d'éditeur de définitions lexicographiques formelles.

5.2.1 Définition de la structure actancielle de l'étiquette sémantique

La tâche de définition de la structure actancielle de /peigne$_{\textbf{B.2.d}}$\ consiste, pour Sophie, à choisir dans la hiérarchie des TUSemP le plus proche parent de /peigne$_{\textbf{B.2.d}}$\, et à en spécialiser ensuite la structure actancielle.

Sophie commence par ouvrir une nouvelle fenêtre dans l'éditeur. Le TUSemP /peigne$_{\textbf{B.2.d}}$\ apparaît dans une boîte vide comme illustré par la figure 5.1a. Ensuite, Sophie doit sélectionner l'étiquette sémantique de ⌈PEIGNE$_{\textbf{B.2.D}}$⌉, qui correspond au plus proche parent de /peigne$_{\textbf{B.2.d}}$\. Elle ouvre pour cela une fenêtre de navigation dans la hiérarchie des TUSemP, et peut sélectionner le TUSemP /outil\ (cf., fig. 5.1b). Selon notre conceptualisation étendue des prédicats linguistiques, les TUSemP (= étiquettes sémantiques) possèdent une structure actancielle, qui est héritée et éventuellement spécialisée. Par défaut, /peigne$_{\textbf{B.2.d}}$\ hérite donc de la structure actancielle de /outil\ (cf., fig. 5.1c), que nous supposons définie de la manière suivante :

– une PosA obligatoire *utilisateur*, de signature /personne\ ;
– une PosA obligatoire *activité*, de signature /activité\ ;
– une PosA optionnelle *profession*, de signature /profession\.

Sophie peut finalement spécialiser la structure actancielle de /peigne$_{\textbf{B.2.d}}$\ :

1. /peigne$_{\textbf{B.2.d}}$\ est conçu pour démêler. Sophie clique donc sur /activité\, et choisit le sous-type /démêler\ dans la hiérarchie des TUSemP.
2. /peigne$_{\textbf{B.2.d}}$\ est conçu pour la profession de tisserand. Sophie clique donc sur /profession\, et choisit le sous-type /tisserand\ dans la hiérarchie des TUSemP.
3. la PosA *profession* est obligatoire pour /peigne$_{\textbf{B.2.d}}$\. Sophie clique donc sur le symbole '⇒? :', qui devient '⇒! :'.
4. /peigne$_{\textbf{B.2.d}}$\ introduit une nouvelle PosA obligatoire *peigné* pour la variable Y, avec la signature /objet\ (pour l'objet dont on veut démêler les fibres). Sophie clique donc sur le symbole ⊞ Nouvelle PosA , et remplit un formulaire :
   (a) elle introduit un nouveau rôle sémantique lexicalisé *peigné* ;
   (b) elle spécifie que cette PosA est obligatoire ;
   (c) elle spécifie la signature comme étant /objet\.

En sortie de cette étape, la description de la structure actancielle de /peigne$_{\textbf{B.2.d}}$\ est complète et telle que détaillée sur la figure 5.1d.

/peigne$_{\textbf{B.2.d}}$\ ≲ ?
⊞ Nouvelle PosA

(a) Point de départ pour définir ⌈PEIGNE$_{\textbf{B.2.D}}$⌉

▽ /entité\ [(⇒) 1]
  ▽ /objet physique\ [(⇒) 1]
    ▽ /artefact\ [(⇒) 1]
      ▽ /instrument\ [⇒ 2]
        ▷ /outil\ [(⇒) 1, ⇒ 2]

(b) Vue de la hiérarchie des TUSemP

/peigne$_{\textbf{B.2.d}}$\ ≲ /outil\
⇒ ! : *utilisateur* : /personne\
⇒ ! : *activité* : /activité\
⇒ ? : *profession* : /profession\
⊞ Nouvelle PosA

(c) /peigne$_{\textbf{B.2.d}}$\ hérite de la structure actancielle de /outil\

/peigne$_{\textbf{B.2.d}}$\ ≲ /outil\
⇒ ! : *utilisateur* : /personne\
⇒ ! : *activité* : /démêler\
⇒ ! : *profession* : /tisserand\
⇒ ! : *peigné* : /objet\
⊞ Nouvelle PosA

(d) Structure actancielle de /peigne$_{\textbf{B.2.d}}$\ complétée.

FIGURE 5.1 – Définition de la structure actancielle de /peigne$_{\textbf{B.2.d}}$\.

5.2.2 Élaboration de la définition lexicographique formelle par manipulation de graphe

La prochaine tâche est de construire le graphe qui représente SIT($\lceil$PEIGNE$_{\mathbf{B.2.D}}\rceil$). Le point de départ est la représentation de la structure actancielle de /peigne$_{\mathbf{B.2.d}}$\, comme illustré par la figure 5.1d.

L'idée principale ici est que chacun des TUSemP en signature des PosA peut lui-même posséder une structure actancielle. Souvenons-nous des deux visualisations équivalentes de la signature TUSemP présentées dans la figure 3.10. Sophie peut donc sortir certaines PosA en dehors de la boîte par glisser-déposer. Cela permet d'expliciter par exemple que /démêler\ possède deux PosA obligatoires (cf., figure 5.2) : *démêleur* et *fibres*.

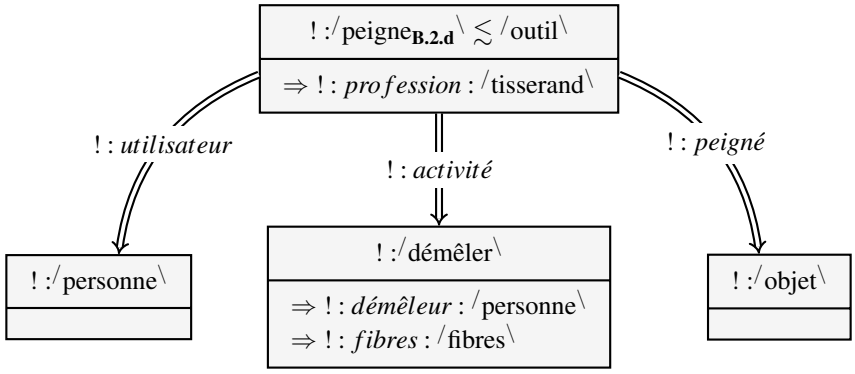


FIGURE 5.2 – Définition lexicographique formelle de /peigne$_{\mathbf{B.2.d}}$\ : attribution d'un nœud aux participants intéressants de la structure actancielle de /peigne$_{\mathbf{B.2.d}}$\ par glisser-déposer.

Sophie peut alors fusionner deux nœuds participants, par glisser-déposer également. Par exemple, le *démêleur* de /démêler\ et le *utilisateur* de /peigne\ représentent le même participant (cf., figure 5.3).

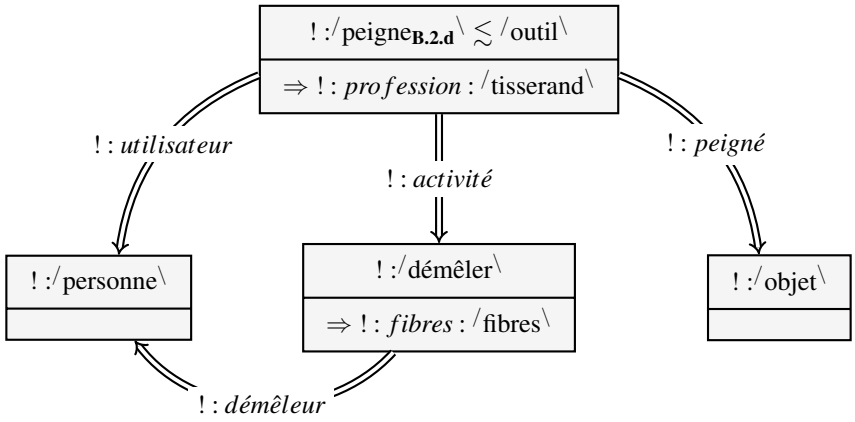


FIGURE 5.3 – Définition lexicographique formelle de /peigne$_{\mathbf{B.2.d}}$\ : fusion des nœuds de participant par glisser-déposer.

Ensuite, le *peigné* de /peigne$_{\textbf{B.2.d}}$\ et les *fibres* de /démêler\ doivent être liés par une relation de méronymie. Nous supposons qu'il existe un TUSemP /partieDe\ qui porte ce sens. Sophie clique donc sur un bouton "ajouter un nœud de participant", et cherche /partieDe\ dans la hiérarchie des TUSemP. Un nœud de participant typé /partieDe\ est alors ajouté (cf., figure 5.4).

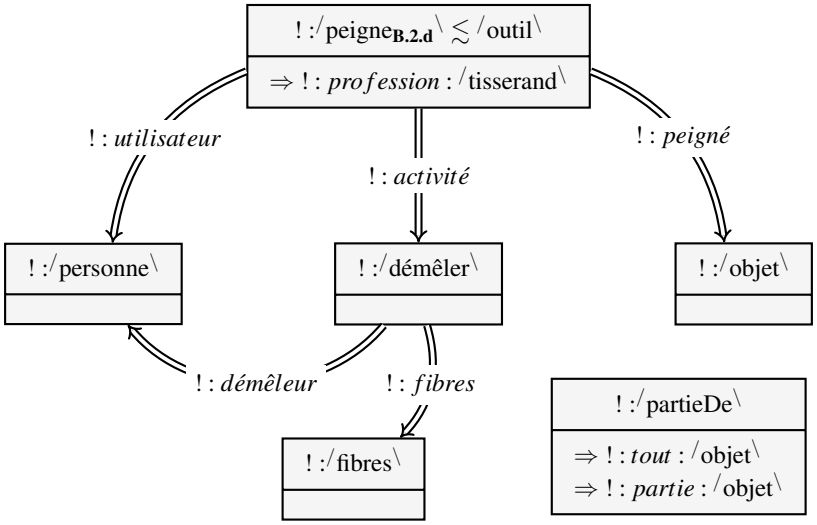


FIGURE 5.4 – Définition lexicographique formelle de /peigne$_{\textbf{B.2.d}}$\ : ajout d'un nœud de participant.

Sophie glisse la PosA *tout* de /partieDe\ et la dépose sur le *peigné* de /peigne$_{\textbf{B.2.d}}$\, puis glisse la *partie* et la dépose sur les *fibres* de /démêler\. Le résultat de cette tâche est illustré par la figure 5.5.

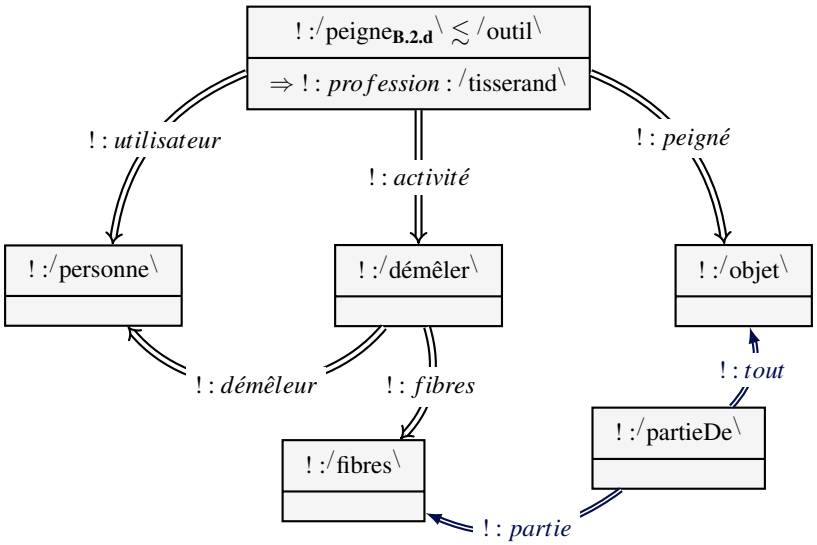


FIGURE 5.5 – Définition lexicographique formelle de /peigne$_{\textbf{B.2.d}}$\ : résultat final.

5.2.3 Définition de la structure actancielle au niveau sémantique de surface

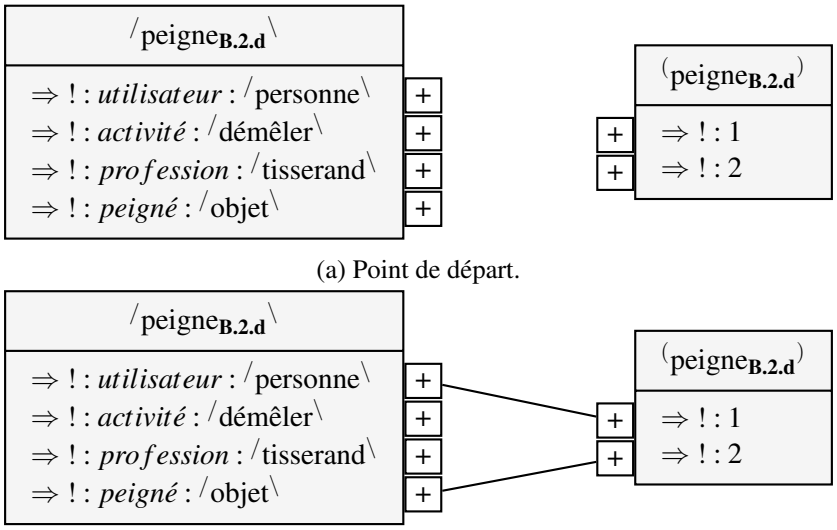
La troisième tâche, l'édition de la structure actancielle du TUSemS $^{(}\text{peigne}_{\textbf{B.2.d}}^{)}$, existe déjà dans le processus du projet RELIEF. En effet, selon notre conceptualisation étendue, la structure actancielle du TUSemS correspond à la structure actancielle sémantique du type de lexie que l'on décrit. Cette tâche correspond donc à la seconde tâche du processus du projet RELIEF.

5.2.4 Mise en correspondance des structures actancielles aux niveaux sémantique de surface et sémantique profond

La dernière tâche consiste à spécifier la structure actancielle de $^{(}\text{peigne}_{\textbf{B.2.d}}^{)}$, puis à mettre en correspondance les structures actancielles du TUSemP $^{/}\text{peigne}_{\textbf{B.2.d}}^{\backslash}$ et du TUSemS $^{(}\text{peigne}_{\textbf{B.2.d}}^{)}$. Dans une fenêtre dédiée, Sophie visualise :

- à gauche, une boîte pour $^{/}\text{peigne}_{\textbf{B.2.d}}^{\backslash}$ et sa structure actancielle ;
- à droite, une boîte pour $^{(}\text{peigne}_{\textbf{B.2.d}}^{)}$ et sa structure actancielle.

Un bouton est situé en face de chacune des PosA, comme illustré par la figure 5.6a. Sophie crée une correspondance en liant ces boutons par glisser-déposer. Le résultat de cette tâche est valide si le critère de correspondance simple défini dans la section 3.3.3.2 est satisfait.



(a) Point de départ.

(b) Mise en correspondance complète.

FIGURE 5.6 – Mise en correspondance des structures actancielles de $^{(}\text{peigne}_{\textbf{B.2.d}}^{)}$ et $^{/}\text{peigne}_{\textbf{B.2.d}}^{\backslash}$.

5.3 Maquettage et prototypage d'un éditeur de définitions lexicographiques formelles

Cette section décrit le résultat du stage orienté IHM de Romain GUGERT, qui avait pour objectif le développement et la validation d'un prototype d'éditeur de définitions lexicographiques formelles pour le projet RELIEF, selon le nouveau processus proposé.

Le développement du prototype avait été précédé par l'élaboration d'une maquette. Nous nous concentrerons ici sur le prototype et son évaluation. Néanmoins, les fonctionnalités proposées dans la maquette pourront être implémentées dans une version ultérieure de ce prototype.

Une vidéo de démonstration du prototype est accessible sur le site de l'équipe Wimmics à l'adresse http://wimmics.inria.fr/doc/video/UnitGraphs/editor1.html. Nous avons présenté cette démonstration lors de la 3e journée RELIEF [1], et lors des 25es Journées francophones d'Ingénierie des Connaissances.

5.3.1 Aperçu de l'éditeur

L'éditeur a été conçu comme une implémentation du processus présenté à la section 5.2, et spécifiquement autour de l'exemple de $\lceil \text{PEIGNE}_{\mathbf{B.2.D}} \rceil$.

Les images 5.7 à 5.10 représentent l'écran pour les quatre tâches principales. L'interface comprend un fil d'Ariane sur la gauche pour représenter le processus que nous proposons :

1. spécification de la structure actancielle du TUSemP ;
2. élaboration de la définition lexicographique formelle ;
3. spécification de la structure actancielle du TUSemS ;
4. mise en correspondance des structures actancielles.

Pour l'utilisateur, les opérations d'édition de la définition sont des opérations de manipulation de graphes. Le "glisser-déposer" lui permet de sortir une position actancielle, pour éventuellement se rendre compte que sa signature comporte elle-même des positions actancielles, ou pour fusionner des nœuds. Nous nous sommes inspiré d'UML, mais nous ne connaissons pas d'éditeur d'UML qui permette de "sortir" un attribut à l'extérieur de sa classe pour ainsi obtenir une association vers une autre classe. Techniquement, ces opérations sont implémentées en JavaScript au dessus de *mxGraph* [2], qui permet la visualisation et la manipulation de graphes.

1. http://www.atilf.fr/spip.php?article3895
2. mxGraph - visualisation de graphes en JavaScript - http://www.jgraph.com/mxgraph.html

FIGURE 5.7 – Éditeur de définitions lexicographiques formelles : étape 1/4, spécification de la structure actancielle du TUSemP.

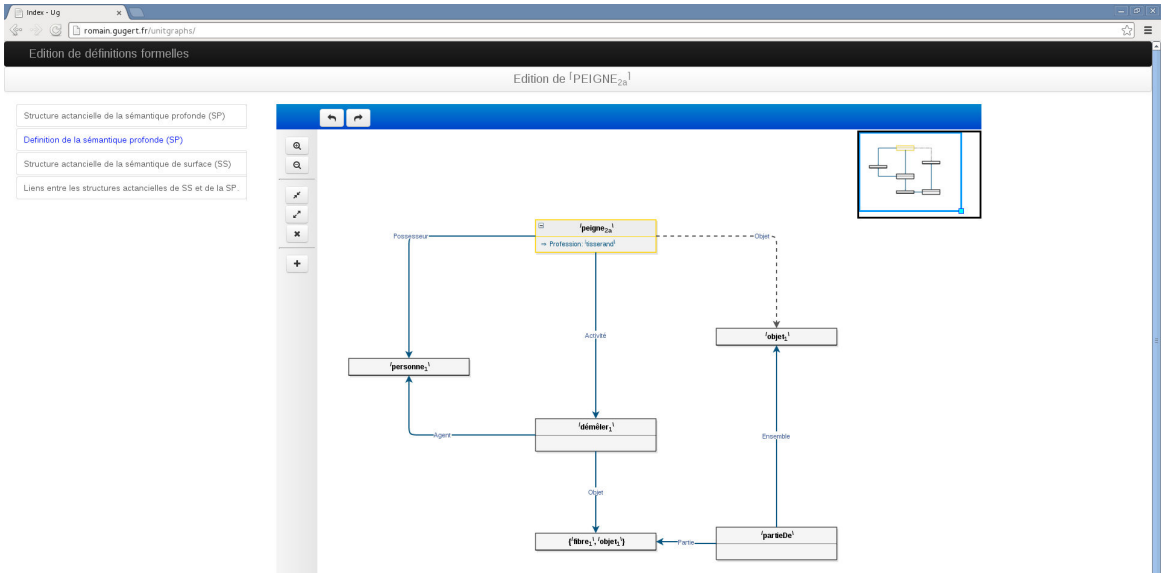


FIGURE 5.8 – Éditeur de définitions lexicographiques formelles : étape 2/4, élaboration de la définition lexicographique formelle par manipulation de graphe.

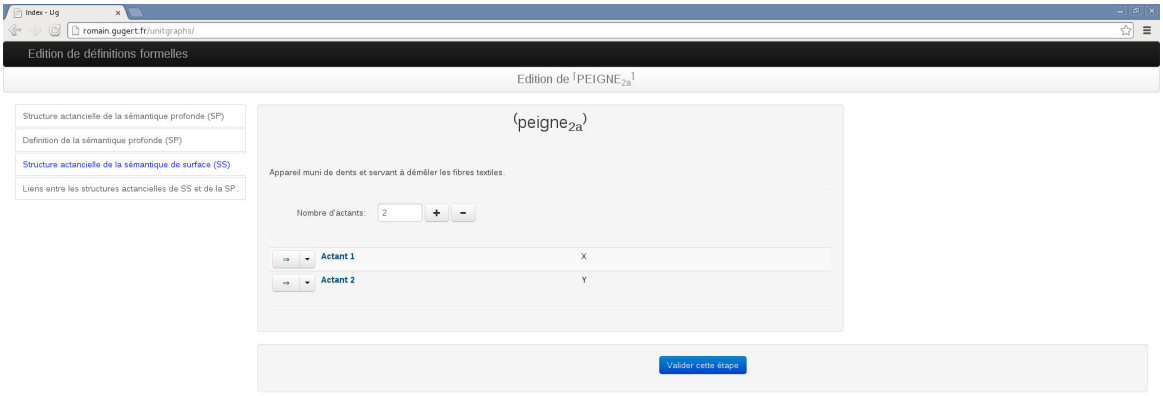


FIGURE 5.9 – Éditeur de définitions lexicographiques formelles : étape 3/4, spécification de la structure actancielle du TUSemS.

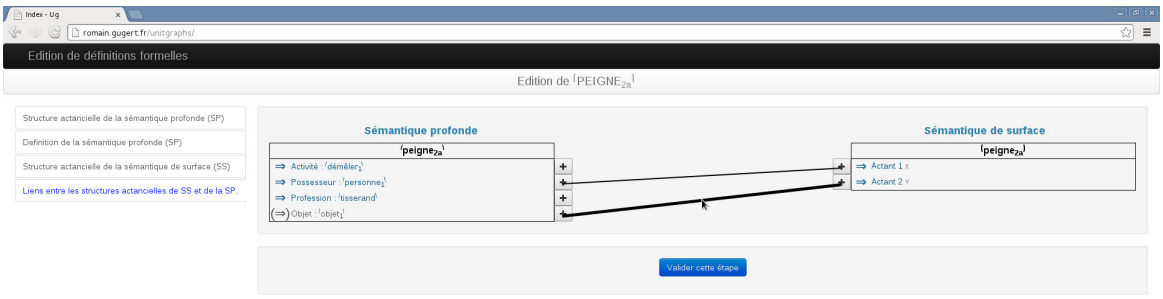


FIGURE 5.10 – Éditeur de définitions lexicographiques formelles : étape 4/4, mise en correspondance des structures actancielles.

5.3.2 Évaluation auprès des lexicographes du projet RELIEF

Nous avons présenté notre prototype d'éditeur aux lexicographes du projet RELIEF à l'occasion de la 3e journée RELIEF. Nous avons ensuite mené une évaluation sur 6 lexicographes en nous basant sur une démarche classique en IHM : l'évaluation coopérative (Monk *et al.*, 1993). Il s'agit de demander au lexicographe d'effectuer une tâche précise : définir formellement $\ulcorner \text{PEIGNE}_{\mathbf{B.2.D}} \urcorner$. Pendant toute l'évalution, le lexicographe nous fait part de ses remarques, interrogations, ou difficultés.

Cette évaluation de ce premier prototype a été décrite par Gugert (2013). Elle confirme les attentes des lexicographes pour un outil pratique leur permettant :

- de faciliter la visualisation des définitions lexicographiques et des représentations linguistiques ;
- de faciliter l'élaboration des définitions lexicographiques en cohérence avec les autres définitions lexicographiques dans le dictionnaire ;
- d'effectuer des requêtes complexes sur le dictionnaire ;
- de raisonner avec les définitions lexicographiques.

Le travail déjà effectué forme donc une base solide sur laquelle les futures discussions peuvent s'appuyer, et permettre le développement d'une nouvelle version de l'éditeur.

Notons également que la présentation des définitions lexicographiques formelles et du processus lors de la conférence internationale sur la Théorie Sens-Texte a suscité l'intérêt d'un groupe de lexicographes de l'Académie des Sciences de Moscou.

Un certain nombre d'améliorations de ce prototype est cependant nécessaire. Nous avons identifié trois directions principales pour cela.

Workflow pour les lexicographes. Nous devrions rendre le processus plus semblable à celui qui est utilisé actuellement. Par exemple, la définition de la structure actancielle sémantique du type de lexie devrait avoir lieu avant toute autre chose. Il s'agit en effet de l'étape sur laquelle chaque testeur s'est précipité, par habitude, ce qui a mené à une grande confusion avec la notion de structure actancielle du TUSemP.

Manipulation de graphes. La manipulation de graphes telle qu'elle a été conçue n'était pas suffisamment intuitive pour les lexicographes, qui préféraient ne rien essayer plutôt que de faire une bêtise. Nous constatons en particulier qu'ils n'avaient pas l'habitude de penser en termes de graphe : ces représentations abondent dans la littérature, mais sont peu utilisées sur le terrain. Nous devrions en particulier travailler sur l'aide à la prise en main des représentations sous la forme de graphe.

Niveau sémantique profond. Enfin, nous devrions masquer au mieux l'utilisation du niveau sémantique profond, inconnu et perturbant pour les lexicographes.

Conclusion

Nous avons donc étudié la tâche d'édition de définitions lexicographiques formelles dans le projet RELIEF de lexicologie explicative et combinatoire, et nous avons proposé un nouveau processus adapté à la conceptualisation étendue que nous avons développée. Ce processus a été implémenté dans un prototype d'éditeur, et a été évalué auprès des lexicographes du projet RELIEF.

Ce travail représente une première étape importante en vue de l'intégration future de nos travaux dans des logiciels de lexicographie. En particulier, l'évaluation par les lexicographes de la TST nous a permis de confirmer leurs attentes pour un tel outil pratique, et d'identifier trois directions d'amélioration pour notre prototype :

- rendre le processus plus semblable à ce qu'il est aujourd'hui ;
- rendre la manipulation de graphe plus intuitive ;
- masquer si possible l'existence du niveau sémantique profond.

Le travail que nous avons présenté offre donc une première base essentielle sur laquelle de futures discussions peuvent s'appuyer, et il permettra, nous l'espérons, le développement agile d'une version améliorée de notre prototype.

Troisième partie

Formalisation des prédicats linguistiques et des définitions lexicographiques

CHAPITRE 6

Inadéquation des logiques de description et des Graphes Conceptuels

---

*_ **I**nnover, ce n'est pas avoir une nouvelle idée mais arrêter d'avoir une vieille idée.*
Edwin Herbert Land

Sommaire

Introduction

Maintenant que nous avons étendu la conceptualisation des prédicats linguistiques et des définitions lexicographiques, nous cherchons un formalisme de représentation des connaissances adéquat. Nous étudions dans ce chapitre deux formalismes candidats, les logiques de description et les Graphes Conceptuels, comme nous l'avons justifié dans la section 2.2.2.2. Nous en évaluerons l'adéquation au regard des critères de choix formulés dans la section 2.2.2.1.

Nous avons remarqué dans la section 1.3.1.2 quelques différences entre les prédicats linguistiques et les prédicats logiques.

– Une Position Actancielle (PosA) peut être optionnelle ou obligatoire ;
– Une PosA, même obligatoire, n'est pas obligatoirement exprimée dans les textes.

Nous souhaitons qu'un modèle logique des prédicats linguistiques rende compte de ces possibilités.

Par ailleurs, pour un type de lexie $\ulcorner \mathrm{B} \urcorner$ de genre proche $\ulcorner \mathrm{A} \urcorner$, nous souhaitons qu'un modèle logique de sa définition lexicographique formelle $D(/\mathrm{B}\backslash)$ permette l'expansion et la contraction, vues comme des inférences logiques.

Expansion. Si une Instance d'Unité Sémantique Profonde (IUSemP) de type $/\mathrm{L}\backslash$ est rencontrée dans une RSémP avec une partie de sa structure actancielle, alors une instanciation de $D(/\mathrm{B}\backslash)$ peut être inférée et ajoutée à la représentation.

Contraction. Si une Représentation Sémantique Profonde (RSémP) contient une instance de $D(/\mathrm{B}\backslash)$, alors le Type d'Unité Sémantique Profonde (TUSemP) $/\mathrm{B}\backslash$ et une partie de sa structure actancielle peuvent être inférés et ajoutés à la représentation.

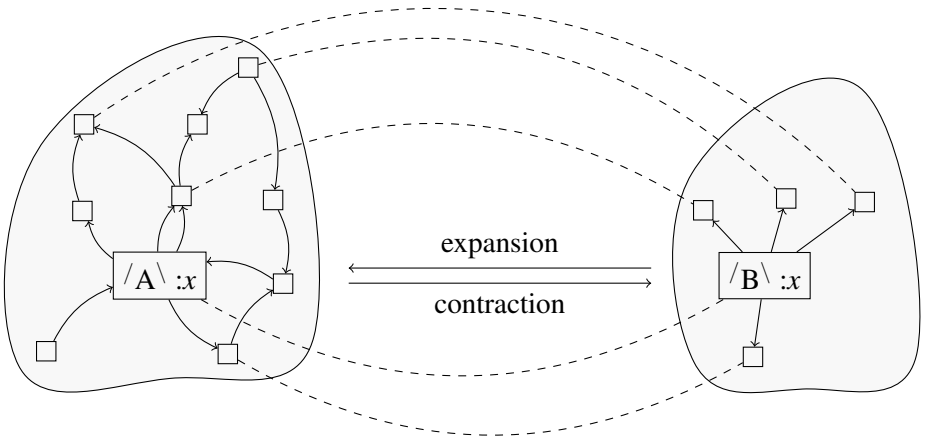


FIGURE 6.1 – Les règles de contraction et d'expansion associées à une définition lexicographique formelle $D(/\mathrm{B}\backslash)$, de genre proche $/\mathrm{A}\backslash$

6.1 Les logiques de description

Les logiques de description ont une base commune et différentes extensions [1]. Plus on choisit d'extensions, et plus le langage est expressif, et plus il est complexe. L'étude de la complexité des différents fragments des logiques de descriptions fait l'objet d'un grand nombre de travaux [2]. L'expressivité maximale est équivalente à la logique du premier ordre avec relations binaires, dans laquelle l'inférence de formules est indécidable. Nous devons donc limiter au maximum les extensions utilisées.

Nous proposerons une représentation des types d'unité linguistique (§6.1.1), puis nous étudierons la représentation de deux définitions lexicographiques simples (§6.1.2). Enfin, nous conclurons sur l'inadéquation des logiques de description pour notre étude (§6.1.3).

6.1.1 Représentation des types d'unité linguistique

Les logiques de description utilisent les notions de concept, de rôle, et d'individu. Naturellement, un type d'unité linguistique serait représenté par un concept, ses instances seraient des individus et seraient liées entre elles par des rôles dans les représentations linguistiques.

Les relations syntaxiques de surface et profonde, ainsi que les relations sémantiques de surface et profonde, seraient toutes représentées par des rôles.

6.1.1.1 Représentation de la structure actancielle

Au plus une unité linguistique peut remplir une PosA donnée. Les LD permettent de représenter cette contrainte à l'aide de l'axiome de rôle fonctionnel. Tout rôle qui représente une PosA doit donc être déclaré fonctionnel. Par exemple, l'équation suivante en syntaxe des LD déclare la relation sémantique profonde *eater* fonctionnelle.

$$\top \sqsubseteq (\leq 1.eater) \tag{6.1}$$

Il est possible d'associer à chaque rôle un domaine, qui identifie le type de toute unité qui gouverne d'autres unités linguistiques via ce rôle. Par exemple, l'équation suivante en syntaxe des LD déclare que le domaine de *eater* est $^{/}$to eat$^{\backslash}$.

$$(\geq 1.eater) \sqsubseteq {}^{/}\text{to eat}^{\backslash} \tag{6.2}$$

Nous pouvons ensuite représenter la nature obligatoire d'une PosA à l'aide de l'axiome de cardinalité minimale de 1, et la nature interdite d'une PosA à l'aide de l'axiome de cardinalité maximale de 0. Par exemple, les équations suivantes en syntaxe des LD déclarent les PosA *eater* et *container* de $^{/}$to graze$^{\backslash}$ comme respectivement obligatoire et interdite.

$$^{/}\text{to graze}^{\backslash} \sqsubseteq (\geq 1.eater) \tag{6.3}$$

$$^{/}\text{to graze}^{\backslash} \sqsubseteq (\leq 0.container) \tag{6.4}$$

1. Les logiques de description, http://fr.wikipedia.org/wiki/Logique_de_description
2. Le site http://www.cs.man.ac.uk/~ezolin/dl/ tient à jour les résultats de complexité pour les différents fragments des logiques de description.

La signature d'un TUSemP pour une PosA peut être représentée à l'aide d'une restriction universelle. Par exemple, l'équation suivante en syntaxe des LD déclare que la signature de *eaten* pour $^{/}$to graze$^{\backslash}$ est $^{/}$vegetal$^{\backslash}$.

$$^{/}\text{to graze}^{\backslash} \sqsubseteq (\forall \textit{eaten}.{}^{/}\text{vegetal}^{\backslash}) \tag{6.5}$$

Nous pouvons donc ainsi représenter le type d'unité linguistique qui introduit une PosA, ainsi que la nature et la signature de chaque PosA d'un type d'unité linguistique. Évaluons maintenant l'adéquation de la représentation des hiérarchies des types d'unité sémantique de surface et profonde ainsi obtenue.

6.1.1.2 Hiérarchie des types d'unité sémantique de surface

La figure 3.5 de la section 3.2.2 illustrait l'organisation hiérarchique des TUSemS dirigée par leur structure actancielle. L'ensemble d'axiomes de logiques de description suivant représente cette hiérarchie.

$$\top \sqsubseteq (\leq 1.\ 1) \tag{6.6}$$
$$(\geq 1.\ 1) \sqsubseteq {}^{(}?^{)} \tag{6.7}$$
$${}^{(}??^{)} \sqsubseteq {}^{(}?^{)} \tag{6.8}$$
$$\top \sqsubseteq (\leq 1.\ 2) \tag{6.9}$$
$$(\geq 1.\ 2) \sqsubseteq {}^{(}??^{)} \tag{6.10}$$
$${}^{(}???^{)} \sqsubseteq {}^{(}??^{)} \tag{6.11}$$
$$\top \sqsubseteq (\leq 1.\ 3) \tag{6.12}$$
$$(\geq 1.\ 3) \sqsubseteq {}^{(}???^{)} \tag{6.13}$$
$${}^{(}????^{)} \sqsubseteq {}^{(}???^{)} \tag{6.14}$$
$$\top \sqsubseteq (\leq 1.\ 4) \tag{6.15}$$
$$(\geq 1.\ 4) \sqsubseteq {}^{(}????^{)} \tag{6.16}$$
$${}^{(}!^{)} \sqsubseteq {}^{(}?^{)} \tag{6.17}$$
$${}^{(}!^{)} \sqsubseteq (\geq 1.\ 1) \tag{6.18}$$
$${}^{(}!?^{)} \sqsubseteq {}^{(}!^{)} \tag{6.19}$$
$${}^{(}!?^{)} \sqsubseteq {}^{(}??^{)} \tag{6.20}$$
$${}^{(}!!^{)} \sqsubseteq {}^{(}!?^{)} \tag{6.21}$$
$${}^{(}!!^{)} \sqsubseteq (\geq 1.\ 2) \tag{6.22}$$
$${}^{(}!??^{)} \sqsubseteq {}^{(}!?^{)} \tag{6.23}$$
$${}^{(}!??^{)} \sqsubseteq {}^{(}???^{)} \tag{6.24}$$
$${}^{(}!!?^{)} \sqsubseteq {}^{(}!!^{)} \tag{6.25}$$
$${}^{(}!!?^{)} \sqsubseteq {}^{(}!??^{)} \tag{6.26}$$
$${}^{(}\text{manger}^{)} \sqsubseteq {}^{(}!!^{)} \tag{6.27}$$

$$^{(}\text{to eat}^{)} \sqsubseteq {}^{(}!!?^{)} \tag{6.28}$$

Un léger problème d'encodage ici est qu'il n'est pas directement possible de répondre à la question : *le type d'unité sémantique* $^{(}$*to eat*$^{)}$ *a-t-il une PosA 3 ?*. En effet, tout ce qui est déductible par rapport au rôle 3 est que : si une unité linguistique gouverne une autre unité linguistique via à la PosA 3, alors elle est du type $^{(}???^{)}$. Il n'est donc pas directement possible de connaître le nombre de PosA total d'une unité sémantique.

Ainsi, la représentation de la hiérarchie des types d'unité sémantique de surface est claire et cohérente, mais possède une légère déformation d'encodage.

6.1.1.3 Hiérarchie des types d'unité sémantique profonde

Nous avons montré comment il est possible d'associer une structure actancielle aux TUSemP.

Nous pouvons maintenant montrer que le principe d'héritage et de spécialisation de la structure actancielle des types d'unité sémantique profonde (cf., définition 17 p.65), et le principe d'héritage et de spécialisation d'une PosA (définition 19 p.70) est bien représenté :

- un TUSemP hérite des PosA de ses parents, et peut en introduire de nouvelles ;
- une PosA optionnelle peut devenir obligatoire, interdite, mais pas les deux à la fois, comme l'illustre la figure 6.2 ;
- la signature d'une PosA ne peut être que spécialisée.

La figure 6.2 illustre le squelette de la hiérarchie des TUSemP qui concerne la PosA *eater*. Après avoir été introduite par un concept anonyme $(\leq 1.eater)$, la PosA peut devenir obligatoire ou interdite. Par contre, avoir une PosA qui serait à la fois obligatoire et interdite serait absurde.

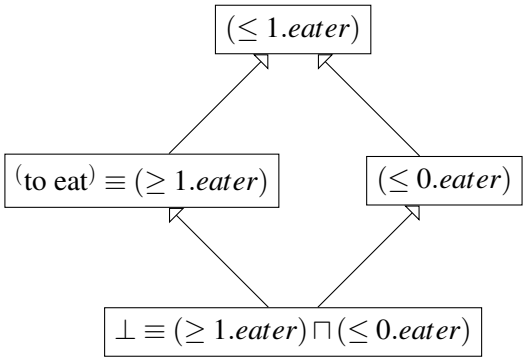


FIGURE 6.2 – Illustration de la direction de la hiérarchie des TUSemP par la nature des PosA.

Ce qui est un peu plus problématique, comme pour la représentation de la hiérarchie des TUSemS, est de déterminer les PosA optionnelles d'un TUSemP.

Voyons maintenant s'il est possible de représenter des définitions lexicographiques à l'aide des logiques de description.

6.1.2 Tentative de représentation des définitions lexicographiques formelles

Nous souhaitons donc représenter la définition lexicographique formelle d'un type de lexie $^{/}\mathrm{L}^{\backslash}$ de sorte que les opérations de contraction et d'expansion décrites dans la section 6 soient possibles.

La section 6.1.2.1 pose le problème de la représentation des définitions lexicographiques formelles à l'aide des logiques de description. Nous étudierons ensuite deux cas particuliers dans les sections 6.1.2.2 et 6.1.2.3.

6.1.2.1 Expression du problème

Les règles de contraction et d'expansion peuvent être écrites en logique du premier ordre sous la forme suivante :

Définition 23 (Règle de contraction)**.** Une règle de contraction de $^{/}\mathrm{B}^{\backslash}$ est une expression en logique du premier ordre de la forme :

$$(\forall x, y_1, \ldots, y_n)\Big[(\exists z_1, \ldots, z_m)[E(x, y_1, \ldots, y_n, z_1, \ldots, z_m)] \implies C(x, y_1, \ldots, y_n)\Big] \tag{6.29}$$

où :
- $E(x, y_1, \ldots, y_n, z_1, \ldots, z_m)$ est une expression du *graphe expansé* de $^{/}\mathrm{B}^{\backslash}$;
- $C(x, y_1, \ldots, y_n)$ est une expression du *graphe contracté* de $^{/}\mathrm{B}^{\backslash}$;
- x représente l'instance de $^{/}\mathrm{B}^{\backslash}$, il s'agit de la variable centrale de la règle ;
- $y1, \ldots, y_n$ sont les variables qui représentent les PosAs exprimées de x ;
- $z1, \ldots, z_m$ sont d'autres variables impliquées dans le graphe expansé de $^{/}\mathrm{B}^{\backslash}$.

Définition 24 (Règle d'expansion)**.** Une règle d'expansion de $^{/}\mathrm{B}^{\backslash}$ est une expression en logique du premier ordre de la forme :

$$(\forall x, y_1, \ldots, y_n)\Big[C(x, y_1, \ldots, y_n) \implies (\exists z_1, \ldots, z_m)[E(x, y_1, \ldots, y_n, z_1, \ldots, z_m)]\Big] \tag{6.30}$$

où :
- $C(x, y_1, \ldots, y_n)$ est une expression du *graphe contracté* de $^{/}\mathrm{B}^{\backslash}$;
- $E(x, y_1, \ldots, y_n, z_1, \ldots, z_m)$ est une expression du *graphe expansé* de $^{/}\mathrm{B}^{\backslash}$;
- x représente l'instance de $^{/}\mathrm{B}^{\backslash}$, il s'agit de la variable centrale de la règle ;
- $y1, \ldots, y_n$ sont les variables qui représentent les PosA exprimées de x ;
- $z1, \ldots, z_m$ sont d'autres variables impliquées dans le graphe expansé de $^{/}\mathrm{B}^{\backslash}$.

Comme dans des travaux préliminaires (Lefrançois et Gandon, 2011a), nous proposons de projeter la définition d'un TUSemP sur son type.

Définition 25 (Projection d'une définition formelle sur un type)**.** La projection d'une définition lexicographique formelle sur un type est un ensemble d'axiomes de Logique de Description qui contient au moins deux axiomes de la forme :

$$F(A) \sqsubseteq G(B) \tag{6.31}$$
$$G'(B) \sqsubseteq F'(A) \tag{6.32}$$

Où F, F', G, G', sont des expressions de classes.

En logique du premier ordre, ces règles ont les expressions suivantes :

$$(\forall x)[F(A)(x) \implies G(B)(x)] \tag{6.33}$$
$$(\forall x)[G'(B)(x) \implies F'(A)(x)] \tag{6.34}$$

où $(\forall x)[F(A)(x)]$, $(\forall x)[F'(A)(x)]$, $(\forall x)[G(B)(x)]$, et $(\forall x)[B'(B)(x)]$, sont les expressions en logique du premier ordre équivalentes aux expressions de classes $F(A)$, $F'(A)$, $G(B)$, et $G'(B)$ respectivement.

Une série de questions se pose alors pour notre thèse :
– Comment obtenir l'expression de la projection d'une définition lexicographique formelle sur le type défini ?
– Une telle expression existe-t-elle toujours ?
– Est-elle satisfaisante selon les critères de Gruber énoncés dans la section 2.2.2.1 ?
Les sections suivantes étudient deux cas particuliers de définition lexicographique formelle.

6.1.2.2 Cas d'étude 1 : la PosA d'une PosA est une PosA

Considérons la définition lexicographique formelle $D(/\mathrm{B}\backslash)$ illustrée par la figure 6.3. Le genre proche de $/\mathrm{B}\backslash$ est $/\mathrm{A}\backslash$, et un nœud participant correspond à la fois à la PosA obligatoire g, et à la PosA obligatoire f_2 de la PosA obligatoire f_1.

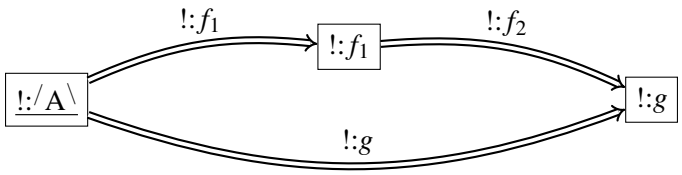


FIGURE 6.3 – Cas d'étude 1 : définition lexicographique formelle de $/\mathrm{B}\backslash$.

Nous savons au moins que les rôles f_1, f_2, et g sont fonctionnels :

$$\top \sqsubseteq (\leq 1.f_1) \tag{6.35}$$
$$\top \sqsubseteq (\leq 1.f_2) \tag{6.36}$$
$$\top \sqsubseteq (\leq 1.g) \tag{6.37}$$

Modélisation de la règle de contraction. La règle de contraction de $/\mathrm{B}\backslash$ est illustrée par la figure 6.4, et possède l'expression suivante en logique du premier ordre :

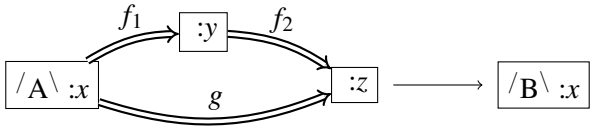


FIGURE 6.4 – Règle de contraction de $/\mathrm{B}\backslash$.

$$(\forall x)\Big[(\exists y,z)\big[A(x)\wedge f_1(x,y)\wedge f_2(y,z)\wedge g(x,z)\big] \implies B(x)\Big] \tag{6.38}$$

Introduisons un concept auxiliaire R.

Proposition 6.1.1. *Sachant que les rôles f_1, f_2, et g sont fonctionnels, l'axiome de LD suivant implique la règle de contraction 6.38.*

$$A\sqcap\neg B\sqsubseteq(\forall g.R)\sqcap\big(\forall f_1.(\forall f_2.\neg R)\big) \tag{6.39}$$

Preuve: voir annexe, p. 301.

Modélisation de la règle d'expansion. La règle d'expansion de $/\mathrm{B}\backslash$ est la combinaison de deux expressions en logique du premier ordre, illustrées sur la figure 6.5, selon qu'une PosA est exprimée ou non :

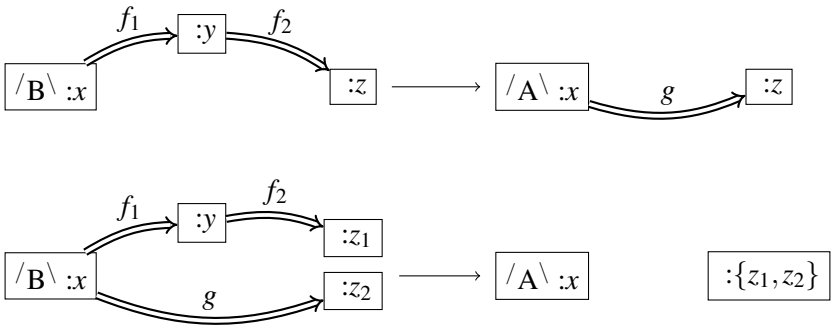


FIGURE 6.5 – Règles d'expansion de $/\mathrm{B}\backslash$.

$$(\forall x,y,z)\big[B(x)\wedge f_1(x,y)\wedge f_2(y,z)\implies A(x)\wedge g(x,y)\big] \tag{6.40}$$
$$(\forall x,y,z_1,z_2)\big[B(x)\wedge f_1(x,y)\wedge f_2(y,z_1)\wedge g(x,z_2)\implies A(x)\wedge(z_1=z_2)\big] \tag{6.41}$$
$$\tag{6.42}$$

Introduisons un rôle auxiliaire h, tel que les expressions suivantes soient vraies.

$$(\forall x,y,z)\,[f_1(x,y)\wedge f_2(y,z)\implies h(x,z)] \tag{6.43}$$
$$(\forall x,y)\,[g(x,z)\implies h(x,z)] \tag{6.44}$$

Ces expressions sont équivalentes aux axiomes suivants en LD :

$$f_1\circ f_2\sqsubseteq h \tag{6.45}$$
$$g\sqsubseteq h \tag{6.46}$$

Proposition 6.1.2. *Sachant que les rôles f_1, f_2, et g sont fonctionnels, et ayant posé les axiomes 6.45 et 6.46, l'axiome suivant implique les règles d'expansion 6.40 et 6.41 :*

$$B \sqsubseteq A \sqcap (\leq 1.h) \tag{6.47}$$

Preuve: voir annexe, p. 301.

Cette expression en LD est problématique puisqu'elle appartient à un fragment indécidable des logiques de description. En effet, la hiérarchie des rôles doit être régulière (cf. Rudolph, 2011, §2.1) afin de garantir la décidabilité des problèmes de raisonnement classiques. Ici, l'axiome 6.45 implique que h est un rôle non simple, et un rôle non simple ne doit pas être utilisé dans des restrictions de cardinalité, telles que $(\leq 1.h)$.

6.1.2.3 Cas d'étude 2 : Le nœud participant central est une PosA de l'une de ses PosA

Considérons la définition lexicographique formelle $D({}^{/}\mathrm{B}^{\backslash})$ illustrée par la figure 6.6. Le genre proche de ${}^{/}\mathrm{B}^{\backslash}$ est ${}^{/}\mathrm{A}^{\backslash}$, et le nœud participant central correspond à la PosA obligatoire f_2 de sa PosA obligatoire f_1.

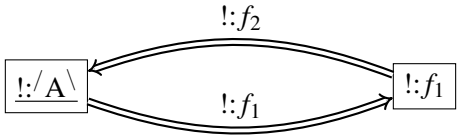


FIGURE 6.6 – Cas d'étude 2 : Définition lexicographique formelle de ${}^{/}\mathrm{B}^{\backslash}$.

Nous savons au moins que les rôles f_1, et f_2 sont fonctionnels :

$$\top \sqsubseteq (\leq 1.f_1) \tag{6.48}$$
$$\top \sqsubseteq (\leq 1.f_2) \tag{6.49}$$

Modélisation de la règle de contraction. La règle de contraction de ${}^{/}\mathrm{B}^{\backslash}$ est illustrée par la figure 6.7, et a l'expression suivante :

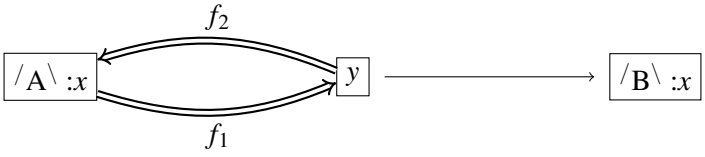


FIGURE 6.7 – Règle de contraction de ${}^{/}\mathrm{B}^{\backslash}$.

$$(\forall x)\,[(\exists y)\,[A(x) \wedge f_1(x,y) \wedge f_2(y,x)] \implies B(x)] \tag{6.50}$$

Introduisons un concept auxiliaire R.

Proposition 6.1.3. *Sachant que les rôles f_1 et f_2 sont fonctionnels, l'axiome suivant implique la règle de contraction 6.50 :*

$$A \sqcap \neg B \sqsubseteq R \sqcap \left(\forall f_1.(\forall f_2.\neg R)\right) \tag{6.51}$$

Preuve: voir annexe, p. 302.

Modélisation de la règle d'expansion. La règle d'expansion de $/\mathrm{B}\backslash$ est la combinaison de deux expressions en logique du premier ordre, illustrées sur la figure 6.8, selon que les nœuds de participant se correspondent explicitement ou non :

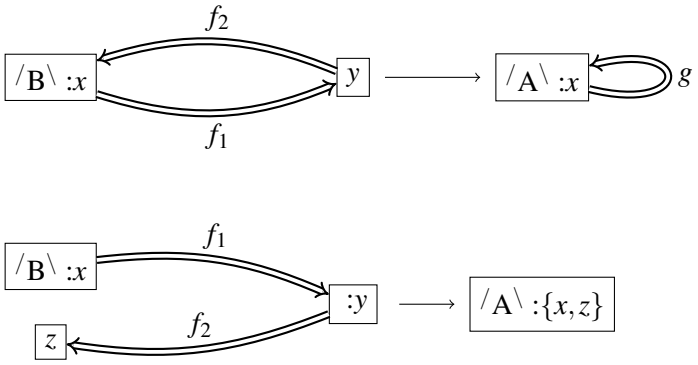


FIGURE 6.8 – Règles d'expansion de $/\mathrm{B}\backslash$.

$$(\forall x,y,z)\,[B(x) \wedge f_1(x,y) \wedge f_2(y,z) \implies A(x) \wedge g(x,x)] \tag{6.52}$$
$$(\forall x,y,z)\,[B(x) \wedge f_1(x,y) \wedge f_2(y,z_1) \implies A(x) \wedge (x = z)] \tag{6.53}$$

Introduisons une relation auxiliaire g, telle que les expressions suivantes soient vraies :

$$(\forall x,y,z)[f_1(x,y) \wedge f_2(y,z) \implies g(x,z)] \tag{6.54}$$
$$(\forall x)[B(x) \implies g(x,x)] \tag{6.55}$$

Ces expressions sont équivalentes aux axiomes suivants en LD :

$$f_1 \circ f_2 \sqsubseteq g \tag{6.56}$$
$$B \sqsubseteq \mathbf{hasSelf}.g \tag{6.57}$$

Proposition 6.1.4. *Sachant que les rôles f_1 et f_2 sont fonctionnels, et ayant posé les axiomes 6.56 et 6.57, l'axiome suivant implique les règles d'expansion 6.52 et 6.53 :*

$$B \sqsubseteq A \sqcap (\leq 1.g) \tag{6.58}$$

Preuve: voir annexe, p. 303.

À nouveau, cette expression en LD est problématique puisqu'elle rend la hiérarchie des rôles non régulière, et donc les raisonnements indécidables. En effet, l'axiome 6.56 implique que g est un rôle non simple, et un rôle non simple ne doit pas être utilisé dans des restrictions de cardinalité, telles que $(\leq 1.g)$.

6.1.3 Conclusion sur l'inadéquation des logiques de description pour notre étude

Pour ces deux cas d'étude, nous avons montré comment projeter la définition lexicographique formelle sur le TUSemP. Nous ne prouverons pas que toute définition lexicographique peut effectivement être projetée sur un concept, mais nous identifions déjà trois problèmes fondamentaux sur ces cas d'étude simples :

- Les règles d'expansion des cas 1 et 2 ne sont pas dans un fragment décidable des logiques de description, et le raisonnement est rendu indécidable. Le critère de minimalité de l'engagement ontologique n'est donc pas satisfait.
- Malgré la simplicité de ces cas d'étude, nous requérons un grand nombre d'axiomes, ainsi que l'introduction de relations et/ou concepts auxiliaires. On peut donc affirmer que les critères de clarté et de minimalité de la déformation d'encodage ne sont pas satisfaits par cette solution.
- Nous n'avons traité que deux cas simples, avec des participants obligatoires. Si certains participants étaient optionnels, plus de règles de contraction ou d'expansion possibles devraient être modélisées. Cela ferait exploser le nombre d'axiomes nécessaires, et limiterait d'autant la clarté de la représentation.

Nous avons ainsi montré que la capacité de modélisation des logiques de description est insatisfaisante pour la représentation des définitions lexicographiques de la TST.

6.2 Graphes Conceptuels

Nous avons présenté dans la section 2.3.2 les similarités entre le formalisme des Graphes Conceptuels et la TST, et avons justifié l'étude de l'adéquation de ce formalisme pour représenter les définitions lexicographiques formelles. Cette section présente deux options de modélisation différentes.

- La première option (§6.2.1) consiste à représenter un prédicat linguistique :
  - par un type de concept, puisqu'il est instancié dans les représentations linguistiques ;
  - et par une relation n-aire, qui lie une instance à ses actants dans une représentation linguistique.
- La seconde option (§6.2.2) est similaire à la solution utilisée avec les logiques de description, et consiste à représenter chaque relation actancielle par une relation binaire.

6.2.1 Utilisation de relations n-aires

Chacun des types d'unité linguistique pourrait être modélisé par un couple ⟨type de concept, relation⟩. Cette approche semble intéressante puisqu'elle permet d'envisager la réutilisation des travaux de Leclère (1998) concernant la formalisation et l'opérationnalisation des définitions de types de concept et de relation.

La relation n-aire associée à un type d'unité linguistique devrait posséder un argument pour les instances associées, et autant d'arguments supplémentaires que le nombre de PosA prises en compte. Le formalisme des GC impose alors l'aspect fonctionnel des PosA, et nous pouvons modéliser la signature d'un type d'unité linguistique à l'aide du mécanisme existant de spécification de la signature d'une relation n-aire. Par exemple, la figure 6.9 illustre la modélisation du TUSemP ⟨démêler⟩ et de ses deux PosA obligatoires.

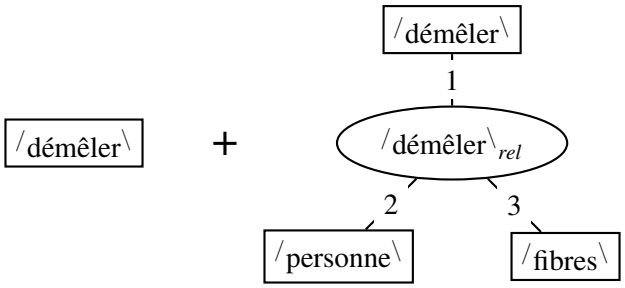


FIGURE 6.9 – Représentation du TUSemP ⟨démêler⟩ et de ses deux PosA obligatoires avec les Graphes Conceptuels, si l'on modélise chaque relation actancielle par un couple ⟨type de concept, relation⟩.

Un premier problème avec cette modélisation est que nous devons stocker à part la correspondance entre les PosA d'un type d'unité linguistique, et les arguments de la relation *n*-aire correspondante. Par exemple, les arguments 2 et 3 de la relation *n*-aire /démêler\$_{rel}$ correspondent respectivement aux PosA *démêleur* et *fibres* du TUSemP /démêler\. Les représentations linguistiques construites suivant ce choix de modélisation manquent alors de clarté, comme l'illustre la figure 6.10.

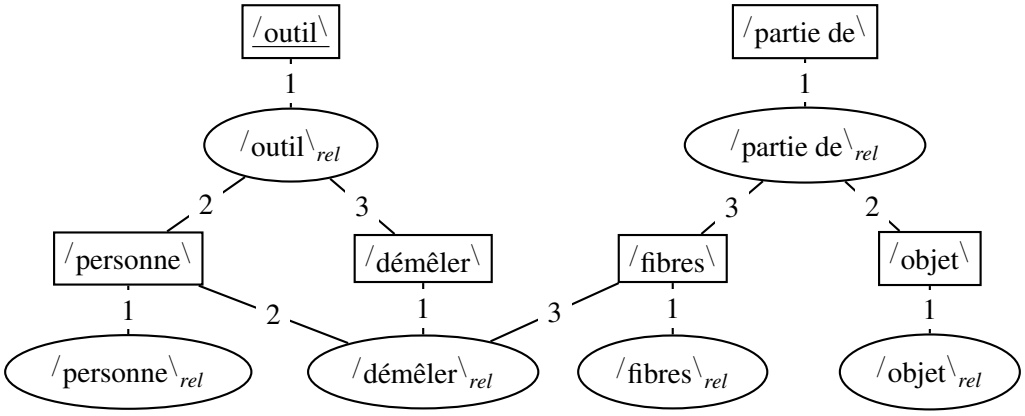


FIGURE 6.10 – Représentation avec les Graphes Conceptuels de l'instanciation de la définition lexicographique de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉, en modélisant chaque relation actancielle par un couple ⟨type de concept, relation⟩.

Par ailleurs, les PosA ne sont pas nécessairement exprimées dans les représentations linguistiques. Nous devrions donc manipuler des représentations linguistiques qui ne sont que des pseudo-graphes conceptuels (cf., Chein et Mugnier, 2009, p.30).

La modélisation des différentes natures de PosA ne peut pas se faire sans une déformation importante d'encodage. En effet, chaque argument d'une relation *n*-aire ne peut qu'être obligatoire. Afin de représenter une PosA optionnelle, il serait nécessaire d'introduire deux relations : une qui prend en compte la PosA, et l'autre non. Nous devrions alors stocker deux correspondances entre les PosA et les arguments des relations, et le nombre de telles relations croîtrait de manière exponentielle avec le nombre de positions optionnelles dans la structure actancielle. Enfin, il est impossible de représenter les PosA interdites.

Quand bien même nous ne considérerions que les PosA obligatoires, le formalisme des GC ne permettrait pas de hiérarchiser des relations qui présentent une arité différente. Nous devrions donc utiliser des règles d'inférence pour spécifier le lien entre les relations $/\text{peigne}_{\textbf{B.2.d}}\backslash_{rel}$ et $/\text{outil}\backslash_{rel}$ par exemple.

Le manque de clarté et la déformation d'encodage ainsi mis en évidence nous pousse à abandonner ce choix de modélisation. Étudions maintenant la possibilité de réifier les types d'unité linguistique comme avec les logiques de description.

6.2.2 Utilisation de relations binaires

Supposons que chacune des relations actancielles soit modélisée par une relation binaire dans le formalisme des GC. La figure 6.11 ci-dessous illustre une représentation linguistique avec ce choix de modélisation.

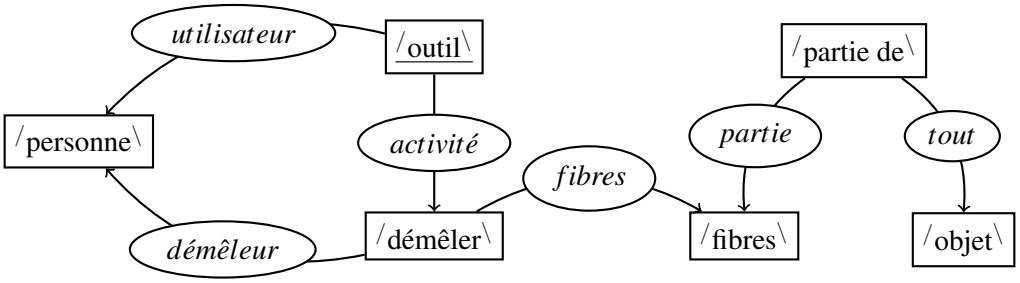


FIGURE 6.11 – Représentation avec les Graphes Conceptuels de l'instanciation de la définition lexicographique de $\lceil \text{PEIGNE}_{\textbf{B.2.D}} \rceil$, en modélisant chaque relation actancielle par une relation binaire.

Cette section étudie successivement la représentation des types d'unité linguistique (§6.2.2.1), puis celle des définitions lexicographiques (§6.2.2.2).

6.2.2.1 Modélisation des types d'unité linguistique

La règle d'inférence illustrée par la figure 6.12 modélise le fait que la relation *eater* est fonctionnelle. Deux choix équivalents de modélisation s'offrent alors à nous : nous pouvons associer une telle règle à chaque relation actancielle *s*, ou bien associer une seule règle à une relation binaire auxiliaire *actant*, et spécifier pour chaque relation actancielle *s* qu'elle est une sous-relation de *actant*. Nous ne trancherons pas ici.

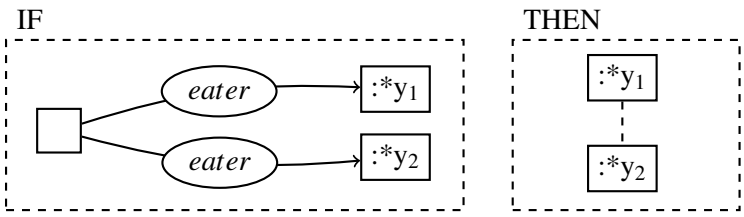


FIGURE 6.12 – Modélisation de l'aspect fonctionnel de la relation *eater* à l'aide d'une règle d'inférence des graphes conceptuels.

Le fait que la PosA *eater* est obligatoire pour le type d'unité linguistique /to eat\ est modélisé par une règle d'inférence, comme illustré par la figure 6.13. Nous choisissons une règle d'inférence

plutôt qu'une règle de contrainte positive, car nous savons qu'une PosA, même obligatoire, n'est pas nécessairement explicitée dans les représentations linguistiques.

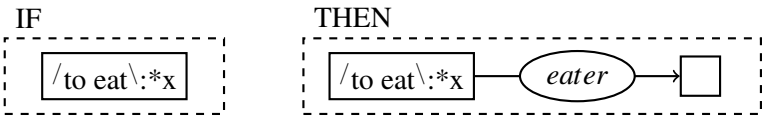


FIGURE 6.13 – Modélisation de la nature obligatoire d'une PosA à l'aide d'une règle d'inférence des graphes conceptuels.

Le fait que la PosA *container* est interdite pour le type d'unité linguistique /to graze\ peut être modélisé par une règle de contrainte négative, comme illustré par la figure 6.14 :

FIGURE 6.14 – Modélisation de la nature interdite d'une PosA à l'aide d'une règle de contrainte négative des graphes conceptuels.

Finalement, le formalisme des GC permet de spécifier la signature des relations. Nous pouvons donc déclarer le type d'unité linguistique qui introduit une PosA, et la signature de cette PosA pour ce type d'unité linguistique. Cependant, la signature d'une relation des GC ne permet pas de représenter directement la signature des unités linguistiques. Nous devons pour cela utiliser des règles d'inférence ou de contrainte. Par exemple, la signature /vegetal\ du type d'unité linguistique /to graze\ pour sa PosA *eaten* est modélisée par une règle d'inférence, comme illustré par la figure 6.15 :

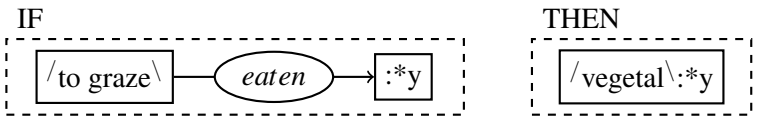


FIGURE 6.15 – Modélisation de la signature d'une PosA à l'aide d'une règle d'inférence des graphes conceptuels.

6.2.2.2 Modélisation des définitions linguistiques

Afin de représenter les définitions lexicographiques, nous ne pouvons pas utiliser la notion de définition de type de concept, car les fonctionnalités d'expansion et de contraction ne seraient pas représentées.

Expansion : il n'est pas possible d'identifier (au sens de fusionner) les PosA du participant défini avec les autres nœuds participants du graphe de définition.

Contraction : il n'est pas possible d'inférer les PosA du participant défini.

En supposant que chaque participant est obligatoire, nous devrions modéliser une définition lexicographique formelle à l'aide de deux règles d'inférence contraposées, comme illustré par la figure 6.16.

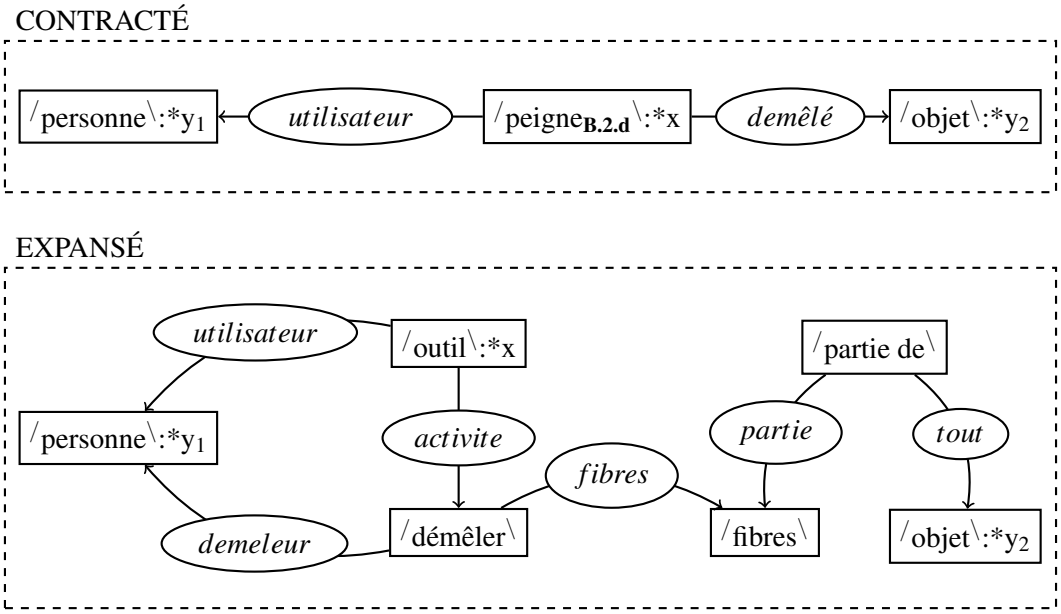


FIGURE 6.16 – Modélisation avec les Graphes Conceptuels de la définition lexicographique formelle de ⌈PEIGNE$_{\mathbf{B.2.D}}$⌉, en modélisant chaque relation actancielle par une relation binaire.

Cette solution de modélisation est donc la plus à même de satisfaire les critères de Gruber : les représentations sont claires, cohérentes, et présentent l'expressivité attendue.

Cependant, cette solution implique une déformation d'encodage que nous aimerions limiter. En effet, le support des graphes conceptuels n'encode pas naturellement le comportement particulier des Symboles de Relation Actancielle (SRelA) et des PosA, et le recours à des règles d'inférence devrait pouvoir être évité. Par ailleurs, nous préférerions associer à chaque type d'unité une sémantique différente de celle imposée par les Graphes Conceptuels. Nous aimerions en particulier qu'un type d'unité possède à la fois la sémantique d'un concept et d'une relation.

Enfin, si l'on veut ajouter un nouveau SRelA ou une nouvelle PosA au dictionnaire, il est nécessaire d'ajouter - à la main - les règles d'inférence associées. Tout oubli fausserait les inférences, et le formalisme des graphes conceptuels ne permet pas d'exprimer ce besoin de cohérence entre le support et l'ensemble des règles d'inférence.

Nous avons donc choisi de construire un nouveau formalisme de représentation des connaissances spécialement adapté aux prédicats linguistiques et aux définitions lexicographiques, tout en nous inspirant des graphes conceptuels.

Conclusion

Nous avons ainsi montré dans ce chapitre que ni les logiques de description, ni le formalisme des Graphes Conceptuels, ne sont adéquats pour formaliser les connaissances linguistiques de la TST.

Les logiques de description permettent de représenter les types d'unité linguistique avec une légère déformation d'encodage. Nous avons défini la projection d'une définition lexicographique sur un type de concept, et l'avons illustré sur deux définitions lexicographiques simples. Cependant, nous avons montré que la représentation ainsi obtenue n'est pas claire et ne respecte pas la contrainte de décidabilité. Nous avons ainsi montré que la capacité de modélisation des logiques de description est insatisfaisante pour la représentation des définitions lexicographiques de la TST.

Nous avons identifié deux options de modélisation à l'aide du formalisme des GC.

La première option, qui consiste à représenter un type d'unité linguistique à l'aide d'un couple ⟨type de concept, relation⟩, mène à un manque de clarté et une déformation d'encodage trop handicapante.

La seconde option consiste à modéliser chaque relation actancielle par une relation binaire. Nous avons montré qu'elle est proche de satisfaire les critères de Gruber. Cependant, nous avons montré qu'elle mène à une déformation d'encodage importante, que la sémantique naturelle des graphes conceptuels n'est pas une sémantique naturelle pour la TST, et que cette solution nécessiterait des mécanismes non standard de vérification de la cohérence de la base de règles d'inférence.

Nous proposons donc de construire un nouveau formalisme de RC spécialement adapté aux conceptualisations des prédicats linguistiques et des définitions lexicographiques présentées dans les chapitres précédents. Nous nous inspirerons néanmoins du formalisme des GC, qui présente le plus de similarités avec la TST. Le résultat de ce développement est le formalisme des *Graphes d'Unités*, qui fait l'objet du reste de ce mémoire.

CHAPITRE 7

Construction du formalisme des Graphes d'Unités pour la représentation des prédicats linguistiques

---

*_____ **L**'esprit n'use de sa faculté créatrice que quand l'expérience lui en impose la nécessité.*
Henri Poincaré

Sommaire

Introduction

Dans le chapitre précédent, nous avons montré que ni les logiques de description, ni le formalisme des Graphes Conceptuels, ne permettent de représenter les prédicats linguistiques et les définitions lexicographiques de la TST de manière adéquate. Nous proposons donc de développer un nouveau formalisme de RC adapté. Nous nous inspirerons néanmoins du formalisme des GC, qui présente le plus de similarités avec la TST.

Puisque nous représentons des unités linguistiques de différentes natures (ex : unités sémantiques, unités grammaticales, lexies), nous choisissons d'utiliser le terme *unité* de manière générique. Nous nommons donc le formalisme résultant *formalisme des Graphes d'Unités (GU)*.

La construction du formalisme des GU commence par l'introduction et la caractérisation d'une hiérarchie de types d'unité munis d'une structure actancielle, adaptée à la conceptualisation introduite au chapitre 3. Nous nous basons sur une distinction claire entre :
- les types d'unité, qui sont décrits dans le dictionnaire ;
- les unités, qui sont représentées dans des GU.

Nous introduirons tout d'abord une structure mathématique pour représenter les types d'unité munis d'une structure actancielle (§7.1). Nous avons montré dans la section 3.1.2 que les unités doivent pouvoir être multitypées. Nous étendrons donc cette première structure mathématique en une hiérarchie de types d'unité conjonctifs (§7.2). Nous caractériserons finalement formellement cette hiérarchie (§7.3).

7.1 Types d'unité primitifs (TUP)

Les types d'unité possèdent une structure actancielle formée de PosA qui ont des symboles, une nature optionnelle, obligatoire, ou interdite, et une signature. Ils sont organisés dans une structure hiérarchique notée $\mathscr{T}$.

Cette section introduit d'abord les types d'unité primitifs et les symboles de relation actancielle (§7.1.1). L'organisation hiérarchique des types d'unité primitifs (§7.1.2), permet d'en calculer les structures actancielles (§7.1.3). Finalement, nous définirons un ensemble de types d'unité primitifs *absurdes*, formé de ceux qui ont une PosA à la fois obligatoire et interdite (§7.1.4).

7.1.1 Définition des TUP et des symboles de relation actancielle (SRelA)

$\mathscr{T}$ contient un ensemble fini de *types d'unité primitifs (TUP) déclarés*, noté T_D.

Définition 26 (Ensemble des TUP déclarés)**.** L'ensemble des *types d'unité primitifs* est un ensemble fini noté T_D. Tout type d'unité primitif t est associé à une URI notée $\mathrm{uri}(t)$.

Cet ensemble contient tous les types d'unité linguistique de la TST, et nous pouvons donc y distinguer quatre sous-ensembles :
- l'ensemble T_D^{SemP} des TUSemP ;
- l'ensemble T_D^{SemS} des TUSemS ;
- l'ensemble T_D^{L} des types de lexie ;
- l'ensemble T_D^{G} des types d'unité grammaticale ;

Afin d'associer aux TUP une structure actancielle, $\mathscr{T}$ contient un ensemble de symboles de relation binaire appelés *symboles de relation actancielle (SRelA)*, noté $\boldsymbol{S}_{\mathscr{T}}$.

Définition 27 (Ensemble des SRelA)**.** Un *ensemble de symboles de relation actancielle* est un ensemble fini de relations binaires noté $\boldsymbol{S}_{\mathscr{T}}$. Tout symbole de relation actancielle s est associé à une URI notée uri(s).

Cet ensemble contient tous les symboles de relation actancielle de la TST, et il contient donc les trois sous-ensembles suivants :

- l'ensemble $\boldsymbol{S}_{\mathscr{T}}{}^{\text{SemP}}$ de rôles sémantiques lexicalisés pour le niveau sémantique profond ;
- l'ensemble $\boldsymbol{S}_{\mathscr{T}}{}^{\text{SemS}}$ de nombres pour le niveau sémantique de surface ;
- l'ensemble $\boldsymbol{S}_{\mathscr{T}}{}^{\text{SynP}} = \{\mathbf{I}, \ldots, \mathbf{VI}\}$ pour le niveau syntaxique profond.

Un TUP possède un ensemble (potentiellement vide) de PosA dont les symboles sont choisis dans l'ensemble des SRelA. Chaque PosA peut être obligatoire, optionnelle, ou interdite.

Afin de représenter ces différentes natures de PosA, et afin de leur assurer une présence cohérente dans la hiérarchie des types d'unité, nous introduisons trois bijections $\boldsymbol{\gamma}$, $\boldsymbol{\gamma_1}$, et $\boldsymbol{\gamma_0}$, de l'ensemble des SRelA vers trois sous-ensembles de T_D : l'ensemble $\boldsymbol{\Gamma}$ des racines de PosA, l'ensemble $\boldsymbol{\Gamma_1}$ des racines de Position Actancielle Obligatoire (PosAObl), et l'ensemble $\boldsymbol{\Gamma_0}$ des racines de Position Actancielle Interdite (PosAInt), respectivement.

Définition 28 (Racines de PosA, PosaObl et PosAInt)**.** La structure actancielle des TUP est calculée à partir de :

- $\boldsymbol{\Gamma}$ est l'ensemble des *racines de PosA*, les TUP qui introduisent les PosA ; $\boldsymbol{\gamma}$ est une bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma}$ qui associe à chaque SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$ sa *racine de PosA* s, le TUP $\boldsymbol{\gamma}(s)$ qui introduit une PosA de symbole s.
- $\boldsymbol{\Gamma_1}$ est l'ensemble des *racines de PosAObl*, les TUP qui rendent les PosA obligatoires ; $\boldsymbol{\gamma_1}$ est une bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma_1}$ qui associe à chaque SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$ sa *racine de PosAObl* s, le TUP $\boldsymbol{\gamma_1}(s)$ qui rend sa PosA de symbole s obligatoire.
- $\boldsymbol{\Gamma_0}$ est l'ensemble des *racines de PosAInt*, les TUP qui rendent les PosA interdites ; $\boldsymbol{\gamma_0}$ est une bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma_0}$ qui associe à chaque SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$ sa *racine de PosAInt* s, le TUP $\boldsymbol{\gamma_0}(s)$ qui rend sa PosA de symbole s interdite.
- $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma_1}$, et $\boldsymbol{\Gamma_0}$ sont disjoints deux à deux, et sont disjoints de l'ensemble des TUP déclarés T_D.

Les éléments de $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma_1}$, et $\boldsymbol{\Gamma_0}$ sont des TUP anonymes auxiliaires. Un travail futur possible consiste à étudier si l'on peut relaxer la définition de $\boldsymbol{\gamma}$, $\boldsymbol{\gamma_1}$, et $\boldsymbol{\gamma_0}$ à des applications de $\boldsymbol{S}_{\mathscr{T}}$ vers T_D. Introduisons maintenant deux TUP auxiliaires importants :

Définition 29 (TUP universel premier, et TUP absurde premier)**.** Nous introduisons deux TUP spéciaux :

- $\top$ est le *TUP universel premier*, associé à une URI notée uri$(\top)$. Par définition, toute unité est une instance de $\top$.
- $\bot$ est le *TUP absurde premier*, associé à une URI notée uri$(\bot)$. Par définition, aucune unité n'est instance de $\bot$.

Aucun des ensembles précédemment définis ne contient $\top$ ou $\bot$.

L'ensemble des TUP est alors noté $\boldsymbol{T}$, et est défini comme l'union disjointe de l'ensemble des TUP déclarés T_D, l'ensemble des racines de PosA $\boldsymbol{\Gamma}$, l'ensemble des racines de PosAObl $\boldsymbol{\Gamma_1}$, l'ensemble des racines de PosAInt $\boldsymbol{\Gamma_0}$, plus le *TUP universel premier* $\top$ et le *TUP absurde premier* $\bot$.

Définition 30 (Ensemble des TUP)**.** Un *ensemble de TUP* est l'union disjointe notée

$$\boldsymbol{T} \stackrel{\text{def}}{=} T_D \cup \boldsymbol{\Gamma} \cup \boldsymbol{\Gamma_1} \cup \boldsymbol{\Gamma_0} \cup \{\top\} \cup \{\bot\}$$

où :

- T_D est l'ensemble fini de *TUP déclarés* ;
- $\mathbf{\Gamma}$ est l'ensemble des racines de PosA ;
- $\mathbf{\Gamma_1}$ est l'ensemble des racines de PosAObl ;
- $\mathbf{\Gamma_0}$ est l'ensemble des racines de PosAInt ;
- $\top$ est le *TUP universel premier* ;
- $\bot$ est le *TUP absurde premier*.

Visualisation. Nous réutilisons les propositions du chapitre 3 concernant la visualisation des types d'unité inspirée d'UML, comme illustré par la figure 7.1.

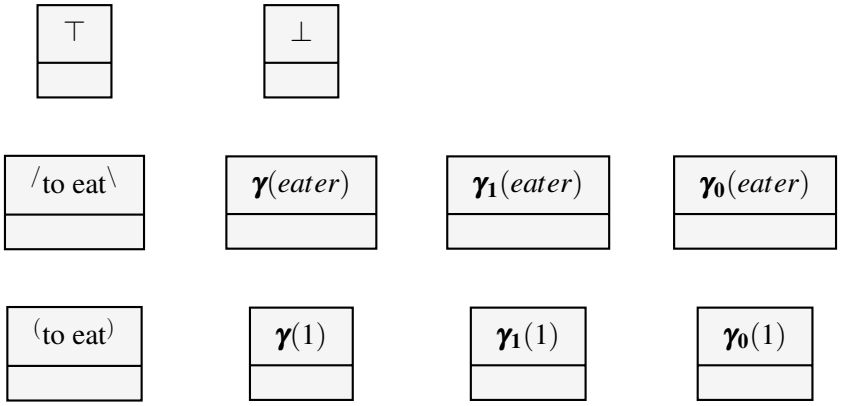


FIGURE 7.1 – Visualisation de différents TUP.

7.1.2 Classification des TUP

Nous souhaitons proposer un moyen d'oganiser les TUP en une hiérarchie. Nous introduisons donc dans cette section une relation de spécialisation sous la forme d'un préordre (§7.1.2.1), puis caractérisons l'ensemble des classes d'équivalence de TUP (§7.1.2.2).

7.1.2.1 Préordre sur l'ensemble des TUP

Nous introduisons un préordre $\lesssim$ sur l'ensemble des TUP $\boldsymbol{T}$, qui modélise une relation de spécialisation par héritage. $t_1 \lesssim t_2$ modélise le fait que t_1 est plus spécifique que t_2, par exemple, $/$to graze$\backslash \lesssim /$to eat$\backslash$ signifie que le TUSemP $/$to graze$\backslash$ est plus spécifique que le TUSemP $/$to eat$\backslash$.

Définition 31 (Préordre sur l'ensemble $\boldsymbol{T}$)**.** L'ensemble des TUP est préordonné par une relation $\lesssim$, qui est le plus petit préordre tel que chacun des points ci-dessous est vrai :

- $\lesssim$ contient l'ensemble C_A des *comparaisons déclarées* ($C_A \subseteq \boldsymbol{T}^2$), i.e., pour tout $(t_y, t_x) \in C_A$, on a $t_x \lesssim t_y$;
- l'ensemble des TUP est borné par un élément maximal $\top$, et un élément minimal $\bot$;
- pour toute SRelA s, la racine de PosAObl s et la racine de PosAInt s sont plus spécifiques que la racine de PosA s, i.e., pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$, on a $\boldsymbol{\gamma_1}(s) \lesssim \boldsymbol{\gamma}(s)$ et $\boldsymbol{\gamma_0}(s) \lesssim \boldsymbol{\gamma}(s)$.

Si $t_x \lesssim t_y$, alors toute unité de type t_x est également de type t_y.

Nous introduisons un graphe orienté nommé *ensemble de comparaisons* sur $\boldsymbol{T}$, et démontrons qu'il est égal à la relation de préordre $\lesssim$.

Définition 32 (Ensemble de comparaisons sur l'ensemble $\boldsymbol{T}$)**.** Considérons le graphe orienté $(\boldsymbol{T}, \boldsymbol{C_T})$ défini sur $\boldsymbol{T}$, où :

- $\boldsymbol{C_T} \stackrel{\text{def}}{=} C_A \cup C_\top \cup C_\bot \cup C_{\boldsymbol{\Gamma_1}} \cup C_{\boldsymbol{\Gamma_0}} \subseteq \boldsymbol{T}^2$;
- $C_\top \stackrel{\text{def}}{=} \{(\top, t)\}_{t \in \boldsymbol{T}}$;
- $C_\bot \stackrel{\text{def}}{=} \{(t, \bot)\}_{t \in \boldsymbol{T}}$;
- $C_{\boldsymbol{\Gamma_1}} \stackrel{\text{def}}{=} \{(\boldsymbol{\gamma}(s), \boldsymbol{\gamma_1}(s))\}_{s \in \boldsymbol{S}_{\mathscr{T}}}$;
- $C_{\boldsymbol{\Gamma_0}} \stackrel{\text{def}}{=} \{(\boldsymbol{\gamma}(s), \boldsymbol{\gamma_0}(s))\}_{s \in \boldsymbol{S}_{\mathscr{T}}}$.

La clôture réflexo-transitive de $\boldsymbol{C_T}$, notée $\boldsymbol{C_T^*}$, est l'*ensemble des comparaisons de TUP*, i.e., $(t, t') \in \boldsymbol{C_T^*}$ ssi t' est un descendant de t dans $\boldsymbol{C_T}$.

Proposition 7.1.1. *La relation de préordre $\lesssim$ sur* $\mathbf{T}$ *est égale à* $\mathbf{C_T^*}$*, i.e.,*

$$\forall t, t' \in \mathbf{T}, t' \lesssim t \iff (t, t') \in \mathbf{C_T^*}$$

Preuve: voir annexe, p. 303.

Visualisation. La figure 7.2 illustre quelques spécialisations dans l'ensemble des TUP. Parmi toutes les spécialisation représentées, seule celle qui lie $/$to graze$\backslash$ à $/$to eat$\backslash$ doit être dans l'ensemble des comparaisons déclarées. Toutes les autres sont imposées par la définition 31.

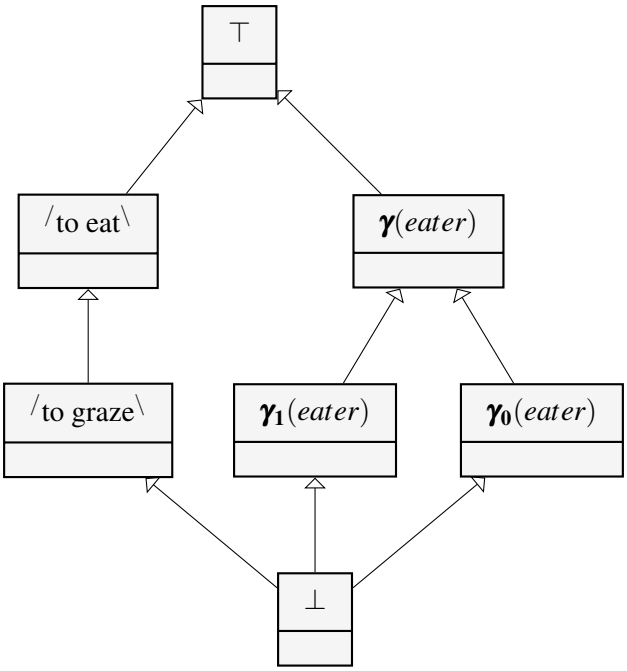


FIGURE 7.2 – Visualisation de différentes spécialisations de TUP.

7.1.2.2 Ordre partiel sur les classes d'équivalence de TUP

Soit $\simeq$ la *relation d'équivalence* [1] *naturelle* sur l'ensemble des TUP définie par :

$$t \simeq t' \iff t \lesssim t' \text{ et } t' \lesssim t$$

L'ensemble des classes d'équivalence de TUP définit une partition de $\boldsymbol{T}$. Soit $t \in \boldsymbol{T}$, nous notons $[t]$ la classe d'équivalence à laquelle t appartient, i.e., $[t] \stackrel{\text{def}}{=} \{t' \in \boldsymbol{T} \mid t' \simeq t\}$.

Définition 33 (Ensemble des classes d'équivalence de TUP)**.** L'*ensemble des classes d'équivalence de TUP* $\boldsymbol{T}^{\sim}$ est le quotient de l'ensemble $\boldsymbol{T}$ par $\simeq$, i.e.,

$$\boldsymbol{T}^{\sim} \stackrel{\text{def}}{=} \boldsymbol{T}/\simeq = \{[t] \mid t \in \boldsymbol{T}\}$$

Nous notons $t^{\sim}$ une classe d'équivalence de TUP variable. Définissons un ordre partiel $\stackrel{\sim}{\lesssim}$ sur $\boldsymbol{T}/\simeq$:

$$t_1^{\sim} \stackrel{\sim}{\lesssim} t_2^{\sim} \text{ si et seulement si } (\exists t_1 \in t_1^{\sim})(\exists t_2 \in t_2^{\sim})[t_1 \lesssim t_2]$$

$\top$ est le TUP universel premier, et est par définition le type de toute unité. Notons $\top^{\sim} \stackrel{\text{def}}{=} [\top]$ la classe d'équivalence à laquelle $\top$ appartient. Tout type qui appartient à $\top^{\sim}$ est dit *universel*, et est le type de toute unité.

$\bot$ est le TUP absurde premier, et n'est par définition le type d'aucune unité. Ainsi, tout TUP qui est plus spécifique que $\bot$ n'est également le type d'aucune unité. Puisque $\bot$ est également un élément minimal de $\boldsymbol{T}$, tout TUP qui est plus spécifique que $\bot$ est en fait équivalent à $\bot$. Notons $\bot^{\sim} \stackrel{\text{def}}{=} [\bot]$ la classe d'équivalence à laquelle $\bot$ appartient. Tout type d'unité dans $\bot^{\sim}$ est dit *absurde*, et n'est le type d'aucune unité. Nous verrons dans la section 7.1.4 que d'autres TUP peuvent également être absurdes.

1. Une relation d'équivalence est une relation réflexive, symétrique, et transitive.

Visualisation. Nous réutilisons la représentation graphique de la spécialisation de TUP afin de visualiser la spécialisation des classes d'équivalence de TUP. Par exemple, la figure 7.3 illustre deux représentations qui sont équivalentes.

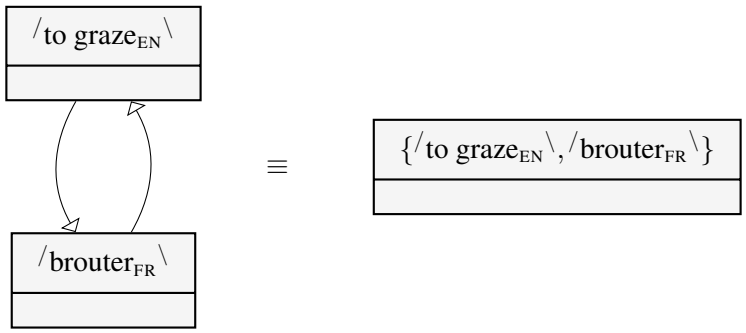


FIGURE 7.3 – Visualisation d'une classe d'équivalence de TUP.

La figure 7.4 représente un TUP par un point, et si $t \lesssim t'$, alors la représentation de t est en-dessous de la représentation de t'. De plus, les classes d'équivalence de TUP sont représentées par des cercles, et tous les points à l'intérieur d'un cercle donné représentent des TUP équivalents. Ainsi, le cercle qui représente $\bot^{\sim}$ (resp. $\bot^{\sim}$) est au sommet (resp. à la base).

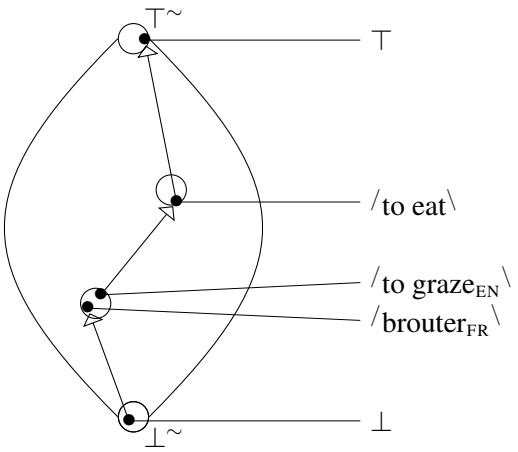


FIGURE 7.4 – Illustration des TUP et des classes d'équivalence. $(/\text{to eat}\backslash, /\text{to graze}_{EN}\backslash) \in C_A$, $(\top, /\text{to eat}\backslash) \in C_\top$, $(/\text{to graze}_{EN}\backslash, \bot) \in C_\bot$.

7.1.3 Structure actancielle d'un TUP

Qu'il soit sémantique profond, de surface, lexical, ou grammatical, un TUP $t \in \boldsymbol{T}$ possède un ensemble (potentiellement vide) de positions actancielles (PosA) dont les symboles sont choisis dans l'ensemble des SRelA $\boldsymbol{S}_{\mathscr{T}}$.

Cette section introduit d'abord les PosA et les PosA obligatoires et interdites des TUP (§7.1.3.1). Nous en dérivons ensuite la définition des PosA optionnelles (§7.1.3.2). Finalement, nous précisons la représentation des signatures, qui permettent de spécifier le type des unités qui prennent les PosA (§7.1.3.3).

7.1.3.1 Positions actancielles (PosA) obligatoires et interdites

Puisque toute PosA possède un symbole, l'ensemble des PosA d'un TUP $t \in \boldsymbol{T}$ est défini par l'ensemble de leurs symboles, noté $\boldsymbol{\alpha}(t)$. Informellement, la structure actancielle de t est définie par les ensembles de racines de PosA, de PosAObl, et de PosAInt :

- t possède une PosA pour chaque SRelA dont la racine de PosA est plus générale ou équivalente à t, et seulement pour ces SRelA ;
- t possède une PosAObl pour chaque SRelA dont la racine de PosAObl est plus générale ou équivalente à t, et seulement pour ces SRelA ;
- t possède une PosAInt pour chaque SRelA dont la racine de PosAInt est plus générale ou équivalente à t, et seulement pour ces SRelA.

Définition 34 (PosA d'un TUP, et valence d'un TUP)**.** L'*ensemble des PosA* d'un TUP est défini par une application $\boldsymbol{\alpha}$ de $\boldsymbol{T}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$, qui associe à chaque TUP $t \in \boldsymbol{T}$ l'ensemble de SRelA dont la racine de PosA est plus générale que t, i.e.,

$$\forall t \in \boldsymbol{T},\ \boldsymbol{\alpha}(t) \stackrel{\text{def}}{=} \{s \in \boldsymbol{S}_{\mathscr{T}} \mid t \lesssim \boldsymbol{\gamma}(s)\}$$

Pour tout $t \in \boldsymbol{T}$, $s \in \boldsymbol{\alpha}(t)$ signifie que t possède une PosA avec le SRelA s. Le nombre de PosA d'un TUP t est appelé la *valence* de t, i.e., $valence(t) \stackrel{\text{def}}{=} |\boldsymbol{\alpha}(t)|$.

Définition 35 (PosAObl d'un TUP)**.** L'*ensemble des Positions Actancielles Obligatoires (PosAObl)* d'un TUP est défini par une application $\boldsymbol{\alpha_1}$ de $\boldsymbol{T}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$, qui associe à chaque TUP $t \in \boldsymbol{T}$ l'ensemble de SRelA dont la racine de PosAObl est plus générale que t,

$$\forall t \in \boldsymbol{T},\ \boldsymbol{\alpha_1}(t) \stackrel{\text{def}}{=} \{s \in \boldsymbol{S}_{\mathscr{T}} \mid t \lesssim \boldsymbol{\gamma_1}(s)\}$$

Pour tout $t \in \boldsymbol{T}$, $s \in \boldsymbol{\alpha_1}(t)$ signifie que t possède une PosAObl avec le SRelA s.

Définition 36 (PosAInt d'un TUP)**.** L'*ensemble des Positions Actancielles Interdites (PosAInt)* d'un TUP est défini par une application $\boldsymbol{\alpha_0}$ de $\boldsymbol{T}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$, qui associe à chaque TUP $t \in \boldsymbol{T}$ l'ensemble de SRelA dont la racine de PosAInt est plus générale que t,

$$\forall t \in \boldsymbol{T},\ \boldsymbol{\alpha_0}(t) \stackrel{\text{def}}{=} \{s \in \boldsymbol{S}_{\mathscr{T}} \mid t \lesssim \boldsymbol{\gamma_0}(s)\}$$

Pour tout $t \in \boldsymbol{T}$, $s \in \boldsymbol{\alpha_0}(t)$ signifie que t possède une PosAInt avec le SRelA s.

Note. Soit s une SRelA. Par abus de langage, plutôt que d'écrire *PosA avec le SRelA* s, nous écrivons simplement : *PosA* s.

Note. Dans la suite de ce mémoire, nous utilisons *plus général* et *plus spécifique* dans leur sens faible, i.e., respectivement *plus général ou équivalent* et *plus spécifique ou équivalent*, sauf mention contraire.

Par exemple, et dans un but d'illustration, supposons que le TUP /to graze\ soit plus spécifique que :

- la racine de PosAObl *eater* ;
- la racine de PosA *eaten* ;
- la racine de PosAInt *container*.

Alors nous en déduisons que :

- $eater \in \boldsymbol{\alpha_1}(/\text{to graze}\backslash)$;
- $eaten \in \boldsymbol{\alpha}(/\text{to graze}\backslash)$;
- $container \in \boldsymbol{\alpha_0}(/\text{to graze}\backslash)$.

La partie gauche de la figure 7.5 illustre cette situation.

Visualisation. La partie droite de la figure 7.5 représente les PosA, les PosAObl et les PosAInt d'un TUP dans la partie inférieure de la représentation des TUP, comme spécifié dans la section 3.1.3.

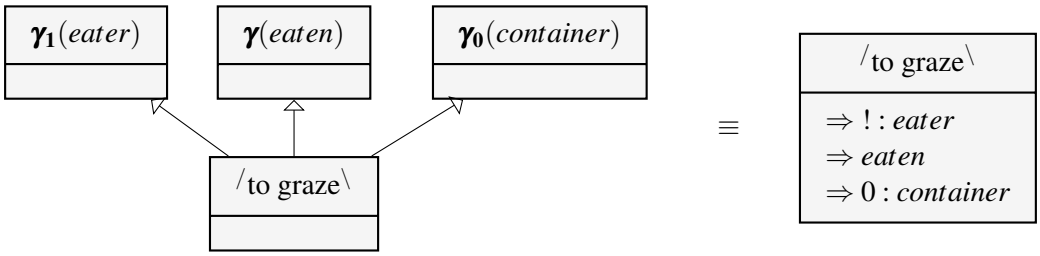


FIGURE 7.5 – Visualisation des PosA, PosAObl, et PosAInt de /to graze\

Implications pour la hiérarchie de TUP. Par implication directe de la définition des PosA, l'ensemble des TUP qui ont une PosA (resp. PosAObl, PosAInt) d'une SRelA donnée $s \in \boldsymbol{S}_{\mathscr{T}}$ est l'ensemble des TUP qui sont plus spécifiques que la la racine de PosA (resp. de PosAObl, de PosAInt) s.

Proposition 7.1.2. *Soit $\downarrow t$ le plus petit ensemble inférieur de* $\mathbf{T}$ *qui contient t, i.e., $\downarrow t \stackrel{def}{=} \{t' \in \mathbf{T} \mid t' \lesssim t\}$. Pour tout $s \in \mathbf{S}_{\mathscr{T}}$, les trois égalités suivantes sont vraies :*

$$\{t \in \mathbf{T} \mid s \in \boldsymbol{\alpha}(t)\} = \quad \downarrow \boldsymbol{\gamma}(s) \tag{7.1}$$

$$\{t \in \mathbf{T} \mid s \in \boldsymbol{\alpha_1}(t)\} = \quad \downarrow \boldsymbol{\gamma_1}(s) \tag{7.2}$$

$$\{t \in \mathbf{T} \mid s \in \boldsymbol{\alpha_0}(t)\} = \quad \downarrow \boldsymbol{\gamma_0}(s) \tag{7.3}$$

Preuve: voir annexe, p. 304.

La figure 7.6 illustre l'ensemble des TUP qui ont :
– une PosA *eater*, i.e., l'ensemble inférieur principal de $\boldsymbol{T}$ généré par $\boldsymbol{\gamma}(eater)$;
– une PosAObl *eater*, i.e., l'ensemble inférieur principal de $\boldsymbol{T}$ généré par $\boldsymbol{\gamma_1}(eater)$;
– une PosAInt *eater*, i.e., l'ensemble inférieur principal de $\boldsymbol{T}$ généré par $\boldsymbol{\gamma_0}(eater)$.

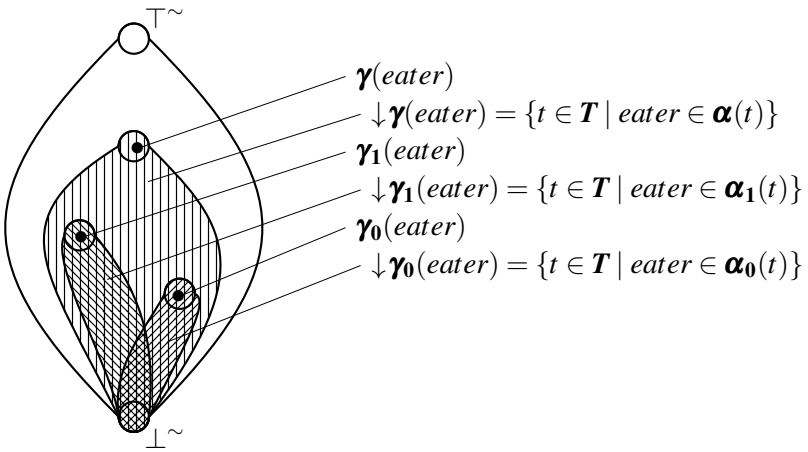


FIGURE 7.6 – Illustration des TUP qui ont *eater* comme PosA (hachures ||) ; comme PosAObl (hachures \\) ; comme PosAInt (hachures //).

En conséquence, quand on navigue vers le bas dans la hiérarchie des TUP, et quand les TUP deviennent de plus en plus spécifiques, l'ensemble des PosA (resp. PosAObl, PosAInt) ne peut qu'augmenter. Nous disons que les PosA (resp. PosAObl, PosAInt) sont *héritées* :

Proposition 7.1.3. *Les PosA, PosAObl et PosAInt sont héritées, i.e., pour tout* $t_x, t_y \in \mathbf{T}$*, si* $t_x \lesssim t_y$*, alors les trois inclusions ci-dessous sont vraies.*

$$\boldsymbol{\alpha}(t_y) \subseteq \boldsymbol{\alpha}(t_x) \tag{7.4}$$
$$\boldsymbol{\alpha_1}(t_y) \subseteq \boldsymbol{\alpha_1}(t_x) \tag{7.5}$$
$$\boldsymbol{\alpha_0}(t_y) \subseteq \boldsymbol{\alpha_0}(t_x) \tag{7.6}$$

De plus, si $t_x \simeq t_y$*, alors* $\boldsymbol{\alpha}(t_y) = \boldsymbol{\alpha}(t_x)$, $\boldsymbol{\alpha_1}(t_y) = \boldsymbol{\alpha_1}(t_x)$, *et* $\boldsymbol{\alpha_0}(t_y) = \boldsymbol{\alpha_0}(t_x)$.

Preuve: voir annexe, p. 304.

Ensuite, pour chaque SRelA *s*, la racine de PosAObl *s* (resp. de PosAInt *s*) est inférieure à la racine de PosA *s*. Donc l'ensemble des PosAObl (resp. des PosAInt) est toujours un sous-ensemble de l'ensemble des PosA :

Proposition 7.1.4. *Pour tout TUP* $t \in \mathbf{T}$*,*

$$\boldsymbol{\alpha_1}(t) \subseteq \boldsymbol{\alpha}(t) \tag{7.7}$$
$$\boldsymbol{\alpha_0}(t) \subseteq \boldsymbol{\alpha}(t) \tag{7.8}$$

Preuve: voir annexe, p. 304.

Puisque $\boldsymbol{\gamma}$ (resp. $\boldsymbol{\gamma_1}$, $\boldsymbol{\gamma_0}$) est une application, toute SRelA possède une racine de PosA (resp. de PosAObl, de PosAInt). Tout élément minimal de $\boldsymbol{T}$, i.e., un TUP dans $\perp^{\sim}$, héritera de chacune des PosA (resp. PosAObl, PosAInt) :

Proposition 7.1.5. $\forall t \in \perp^{\sim}$,

$$\boldsymbol{\alpha}(t) = \boldsymbol{\alpha_1}(t) = \boldsymbol{\alpha_0}(t) = \mathbf{S}_{\mathscr{T}}$$

Preuve: voir annexe, p. 304.

7.1.3.2 PosA optionnelles

Soit $t \in \boldsymbol{T}$ un TUP. Nous définissons l'ensemble des Positions Actancielles Optionnelles (PosAOpt) de t comme étant l'ensemble des PosA qui ne sont ni obligatoires ni optionnelles.

Définition 37 (PosAOpt d'un TUP)**.** L'*ensemble des Positions Actancielles Optionnelles (PosAOpt)* d'un TUP est défini par une application $\boldsymbol{\alpha_?}$ de $\boldsymbol{T}$ vers $2^{\mathbf{S}_{\mathscr{T}}}$, qui associe à chaque TUP $t \in \boldsymbol{T}$ l'ensemble $\boldsymbol{\alpha_?} = \boldsymbol{\alpha} - \boldsymbol{\alpha_1} - \boldsymbol{\alpha_0}$.

Nous représentons les PosAOpt comme spécifié dans la section 3.1.3. La figure 7.7 illustre le fait que le TUP /to eat\ a sa PosA *container* optionnelle.

/to eat\
$\Rightarrow$? : *container*

FIGURE 7.7 – Visualisation de la PosAOpt *container* de /to eat\

Par la proposition 7.1.2, nous savons que tout TUP avec une PosAOpt s est dans l'ensemble inférieur $\downarrow\boldsymbol{\gamma}(s)$, et n'est pas dans les ensembles inférieurs $\downarrow\boldsymbol{\gamma_1}(s)$ ni $\downarrow\boldsymbol{\gamma_0}(s)$. Donc la proposition suivante est vraie :

Proposition 7.1.6. *Pour tout* $s \in \mathbf{S}_{\mathscr{T}}$,

$$\begin{aligned} \{t \in \mathbf{T} \mid s \in \boldsymbol{\alpha_?}(t)\} &= \downarrow\boldsymbol{\gamma}(s) - \downarrow\boldsymbol{\gamma_1}(s) - \downarrow\boldsymbol{\gamma_0}(s) \\ &= \{t \in \mathbf{T} \mid t \lesssim \boldsymbol{\gamma}(s) \text{ et } t \not\lesssim \boldsymbol{\gamma_1}(s) \text{ et } t \not\lesssim \boldsymbol{\gamma_0}(s)\} \end{aligned} \tag{7.9}$$

Preuve: voir annexe, p. 304.

La figure 7.6 illustre cette proposition : les TUP qui n'ont la PosA *eater* ni obligatoire ni interdite, l'ont optionnelle (les TUP dans la région seulement hachurée ⫴).

Ainsi, lorsqu'on navigue vers le bas de la hiérarchie des TUP, une PosA s est introduite par $\boldsymbol{\gamma}(s)$ et définit d'abord une PosAOpt jusqu'à ce que le TUP devienne plus spécifique que $\boldsymbol{\gamma_1}(s)$ (resp. $\boldsymbol{\gamma_0}(s)$). Si cela arrive, alors la PosA s devient obligatoire (resp. interdite). Maintenant, par la proposition 7.1.3, nous en déduisons que deux TUP équivalents possèdent le même ensemble de PosAOpt :

Proposition 7.1.7. $(\forall t,t' \in \mathbf{T})[t \simeq t' \implies \boldsymbol{\alpha_?}(t) = \boldsymbol{\alpha_?}(t')]$.

Preuve: voir annexe, p. 305.

Finalement, par la proposition 7.1.5, nous en déduisons que tout élément minimal de $\boldsymbol{T}$, i.e., tout TUP dans $\perp^{\sim}$, ne possède aucune PosAOpt.

Proposition 7.1.8. $(\forall t \in \perp^{\sim})[\boldsymbol{\alpha_?}(t) = \varnothing]$.

Preuve: voir annexe, p. 305.

7.1.3.3 Signatures des TUP

Nous souhaitons représenter les *signatures des TUP*, définies dans la section 3.10.

Définition 38 (Signatures des TUP)**.** L'*ensemble des signatures des TUP* $\{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}}$ est un ensemble de fonctions de $\boldsymbol{S}_{\mathscr{T}}$ vers $2^{\boldsymbol{T}}$. Pour tout TUP t, $\boldsymbol{\varsigma}_t$ est une fonction de domaine $domaine(\boldsymbol{\varsigma}_t) = \boldsymbol{\alpha}(t)$ qui associe à chaque PosA s de t un ensemble de TUP $\boldsymbol{\varsigma}_t(s)$ qui caractérisent le type des unités qui remplissent cette PosA. L'ensemble des signatures $\{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}}$ doit être tel que :

$$(\forall t_1,t_2 \in \boldsymbol{T})(\forall s \in \boldsymbol{S}_{\mathscr{T}})\left[(t_1 \lesssim t_2 \wedge s \in \boldsymbol{\alpha}(t_2)) \implies (\forall t_2' \in \boldsymbol{\varsigma}_{t_2}(s))(\exists t_1' \in \boldsymbol{\varsigma}_{t_1}(s))\left[t_1' \lesssim t_2'\right]\right]$$

La définition ci-dessus est complexe à cause du fait que des notions importantes seront introduites dans la section suivante. Nous verrons que $2^{\boldsymbol{T}}$ est l'ensemble des Types d'Unité Conjonctifs (TUC), et qu'ainsi la signature $\boldsymbol{\varsigma}_t$ de t est un TUC. De plus, lorsque la relation de préordre $\overset{\cap}{\lesssim}$ sur l'ensemble des TUC sera introduite, nous verrons que $\forall t_2' \in \boldsymbol{\varsigma}_{t_2}(s), \exists t_1' \in \boldsymbol{\varsigma}_{t_1}(s) : t_1' \lesssim t_2'$ implique que $\boldsymbol{\varsigma}_{t_1}(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_2}(s)$. Ainsi l'ensemble des signatures $\{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}}$ doit être tel que pour tout $t_1,t_2 \in \boldsymbol{T}$, et $s \in \boldsymbol{S}_{\mathscr{T}}$ tels que $s \in \boldsymbol{\alpha}(t_2)$,

$$t_1 \lesssim t_2 \Rightarrow \boldsymbol{\varsigma}_{t_1}(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_2}(s) \tag{7.10}$$

Choix de visualisation. Nous retrouvons ici la visualisation proposée dans la section 3.10. Nous représentons la signature d'une PosA d'un TUP en face de sa PosA, comme illustré par la figure 7.8.

/to eat\
$\Rightarrow$! : *eater* : {/animal\}

FIGURE 7.8 – Visualisation de la signature de la PosA *eater* de /to eat\

7.1.4 TUP absurdes

Comme l'illustre la figure 7.6, il peut arriver que certains TUP aient à la fois une PosAObl et une PosAInt ayant le même SRelA s. C'est le cas pour tous les TUP dans $\perp^{\sim}$, comme le montre la proposition 7.1.5, mais cela peut également être le cas pour un TUP t qui n'est pas équivalent à $\perp$.

Nous utilisons la proposition 7.1.5 pour définir l'ensemble des TUP absurdes, comme étant l'union de :

- l'ensemble des TUP qui possèdent au moins une PosA à la fois obligatoire et interdite ;
- l'ensemble des TUP minimaux.

Définition 39 (Ensemble des TUP absurdes)**.** L'*ensemble des TUP absurdes* est noté $A^{\sim}$ et est défini comme étant l'ensemble :

$$A^{\sim} \stackrel{\text{def}}{=} \{t \in \boldsymbol{T} \mid \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing\} \cup \perp^{\sim}$$

Tout TUP dans $A^{\sim}$ n'est le type d'aucune unité.

Proposition 7.1.9. *Si $s \neq \varnothing$, alors* $\perp^{\sim} \subseteq \{t \in \mathbf{T} \mid \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing\}$

Preuve: voir annexe, p. 305.

Le formalisme des GU diffère des autres formalismes de RC par cette définition, puisque des TUP non minimaux peuvent être absurdes. Cette définition peut ainsi sembler prêter à confusion, mais nous verrons qu'elle redevient naturelle pour les TUC (cf., définition 49 page 143).

7.2 Hiérarchie des types d'unité conjonctifs (TUC)

Le formalisme des GU prend en compte le multitypage des unités. Par exemple, une unité syntaxique profonde peut avoir un type lexical et un ou plusieurs types grammaticaux (cf., §3.1.2.1), comme $\{\ulcorner \text{PERSONNE}_{\mathbf{1.I.A}} \urcorner, singulier, defini\}$ pour le type de *"la personne"*.

La section 7.2.1 introduit d'abord l'ensemble des Types d'Unité Conjonctifs (TUC) comme étant l'ensemble des parties de $\boldsymbol{T}$. Nous définissons ensuite l'ensemble des TUC déclarés absurdes. Chaque TUC déclaré absurde est un ensemble de TUP qui ne doivent pas conjointement typer une unité. Tout TUC possède également une structure actancielle, définie à partir de celles des TUP qui le composent. La section 7.2.2 définit les PosA et les signatures des TUC. Ensuite, la section 7.2.3 introduit une relation de préordre de spécialisation sur l'ensemble des TUC, qui est construite à partir du préordre $\lesssim$, des TUC déclarés absurdes, et des signatures absurdes. Une définition naturelle des TUC absurdes est alors donnée, et nous justifions l'intérêt de travailler avec un tel ensemble préordonné de TUC plutôt que de travailler sur l'ensemble préordonné de TUP. Finalement, la section 7.2.4 définit formellement la hiérarchie des types d'unité.

7.2.1 Définition des TUC

Une unité peut être conjointement typée par plusieurs TUP. Nous introduisons donc l'ensemble des Types d'Unité Conjonctifs (TUC) comme étant l'ensemble des parties de $\boldsymbol{T}$.

Définition 40 (Ensemble des TUC)**.** L'*ensemble des TUC* sur $\boldsymbol{T}$ est l'ensemble des parties de $\boldsymbol{T}$, i.e., $\boldsymbol{T}^{\cap} \stackrel{\text{def}}{=} 2^{\boldsymbol{T}}$. $\top^{\cap} = \{\top\}$ est appelé le *TUC universel premier*, et $\bot^{\cap} = \{\bot\}$ le *TUC absurde premier*. Tout TUC singleton $\{t\}$ est dit *primitif*, i.e., un TUC primitif.

La figure 7.9 illustre le passage de l'ensemble des TUP à l'ensemble des TUC.

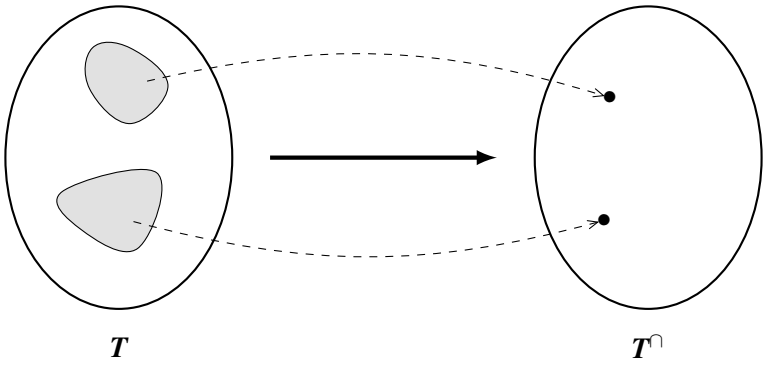


FIGURE 7.9 – Illustration du passage des TUP aux TUC.

Le nombre de types d'unité explose de manière combinatoire ici. Cependant, l'intuition veut que peu de TUP seront compatibles, et qu'ainsi beaucoup de $t^{\cap} \in \boldsymbol{T}^{\cap}$ ne pourront pas être instanciés. L'ensemble des TUC devrait donc être creux.

En particulier, certains TUC tels que $\{def, indef\}$ seront déclarés absurdes. Cela signifie qu'aucune unité ne peut avoir à la fois les types def et $indef$. L'ensemble des TUC déclarés absurdes est défini comme suit.

Définition 41 (Ensemble des TUC déclarés absurdes)**.** L'ensemble des *TUC déclarés absurdes* est un ensemble des parties de l'ensemble des TUP déclarés, noté $\bot_A^{\sqcap}$, i.e., $\bot_A^{\sqcap} \subseteq 2^{T_D} \subseteq \boldsymbol{T}^{\cap}$.

Bien que $\{\bot\}$ et chacun des couples $\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$ pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$ ne puissent pas être déclarés absurdes, on verra qu'ils sont absurdes par définition. $\bot_A^{\sqcap}$ contient seulement les TUC que l'on a réellement besoin de déclarer absurdes. L'ensemble complet des TUC absurdes sera défini dans la section 7.2.3. La figure 7.10 illustre deux TUC déclarés absurdes.

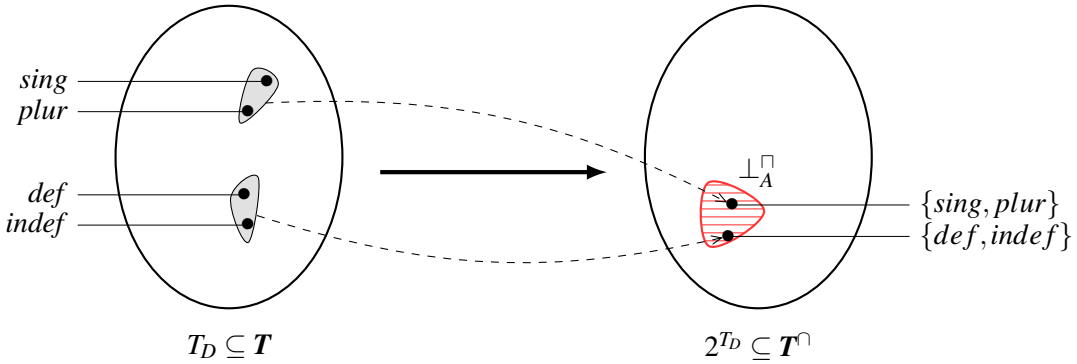


FIGURE 7.10 – Illustration de l'ensemble des TUC déclarés absurdes $\bot_A^{\sqcap}$.

Visualisation. La figure 7.11 illustre un TUC $\{\lceil \text{PERSONNE}_{\textbf{1.I.A}} \rceil, singulier, defini\}$ (*"une personne"*). Nous étendons encore la représentation graphique des classes d'équivalence de TUP de la figure 7.3.

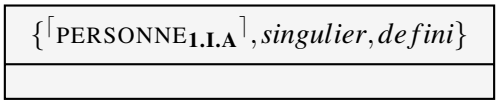


FIGURE 7.11 – Visualisation d'un TUC.

7.2.2 Structure actancielle d'un TUC

La structure actancielle des TUP, i.e., leurs PosA, PosAObl, PosAInt, PosAOpt et signatures, est naturellement étendue aux TUC.

Par exemple, supposons que l'on définisse un TUC $t^{\cap} = \{/\text{to eat}\backslash, /\text{quick}\backslash\}$. $t^{\cap}$ contient le TUP $/\text{to eat}\backslash$ qui possède une PosA *eater*. Puisque $t^{\cap}$ représente une spécialisation de $/\text{to eat}\backslash$, nous construisons le formalisme des GU avec l'hypothèse que $t^{\cap}$ hérite de la structure actancielle des TUP qui le composent [2]. Ainsi, *eater* est également une PosA de $t^{\cap}$:

Définition 42 (PosA d'un TUC, et valence d'un TUC)**.** L'*ensemble des PosA d'un TUC* est défini par une application $\boldsymbol{\alpha}^{\cap}$ de $\boldsymbol{T}^{\cap}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$ telle que :

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})[\boldsymbol{\alpha}^{\cap}(t^{\cap}) \stackrel{\text{def}}{=} \bigcup_{t \in t^{\cap}} \boldsymbol{\alpha}(t)] \tag{7.11}$$

Le nombre de PosA d'un TUC $t^{\cap}$ est appelé *valence* de $t^{\cap}$, i.e.,

$$valence(t^{\cap}) \stackrel{\text{def}}{=} |\boldsymbol{\alpha}^{\cap}(t^{\cap})| \tag{7.12}$$

2. Cette hypothèse ne nous permet pas de représenter une situation dans laquelle un type d'unité grammaticale bloquerait une PosA d'un type de lexie qu'il complémenterait. Même si nous découvrons l'existence de tels cas, nous prévoyons de les prendre en compte à l'aide de règles spécifiques, cf., chapitre 11.

Les définitions des PosAObl et des PosAInt des TUP sont naturellement étendues aux TUC.

Définition 43 (PosAObl d'un TUC)**.** L'*ensemble des PosAObl d'un TUC* est défini par une application $\boldsymbol{\alpha}_{\mathbf{1}}^{\cap}$ de $\boldsymbol{T}^{\cap}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$ telle que :

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})[\boldsymbol{\alpha}_{\mathbf{1}}^{\cap}(t^{\cap}) \stackrel{\text{def}}{=} \bigcup_{t \in t^{\cap}} \boldsymbol{\alpha}_{\mathbf{1}}(t)] \tag{7.13}$$

Définition 44 (PosAInt d'un TUC)**.** L'*ensemble des PosAInt d'un TUC* est défini par une application $\boldsymbol{\alpha}_{\mathbf{0}}^{\cap}$ de $\boldsymbol{T}^{\cap}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$ telle que :

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})[\boldsymbol{\alpha}_{\mathbf{0}}^{\cap}(t^{\cap}) \stackrel{\text{def}}{=} \bigcup_{t \in t^{\cap}} \boldsymbol{\alpha}_{\mathbf{0}}(t)] \tag{7.14}$$

Finalement, comme pour les TUP, l'ensemble des PosAOpt d'un TUC est défini comme étant la différence entre l'ensemble des PosA et les ensembles de PosAObl et de PosAInt :

Définition 45 (PosAOpt d'un TUC)**.** L'*ensemble des PosAOpt d'un TUC* est défini par une application $\boldsymbol{\alpha}_{\mathbf{?}}^{\cap}$ de $\boldsymbol{T}^{\cap}$ vers $2^{\boldsymbol{S}_{\mathscr{T}}}$ telle que :

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})[\boldsymbol{\alpha}_{\mathbf{?}}^{\cap}(t^{\cap}) \stackrel{\text{def}}{=} \boldsymbol{\alpha}^{\cap}(t^{\cap}) - \boldsymbol{\alpha}_{\mathbf{1}}^{\cap}(t^{\cap}) - \boldsymbol{\alpha}_{\mathbf{0}}^{\cap}(t^{\cap})] \tag{7.15}$$

La définition des signatures des TUP est également naturellement étendue au TUC. Cependant, pour les PosA *eaten* de $t^{\cap} = \{/\text{to eat}\backslash, /\text{quick}\backslash\}$ par exemple, nous ne devons considérer l'union des signatures que sur l'ensemble des TUP qui possèdent effectivement la PosA *eaten* : $\{t \in t^{\cap} \mid \mathit{eater} \in \boldsymbol{\alpha}(t)\}$:

Définition 46 (Signature d'un TUC)**.** L'ensemble des signatures des TUC $\{\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}\}_{t^{\cap} \in \boldsymbol{T}^{\cap}}$ est un ensemble de fonctions de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{T}^{\cap}$. Pour tout TUC $t^{\cap}$, $\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}$ est une fonction avec $\mathit{domaine}(\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}) = \boldsymbol{\alpha}^{\cap}(t^{\cap})$ telle que pour tout s dans $\boldsymbol{\alpha}^{\cap}(t^{\cap})$,

$$\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s) \stackrel{\text{def}}{=} \bigcup_{t \in t^{\cap} \mid s \in \boldsymbol{\alpha}(t)} \boldsymbol{\varsigma}_{t}(s) \tag{7.16}$$

En conséquence directe, la structure actancielle d'un TUC primitif (i.e., un singleton) est exactement la structure actancielle de son seul élément :

Proposition 7.2.1. *Pour tout $t \in \mathbf{T}$, chacun des points suivants est vrai :*

- $\boldsymbol{\alpha}^{\cap}(\{t\}) = \boldsymbol{\alpha}(t)$.
- $\boldsymbol{\alpha}_{\boldsymbol{1}}^{\cap}(\{t\}) = \boldsymbol{\alpha}_{\boldsymbol{1}}(t)$.
- $\boldsymbol{\alpha}_{\boldsymbol{0}}^{\cap}(\{t\}) = \boldsymbol{\alpha}_{\boldsymbol{0}}(t)$.
- $\boldsymbol{\alpha}_{\boldsymbol{?}}^{\cap}(\{t\}) = \boldsymbol{\alpha}_{\boldsymbol{?}}(t)$.
- *pour tout $s \in \boldsymbol{\alpha}(t)$, $\boldsymbol{\varsigma}_{\{t\}}^{\cap}(s) = \boldsymbol{\varsigma}_{t}(s)$*

Preuve: voir annexe, p. 305.

Visualisation. En complément de la figure 3.10, la figure 7.12 illustre deux visualisations équivalentes du TUC $t^{\cap} = \{/\text{to eat}\backslash, /\text{quick}\backslash\}$ et de sa structure actancielle :

- La PosA *eaten* peut être illustrée dans la partie inférieure de la boîte (à gauche).
- Puisque les TUC ont une représentation graphique, et puisque la signature d'un TUC pour une PosA est un TUC, la PosA *eaten* peut être illustrée comme un lien d'association étiqueté *eaten* qui lie la boîte de $t^{\cap}$ et celle qui représente la signature de $t^{\cap}$ pour *eaten* (à droite).

La seconde représentation graphique permettrait par exemple de montrer que le TUC en signature possède lui-même une structure actancielle.

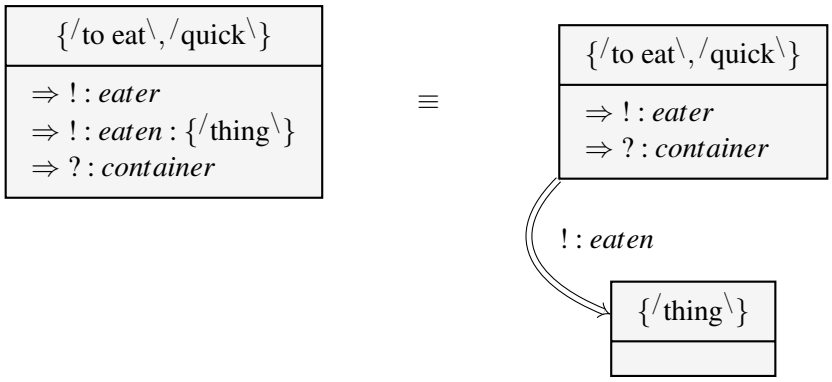


FIGURE 7.12 – Visualisation de la structure actancielle du TUC {/to eat\,/quick\}

7.2.3 Hiérarchisation des TUC

Nous étendons la relation de spécialisation des TUP aux TUC, afin de permettre leur organisation en une hiérarchie. La relation de spécialisation, définie dans la section 7.2.3.1, est cependant légèrement plus complexe que l'extension naturelle d'un préordre sur un ensemble à un préordre sur l'ensemble de ses parties. Nous en tirons l'avantage de définir les TUC universels et absurdes plus naturellement que pour les TUP (§7.2.3.2 et §7.2.3.3). Mais surtout, redéfinir ainsi le préordre de spécialisation facilitera la construction d'une sémantique d'interprétation pour le formalisme des Graphes d'Unités, comme le justifie la section 7.2.3.4.

7.2.3.1 Préordonnancement

Définissons un préordre de spécialisation sur $\boldsymbol{T}^{\cap}$, par exemple, {/happy\,/cat\} $\overset{\cap}{\lesssim}$ {/hasMood\,/animal\} signifie que {/happy\,/cat\} est plus spécifique que {/hasMood\,/animal\}.

Définition 47 (Préordre sur $\boldsymbol{T}^{\cap}$)**.** L'ensemble des TUC $\boldsymbol{T}^{\cap}$ est préordonné par une relation $\overset{\cap}{\lesssim}$, qui est le plus petit préordre tel que chacun des points ci-dessous est vrai :

– $\overset{\cap}{\lesssim}$ contient l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$, i.e.,

$$(\forall t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\cap}) \Big[(\forall t_y \in t_y^{\cap}) \big[(\exists t_x \in t_x^{\cap}) [t_x \lesssim t_y \implies t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}] \big] \Big] \tag{7.17}$$

– $\top^{\cap}$ est un élément maximal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, i.e.,

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap}) [t^{\cap} \overset{\cap}{\lesssim} \top^{\cap}] \tag{7.18}$$

– tout TUC déclaré absurde est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, i.e.,

$$(\forall t_a^{\cap} \in \bot_A^{\sqcap}) (\forall t^{\cap} \in \boldsymbol{T}^{\cap}) [t_a^{\cap} \overset{\cap}{\lesssim} t^{\cap}] \tag{7.19}$$

– pour toute SRelA, le TUC composé de ses racines de PosAObl et de PosAInt est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ (et nous verrons qu'il est donc absurde), c'est-à-dire que :

$$(\forall s \in \boldsymbol{S}_{\mathscr{T}}) (\forall t^{\cap} \in \boldsymbol{T}^{\cap}) [\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\} \overset{\cap}{\lesssim} t^{\cap}] \tag{7.20}$$

– si la signature d'un TUC pour une PosA obligatoire donnée est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, alors ce TUC est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ (et nous verrons qu'il est donc absurde), i.e.,

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})\Big[(\exists s \in \boldsymbol{\alpha}_{\mathbf{1}}^{\cap}(t^{\cap}))\big[(\forall t_x^{\cap} \in \boldsymbol{T}^{\cap})[\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s) \overset{\cap}{\lesssim} t_x^{\cap}]\big] \implies (\forall t_x^{\cap} \in \boldsymbol{T}^{\cap})[t^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}]\Big] \tag{7.21}$$

Introduisons la relation d'équivalence naturelle $\overset{\cap}{\simeq}$, définie par :

$$t_x^{\cap} \overset{\cap}{\simeq} t_y^{\cap} \iff \Big(t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \text{ et } t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}\Big) \tag{7.22}$$

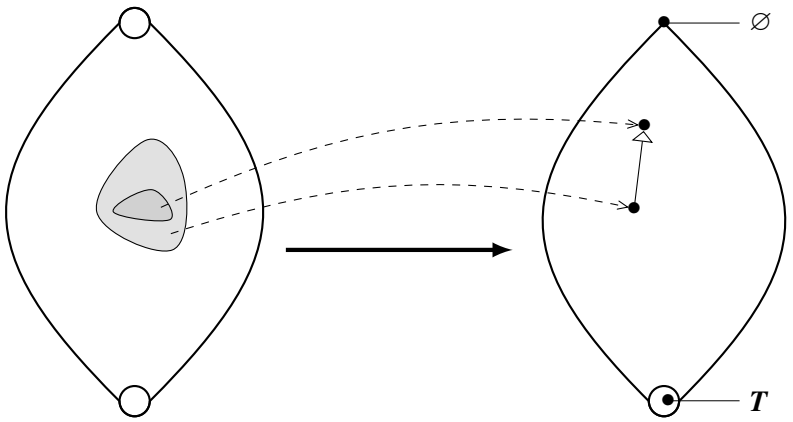
La figure 7.13 illustre l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$. Un résultat intéressant concernant cette extension naturelle, est qu'elle contient la relation d'inclusion inverse sur les ensembles. Dans la théorie mathématique des catégories, les ensembles préordonnés ont une catégorie notée **Ord** dont les fonctions monotones sont les homomorphismes. Dans notre cas, nous avons deux ensembles préordonnés : l'ensemble des parties de $\boldsymbol{T}$ muni de la relation d'inclusion $(\boldsymbol{T}^{\cap}, \subseteq)$, et l'ensemble des parties de $\boldsymbol{T}$ muni de la relation de spécialisation $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$. Le morphisme f qui transforme l'un en l'autre et tel que $f(t^{\cap}) = t^{\cap}$ est anti-monotone :

Proposition 7.2.2. *L'application* $f : (\mathbf{T}^{\cap}, \subseteq) \longrightarrow (\mathbf{T}^{\cap}, \overset{\cap}{\lesssim})$ *telle que* $f(t^{\cap}) = t^{\cap}$ *est un homomorphisme anti-monotone de préordre, i.e.,*

$$(\forall t_x^{\cap}, t_y^{\cap} \in \mathbf{T}^{\cap})[t_x^{\cap} \subseteq t_y^{\cap} \implies t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}] \tag{7.23}$$

Preuve: voir annexe, p. 305.

La figure 7.13 illustre la proposition 7.2.2. Supposons que $t_x^{\cap} = \{/\text{little}\backslash, /\text{cat}\backslash\}$, et $t_y^{\cap} = \{/\text{cute}\backslash, /\text{little}\backslash, /\text{cat}\backslash\}$. $t_x^{\cap}$ est une partie de $t_y^{\cap}$, donc $t_y^{\cap}$ est une spécialisation de $t_x^{\cap}$.



$(\boldsymbol{T}, \lesssim)$ $\qquad$ $\boldsymbol{T}^{\cap}$ muni de l'extension naturelle du préordre $\boldsymbol{T}$

FIGURE 7.13 – Illustration de l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$; illustration de la proposition 7.2.2 : si $t_x^{\cap}$ est inclus dans $t_y^{\cap}$, alors $t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}$.

Il s'ensuit par la proposition 7.2.2 que $\varnothing$ et $\boldsymbol{T}$ sont respectivement un élément maximal et un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.

Proposition 7.2.3. *$\varnothing$ et $\mathbf{T}$ sont respectivement un élément maximal et un élément minimal de* $(\mathbf{T}^{\cap}, \overset{\cap}{\lesssim})$.

Preuve: voir annexe, p. 305.

Le préordre $\overset{\cap}{\lesssim}$ est également défini de sorte que la base (les TUC les plus spécifiques) de l'ensemble préordonné $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ est aplatie. La figure 7.14 représente l'ensemble préordonné $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ avec les éléments maximaux et minimaux connus. Le cercle bleu du dessus représente l'ensemble des éléments maximaux. L'ellipse rouge du dessous contient l'ensemble des éléments minimaux.

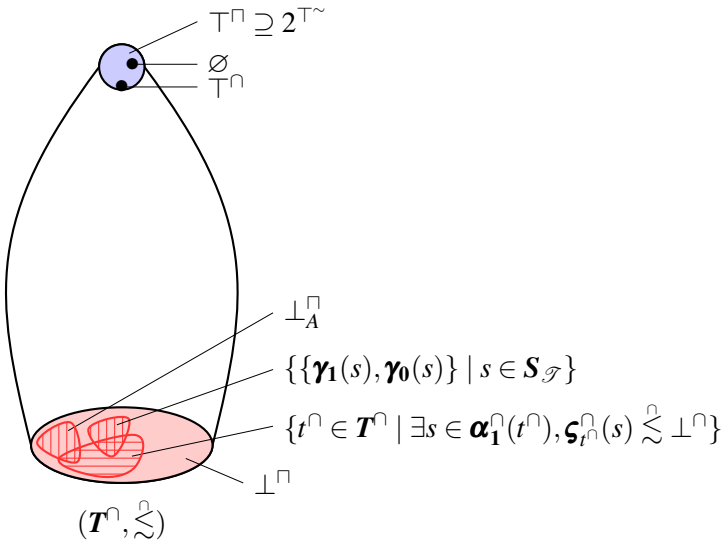


FIGURE 7.14 – Illustration des éléments extrémaux connus du préordre $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.

7.2.3.2 TUC universels

L'ensemble des éléments maximaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ contient les TUC les plus génériques, chacun étant le type de toute unité. Il forme ainsi l'ensemble des *TUC universels*.

Définition 48 (Ensemble des TUC universels)**.** L'ensemble des TUC universels est noté $\top^{\sqcap}$ et est l'ensemble des éléments maximaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$:

$$\top^{\sqcap} \overset{\text{def}}{=} \left\{ t^{\cap} \in \boldsymbol{T}^{\cap} \mid \left(\forall t_x^{\cap} \in \boldsymbol{T}^{\cap}, t_x^{\cap} \overset{\cap}{\lesssim} t^{\cap} \right) \right\} \tag{7.24}$$

Le préordre $\overset{\cap}{\lesssim}$ est défini de sorte que le sommet (les TUC les plus génériques) de l'ensemble préordonné $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ est aplati. Ainsi $\varnothing$ et $\top^{\cap}$ sont tous deux des éléments maximaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.

Plus généralement, tout sous-ensemble de $\top^{\sim}$ est un élément maximal de $\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim}$.

Proposition 7.2.4. *Tout sous-ensemble de $\top^{\sim}$ est un élément maximal de $(\mathbf{T}^{\cap}, \overset{\cap}{\lesssim})$, parmi lesquels $\varnothing$ et $\top^{\cap}$. i.e.,*

$$(\forall t^{\cap} \subseteq \top^{\sim})(\forall t_x^{\cap} \in \mathbf{T}^{\cap})[t_x^{\cap} \overset{\cap}{\lesssim} t^{\cap}]$$

Preuve: voir annexe, p. 305.

La proposition 7.2.4 est illustrée par la figure 7.14. L'ensemble des TUC universels peut également être écrit comme suit.

Proposition 7.2.5.

$$\top^{\sqcap} = \{t^{\cap} \mid t^{\cap} \overset{\cap}{\cong} \top^{\cap}\}$$

Preuve: voir annexe, p. 305.

7.2.3.3 TUC absurdes

L'ensemble des éléments minimaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ contient les TUC les plus spécifiques, chacun n'étant le type d'aucune unité. Il forme ainsi l'ensemble des *TUC absurdes*.

Définition 49 (Ensemble des TUC absurdes)**.** L'ensemble des TUC absurdes est noté $\bot^{\sqcap}$ et est l'ensemble des éléments minimaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$:

$$\bot^{\sqcap} \overset{\text{def}}{=} \left\{ t^{\cap} \in \boldsymbol{T}^{\cap} \mid \left(\forall t_x^{\cap} \in \boldsymbol{T}^{\cap}, t^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}\right) \right\} \tag{7.25}$$

Ainsi, la propriété d'être absurde est héréditaire : si un TUC $t^{\cap}$ est absurde, alors tous les TUC inférieurs sont également absurdes. Par exemple si $\{def, indef\}$ est déclaré absurde, et que $\ulcorner \text{USA} \urcorner \lesssim def$, alors $\{\ulcorner \text{USA} \urcorner, indef\}$ est absurde.

La proposition suivante liste des TUC absurdes remarquables.

Proposition 7.2.6. *Tous les TUC suivants sont absurdes :*

1. $\mathbf{T}$ *;*
2. *tout sous-ensemble non-vide de l'ensemble des TUP absurdes, i.e., tel que $t^{\cap} \subseteq A^{\sim}$ (cf., définition 39), parmi lesquels $\bot^{\cap}$, et tout sous-ensemble non-vide de $\bot^{\sim}$.*
3. *tout $t^{\cap} \in \mathbf{T}^{\cap}$ naturellement plus spécifique que $\bot^{\cap}$, i.e., tel que $\exists t \in t^{\cap}, t \lesssim \bot$;*
4. *tout $t^{\cap} \in \mathbf{T}^{\cap}$ qui est naturellement plus spécifique que l'un des TUC déclarés absurdes, i.e., $\exists t_a^{\cap} \in \bot^{\sqcap}$ tel que $\forall t_a \in t_a^{\cap}, \exists t \in t^{\cap}, t \lesssim t_a$;*
5. *tout $t^{\cap} \in \mathbf{T}^{\cap}$ qui possède une PosA à la fois obligatoire et interdite, i.e., tel que $\boldsymbol{\alpha}_{\boldsymbol{1}}^{\cap}(t^{\cap}) \cap \boldsymbol{\alpha}_{\boldsymbol{0}}^{\cap}(t^{\cap}) \neq \varnothing$;*
6. *tout $t^{\cap} \in \mathbf{T}^{\cap}$ qui possède une signature absurde, i.e., tel que $\exists s \in \mathbf{S}_{\mathscr{T}}$ tel que $\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s) \in \bot^{\sqcap}$.*

Preuve: voir annexe, p. 305.

L'ensemble des TUC absurdes peut également être écrit de la manière suivante.

Proposition 7.2.7.

$$\bot^{\cap} = \{t^{\cap} \mid t^{\cap} \cong \bot^{\cap}\}$$

Preuve: voir annexe, p. 306.

Comme on peut le remarquer, la définition du caractère absurde d'un TUC est récursive, puisqu'elle implique potentiellement le caractère absurde d'un TUC en signature. La section 7.3.2 introduira la condition nécessaire et suffisante pour qu'un TUC soit absurde.

7.2.3.4 Avantages de la redéfinition du préordre de spécialisation

Dans le reste de cette section, nous discutons de l'avantage d'utiliser le préordre $\overset{\cap}{\lesssim}$ par rapport à l'extension naturelle du préordre $\lesssim$. En particulier, nous précisons les propriétés intéressantes que la définition 47 apporte aux TUC, et qui ne sont pas possibles avec les TUP.

Supposons que nous définissions une sémantique d'interprétation naïve pour les TUP et les instances d'unité.

Soit **D** le domaine, qui représente l'ensemble des unités, et associons aux TUP l'interprétation triviale d'un ensemble, i.e.,

$$(\forall t \in \boldsymbol{T})[t^{\mathbf{I}} \subseteq \mathbf{D}]$$

Associons au préordre $\lesssim$ sur $\boldsymbol{T}$ l'interprétation de la relation d'inclusion sur les interprétations de $\boldsymbol{T}$, i.e.,

$$(\forall t_x, t_y \in \boldsymbol{T})[t_x \lesssim t_y \implies t_x^{\mathbf{I}} \subseteq t_y^{\mathbf{I}}]$$

Seul le TUP absurde premier est par définition le type d'aucune unité (cf., définition 29), $\bot^{\mathbf{I}} = \varnothing$.
Une telle interprétation présente deux limites importantes :

- Tout d'abord, bien qu'un TUP t ayant une PosA s à la fois obligatoire et interdite soit défini comme étant absurde (cf., définition 39), nous ne pouvons pas prédire simplement que t possède une interprétation vide s'il n'appartient pas à l'ensemble $\bot^{\sim}$. Ce cas est illustré dans la partie gauche de la figure 7.15a.
- Ensuite, si la signature d'un TUP t est absurde pour une PosAObl donnée s (par exemple $\boldsymbol{\varsigma}_t(s) = \{\bot\}$), alors t ne rentre pas dans la définition 39 des TUP absurdes. Cependant, son interprétation devrait être vide également. Ce cas est illustré dans la partie gauche de la figure 7.15b.

Maintenant, considérons une sémantique d'interprétation naïve similaire pour les TUC et les unités. **D** représente toujours l'ensemble des unités, mais nous associons maintenant aux TUC l'interprétation triviale d'un ensemble, i.e.,

$$(\forall t^{\cap} \in \boldsymbol{T}^{\cap})[t^{\cap \mathbf{I}} \subseteq \mathbf{D}]$$

Associons au préordre $\overset{\cap}{\lesssim}$ sur $\boldsymbol{T}^{\cap}$ l'interprétation de la relation d'inclusion sur les interprétations de $\boldsymbol{T}^{\cap}$, i.e.,

$$(\forall t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\cap})[t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \implies t_x^{\cap\mathbf{I}} \subseteq t_y^{\cap\mathbf{I}}]$$

Bien que le TUC absurde premier soit le seul TUC défini comme étant le type d'aucune unité, i.e., $\bot^{\cap\mathbf{I}} = \varnothing$, maintenant :

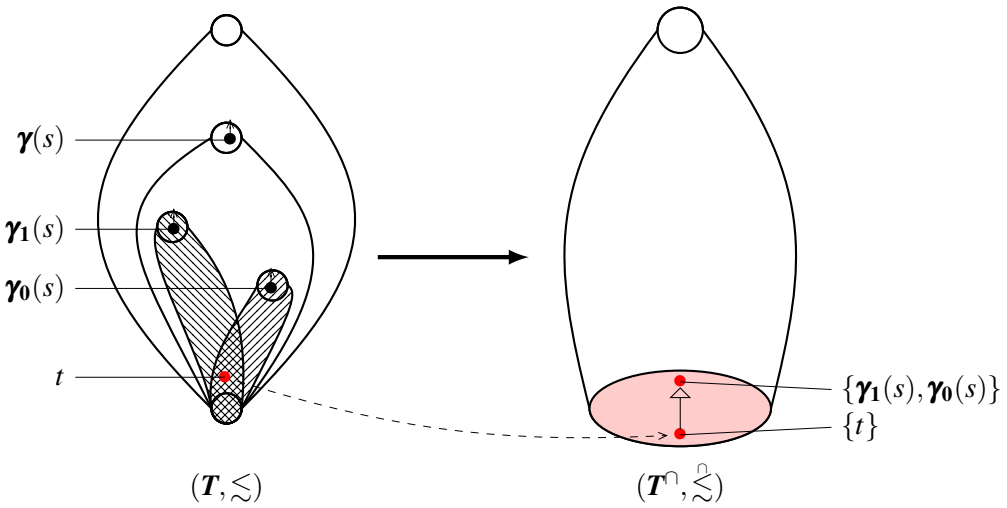
- Tout TUC $t^{\cap}$ qui possède une PosA s à la fois obligatoire et interdite est un élément minimal de $\boldsymbol{T}^{\cap}$, donc est absurde (cf., définitions 47 et 49). Donc $t^{\cap} \overset{\cap}{\lesssim} \bot^{\cap}$, et $t^{\cap\mathbf{I}} \subseteq \bot^{\cap\boldsymbol{I}} = \varnothing$. Le cas simple où $t^{\cap}$ est primitif est illustré sur la figure 7.15a.
- De plus, si la signature d'un TUC $t^{\cap}$ est absurde pour une PosAObl donnée s, alors la proposition 7.2.6 montre que $t^{\cap}$ est également un élément minimal de $\boldsymbol{T}^{\cap}$. Donc $t^{\cap} \overset{\cap}{\lesssim} \bot^{\cap}$, et $t^{\cap\mathbf{I}} \subseteq \bot^{\cap\boldsymbol{I}} = \varnothing$. Le cas simple où $t^{\cap}$ est primitif et $\boldsymbol{\varsigma}^{\cap}_{t^{\cap}}(s) = \bot^{\cap}$ est illustré sur la figure 7.15b.

Illustrons le second point avec un exemple un peu plus élaboré. Considérons par exemple que l'on souhaite définir /pleuvoir-grêler\ comme étant une spécialisation d'à la fois /pleuvoir\ et /grêler\. Supposons que ces deux TUSemP héritent d'une PosA *tombe* d'un parent commun /tomber\. Par la définition 38, /pleuvoir-grêler\ possède la PosA *tombe*, dont la signature est l'union de celles de /pleuvoir\ et /grêler\ :

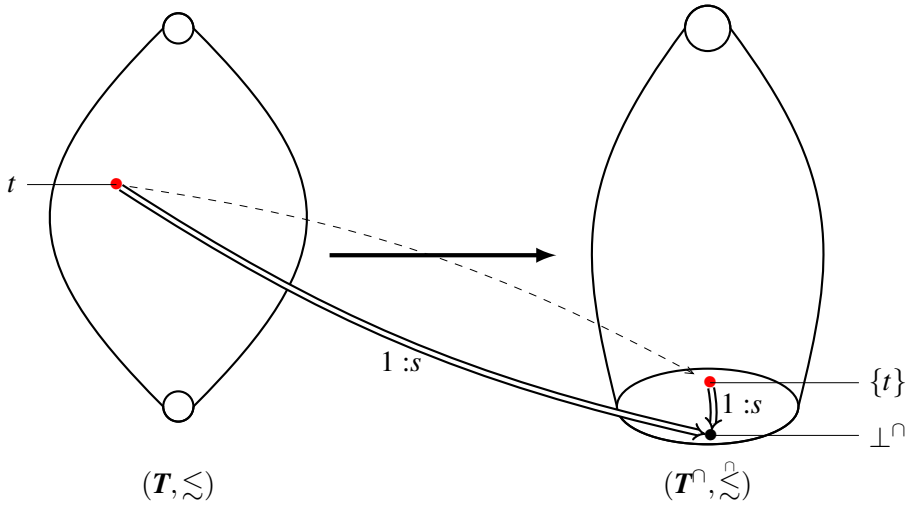
$$\begin{aligned}\boldsymbol{\varsigma}_{/\text{pleuvoir-grêler}\backslash}(tombe) &= \boldsymbol{\varsigma}_{/\text{pleuvoir}\backslash}(tombe) \cup \boldsymbol{\varsigma}_{/\text{grêler}\backslash}(tombe)\\ &= \{/\text{eau}\backslash, /\text{liquide}\backslash\} \cup \{/\text{eau}\backslash, /\text{solide}\backslash\}\\ &= \{/\text{eau}\backslash, /\text{liquide}\backslash, /\text{solide}\backslash\}\end{aligned}$$

Si l'on déclare le TUC $\{/\text{liquide}\backslash, /\text{solide}\backslash\}$ comme étant absurde, i.e., $\{/\text{liquide}\backslash, /\text{solide}\backslash\} \in \bot^{\cap}_{A}$, et puisque l'absurdité est héritée, alors la signature du TUP /pluie-grêle\ est absurde. Cependant, le TUP /pluie-grêle\ n'est pas automatiquement absurde. La définition de $\overset{\cap}{\lesssim}$ permet par contre au TUC $\{/\text{pleuvoir-grêler}\backslash\}$ d'être absurde[3]. La rigueur du formalisme des Graphes d'Unités permet ainsi de motiver l'introduction d'un TUSemP /tomber de l'eau\ par exemple, qui serait plus spécifique que /tomber\, et le parent direct de /pleuvoir\ et /grêler\, et dont la signature pour *tombe* serait $\{/\text{eau}\backslash\}$.

3. Les expressions telles que *pleuvoir des cordes* sont idiomatiques, et devraient être traitées différemment. Leur étude est en dehors du cadre d'étude de cette thèse.



(a) Bien que le TUP t ait une PosA à la fois obligatoire et interdite, son interprétation naïve n'est pas obligatoirement vide. Cette limite est levée par l'utilisation des TUC et du préordre $\overset{\cap}{\lesssim}$.



(b) Bien que le TUP t ait une signature absurde, son interprétation naïve n'est pas obligatoirement vide. Cette limite est levée par l'utilisation des TUC et du préordre $\overset{\cap}{\lesssim}$.

FIGURE 7.15 – Illustration des bénéfices d'utiliser les TUC et la définition du préordre $\overset{\cap}{\lesssim}$.

7.2.4 Définition de la hiérarchie des types d'unité

Nous pouvons maintenant introduire la définition de la hiérarchie des types d'unité, qui est au cœur du formalisme des GU.

Définition 50 (Hiérarchie des types d'unité)**.** Une *hiérarchie des types d'unité* $\mathscr{T} = \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Gamma_1}, \boldsymbol{\gamma_1}, \boldsymbol{\Gamma_0}, \boldsymbol{\gamma_0}, C_A, \bot_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}} \rangle$, est composée de :

- T_D un ensemble de TUP déclarés ;
- $\boldsymbol{S}_{\mathscr{T}}$ l'ensemble des SRelA ;
- $\boldsymbol{\Gamma}$ l'ensemble des racines de PosA ;
- $\boldsymbol{\gamma}$ la bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma}$;
- $\boldsymbol{\Gamma_1}$ l'ensemble des racines de PosAObl ;
- $\boldsymbol{\gamma_1}$ la bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma_1}$;
- $\boldsymbol{\Gamma_0}$ l'ensemble des racines de PosAInt ;
- $\boldsymbol{\gamma_0}$ la bijection de $\boldsymbol{S}_{\mathscr{T}}$ vers $\boldsymbol{\Gamma_0}$;
- C_A un ensemble de comparaisons déclarées de TUP ;
- $\bot_A^{\sqcap}$ un ensemble de TUC déclarés absurdes ;
- $\{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}}$ l'ensemble des signatures des TUP ;

La hiérarchie des types d'unité est l'ensemble minimal d'objets mathématiques nécessaires à la formation de la base cohérente du formalisme des GU : un ensemble préordonné de TUC munis de structures actancielles.

Notons le cas dégénéré suivant pour la hiérarchie des types d'unité : supposons que l'on déclare $\top^{\cap} \in \bot_A^{\sqcap}$. Alors la hiérarchie complète s'effondre, et $\bot^{\sqcap} = \boldsymbol{T}^{\cap}$. Le résultat est le même s'il existe un TUC $t^{\cap} \in \boldsymbol{T}^{\cap}$ tel que $\top^{\cap} \overset{\cap}{\lesssim} t^{\cap}$, et $s \in \boldsymbol{\alpha}^{\cap}(t^{\cap})$ tel que $\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s) \overset{\cap}{\lesssim} \bot^{\cap}$. Alors, par la définition 47, $t^{\cap} \overset{\cap}{\lesssim} \bot^{\cap}$, et par transitivité du préordre, $\bot^{\sqcap} = \boldsymbol{T}^{\cap}$. Ces situations impliquent que la hiérarchie des types d'unité est inconsistante, et peuvent plus facilement être détectées dans le formalisme des GU que dans la modélisation proposée avec le formalisme des GC.

7.3 Caractérisation des TUC

Maintenant que nous avons défini l'ensemble préordonné des TUC munis de structures actancielles, cette section étudie les aspects remarquables des TUC.

La définition du préordre $\overset{\cap}{\lesssim}$ sur $\boldsymbol{T}^{\cap}$ est implicite. Afin de faciliter les démonstrations suivantes, la section 7.3.1 introduit tout d'abord un processus de construction itérative pour $\overset{\cap}{\lesssim}$.

Ensuite, la section 7.3.2 précise la condition nécessaire et suffisante pour qu'un TUC soit : i) universel, ii) absurde, ou iii) comparable à un autre TUC.

Une fois que le préordre $\overset{\cap}{\lesssim}$ aura été caractérisé, la section 7.3.3 montre que les propriétés intéressantes de la structure actancielle des TUP sont préservées pour la plupart des TUC dans la hiérarchie des types d'unité.

La section 7.3.4 caractérise la relation d'équivalence naturelle sur l'ensemble des TUC, et étudie le sous-ensemble remarquable des *TUC clos* : ceux qui possèdent une cardinalité maximale au sein de leur classe d'équivalence.

7.3.1 Construction itérative du préordre sur les TUC

Afin de faciliter les démonstrations suivantes, nous introduisons tout d'abord un processus de construction itérative du préordre $\overset{\cap}{\lesssim}$ sur $\boldsymbol{T}^{\cap}$. La définition 51 introduit un processus de construction itérative d'un ensemble de comparaisons de TUC, dont l'intuition est détaillée juste après. Le résultat principal de cette section est le théorème 7.3.3, qui précise que la suite des ensembles des comparaisons de TUC converge vers $\overset{\cap}{\lesssim}$.

Définition 51 (Ensemble des comparaisons de TUC)**.** Soit $(\boldsymbol{C}_n^{\cap})_{n\in\mathbb{N}}$ une suite d'ensembles de comparaisons sur $\boldsymbol{T}^{\cap}$, i.e., pour tout $n \in \mathbb{N}$, $\boldsymbol{C}_n^{\cap} \subseteq \boldsymbol{T}^{\cap 2}$, défini par :

– pour $i = 0$, $\boldsymbol{C}_0^{\cap} \overset{\text{def}}{=} \boldsymbol{C}_{\overset{\cap}{\lesssim}}^{\cap} \cup \boldsymbol{C}_{\top}^{\cap} \cup \boldsymbol{C}_{\bot}^{\cap} \cup \boldsymbol{C}_{\boldsymbol{\Gamma}}^{\cap}$, où :

$$\boldsymbol{C}_{\overset{\cap}{\lesssim}}^{\cap} \overset{\text{def}}{=} \{(t_y^{\cap}, t_x^{\cap}) \in \boldsymbol{T}^{\cap 2} \mid (\forall t_y \in t_y^{\cap})(\exists t_x \in t_x^{\cap})[t_x \lesssim t_y]\} \tag{7.26}$$

$$\boldsymbol{C}_{\top}^{\cap} \overset{\text{def}}{=} \{(\top^{\cap}, \varnothing)\} \tag{7.27}$$

$$\boldsymbol{C}_{\bot}^{\cap} \overset{\text{def}}{=} \{(\bot^{\cap}, t^{\cap}) \in \boldsymbol{T}^{\cap 2} \mid t^{\cap} \in \bot_A^{\cap}\} \tag{7.28}$$

$$\boldsymbol{C}_{\boldsymbol{\Gamma}}^{\cap} \overset{\text{def}}{=} \{(\bot^{\cap}, \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\})\}_{s\in\boldsymbol{S}_{\mathscr{T}}} \tag{7.29}$$

– pour tout $i > 0$, $\boldsymbol{C}_i^{\cap} \overset{\text{def}}{=} \boldsymbol{C}_{i-1}^{\cap} \cup \boldsymbol{C}_{\boldsymbol{\varsigma}_i}^{\cap} \cup \boldsymbol{C}_{+i}^{\cap}$ où :

$$\boldsymbol{C}_{\boldsymbol{\varsigma}_i}^{\cap} \overset{\text{def}}{=} \{(\bot^{\cap}, t^{\cap}) \in \boldsymbol{T}^{\cap 2} \mid (\exists s \in \boldsymbol{\alpha_1}^{\cap}(t^{\cap}))[(\bot^{\cap}, \boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s)) \in \boldsymbol{C}_{i-1}^{\cap}]\} \tag{7.30}$$

$$\boldsymbol{C}_{+i}^{\cap} \overset{\text{def}}{=} \{(t_z^{\cap}, t_x^{\cap}) \in \boldsymbol{T}^{\cap 2} \mid \exists t_y^{\cap} \in \boldsymbol{T}^{\cap}, (t_z^{\cap}, t_y^{\cap}) \in \boldsymbol{C}_{i-1}^{\cap} \text{ et } (t_y^{\cap}, t_x^{\cap}) \in \boldsymbol{C}_{i-1}^{\cap}\} \tag{7.31}$$

La suite $(\boldsymbol{C}_n^{\cap})$ est une suite croissante monotone bornée, i.e., pour tout $n \in \mathbb{N}$, $\boldsymbol{C}_n^{\cap} \subseteq \boldsymbol{C}_{n+1}^{\cap} \subseteq \boldsymbol{T}^{\cap 2}$, donc elle converge. La limite de la suite $(\boldsymbol{C}_n^{\cap})_{n\in\mathbb{N}}$ est notée $\boldsymbol{C}^{\cap}$, l'*ensemble des comparaisons de TUC*.

L'intuition derrière ce processus de construction itérative est le suivant :

– $\boldsymbol{C}_{\overset{\cap}{\lesssim}}^{\cap}$ représente l'extension naturelle d'un préordre sur un ensemble à un préordre sur l'ensemble des parties de cet ensemble ;

– $\boldsymbol{C}_{\top}^{\cap}$ est introduit pour aplatir le sommet de l'ensemble des parties de $\boldsymbol{T}$ de sorte que $\top^{\cap}$ soit maximal ;

- $\boldsymbol{C}^{\cap}_{\perp}$ est introduit pour aplatir la base de l'ensemble des parties de $\boldsymbol{T}$ de sorte que tout TUC déclaré absurde soit minimal ;
- $\boldsymbol{C}^{\cap}_{\boldsymbol{\Gamma}}$ représente le fait que pour tout SRelA les racines des PosAObl et PosAInt s sont incompatibles ;
- $\boldsymbol{C}^{\cap}_{\boldsymbol{\varsigma}_i}$ représente le fait que si la signature d'un TUC pour une PosAObl est minimale, alors ce TUC est minimal ;
- $\boldsymbol{C}^{\cap}_{+i}$ ferme transitivement l'ensemble des comparaisons $\boldsymbol{C}^{\cap}$.

Les deux lemmes suivants facilitent la preuve du théorème principal de cette section : l'ensemble des comparaisons de TUC est égal à $\overset{\cap}{\lesssim}$ (théorème 7.3.3).

Lemme 7.3.1. *L'ensemble des comparaisons de TUC* $\mathbf{C}^{\cap}$ *définit un préordre sur* $\mathbf{T}^{\cap}$.

Preuve: voir annexe, p. 306.

Lemme 7.3.2. *L'ensemble des comparaisons de TUC* $\mathbf{C}^{\cap}$ *est tel que toutes les équations de la définition 47 sont satisfaites.*

Preuve: voir annexe, p. 306.

Finalement nous pouvons montrer que l'ensemble des comparaisons de TUC est le plus petit préordre tel que toutes les équations de la définition 47 sont vraies, d'où le théorème suivant :

Théorème 7.3.3. *L'ensemble des comparaisons de TUC* $\mathbf{C}^{\cap}$ *est égal à la relation de préordre* $\overset{\cap}{\lesssim}$ *sur* $\mathbf{T}^{\cap}$*, i.e,*

$$(\forall t^{\cap}_x, t^{\cap}_y \in \mathbf{T}^{\cap})[t^{\cap}_x \overset{\cap}{\lesssim} t^{\cap}_y \iff (t^{\cap}_y, t^{\cap}_x) \in \mathbf{C}^{\cap}]$$

□

Preuve: voir annexe, p. 307.

7.3.2 Conditions de comparaison des TUC

Le préordre $\overset{\cap}{\lesssim}$ sur l'ensemble des TUC est plus complexe que la simple extension d'un préordre sur un ensemble à un préordre sur l'ensemble des parties de cet ensemble. Dans cette section, nous utilisons le processus de construction itérative du préordre $\overset{\cap}{\lesssim}$ pour introduire trois conditions nécessaires et suffisantes importantes :

- pour qu'un TUC soit universel (§7.3.2).
- pour qu'un TUC soit absurde (§7.3.2).
- pour qu'un TUC soit comparable à un autre TUC (§7.3.2).

Condition nécessaire et suffisante pour qu'un TUC soit universel.

Proposition 7.3.4. *$t^{\cap}$ est universel si et seulement si l'une des deux conditions suivantes est vraie :*

- $t^{\cap} \subseteq \top^{\sim}$.
- $\top^{\cap} \overset{\cap}{\lesssim} \perp^{\cap}$*, i.e., la hiérarchie des types d'unité est effondrée en une unique classe d'équivalence.*

Preuve: voir annexe, p. 308.

Condition nécessaire et suffisante pour qu'un TUC soit absurde. La définition des TUC est récursive, puisqu'un TUC est absurde s'il possède une signature absurde. Dans cette section, nous proposons une condition nécessaire et suffisante non récursive pour qu'un TUC soit absurde.

Proposition 7.3.5. *$t_0^\cap$ est absurde si et seulement s'il existe une liste de TUC $(t_0^\cap,\dots,t_n^\cap)$, avec $0 \leq n$, telle que chacun des points ci-dessous est vrai :*

1. *pour tout $0 \leq i < n$, l'un des points suivants est vrai,*
   a. $(t_{i+1}^\cap, t_i^\cap) \in \mathbf{C}_i^\cap$ *(propagation par héritage) ;*
   b. $\exists s_i \in \boldsymbol{\alpha_1^\cap}(t_i^\cap), \boldsymbol{\varsigma}_{t_i^\cap}^\cap(s_i) = t_{i+1}^\cap$ *(propagation par signature de PosAObl) ;*
2. *l'un des points suivants est vrai*
   a. $\exists t \in t_n^\cap, t \lesssim \bot$;
   b. $t_n^\cap \in \bot_A^\sqcap$;
   c. $\exists s \in \mathbf{S}_\mathscr{T}, t_n^\cap = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$.

Preuve: voir annexe, p. 308.

La figure 7.16 illustre la double propagation du caractère absurde des TUC.

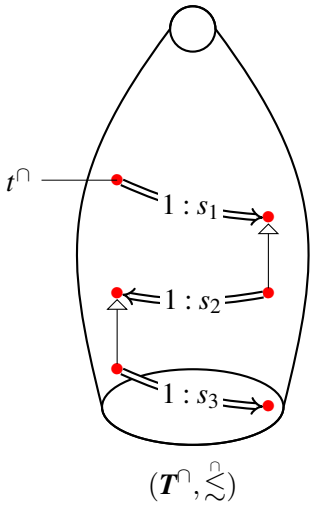


FIGURE 7.16 – Illustration de la condition pour qu'un TUC soit absurde. Dans ce cas, chacun des TUC représentés serait absurde.

Condition nécessaire et suffisante pour comparer deux TUC. La condition nécessaire et suffisante pour que deux TUC $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$ soient comparables peut être décrite comme suit :

$t_x^\cap \overset{\cap}{\lesssim} t_y^\cap$ si et seulement si au moins l'un des cas suivants est vrai :
- $t_x^\cap$ est absurde ;
- $t_x^\cap$ est l'ensemble vide et $t_y^\cap$ est un élément maximal de $\boldsymbol{T}^\cap$;
- $t_x^\cap$ est naturellement plus spécifique que $t_y^\cap$, i.e., d'après l'extension naturelle d'un préordre sur un ensemble à un préordre sur l'ensemble des parties de cet ensemble.

Ou plus formellement :

Proposition 7.3.6. *Soit $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$. $t_x^\cap \overset{\cap}{\lesssim} t_y^\cap$ si et seulement si au moins l'un des cas suivants est vrai :*

1. $t_x^\cap \overset{\cap}{\simeq} \bot^\cap$;
2. $t_x^\cap = \varnothing$ *et* $\top^\cap \overset{\cap}{\simeq} t_y^\cap$;
3. $(\forall t_y \in t_y^\cap)(\exists t_x \in t_x^\cap)[t_x \lesssim t_y]$.

Preuve: voir annexe, p. 309.

7.3.3 Cohérence de la structure actancielle des TUC au sein de la hiérarchie

Maintenant que le préordre $\overset{\cap}{\lesssim}$ a été caractérisé, nous verrons dans cette section que les propriétés des PosA et signatures des TUP sont valides pour les PosA et signatures des TUC, excepté pour certains cas dégénérés : le TUC vide, et tout TUC absurde sauf $\boldsymbol{T}$ lui-même.

PosA, Posa obligatoires et interdites. Tout d'abord, les ensembles de PosAObl et PosAInt sont toujours des sous-ensembles de l'ensemble des PosA.

Proposition 7.3.7. *Pour tout TUC $t^\cap \in \mathbf{T}^\cap$,*

$$\boldsymbol{\alpha}_{\boldsymbol{1}}^\cap(t^\cap) \subseteq \boldsymbol{\alpha}^\cap(t^\cap) \tag{7.32}$$
$$\boldsymbol{\alpha}_{\boldsymbol{0}}^\cap(t^\cap) \subseteq \boldsymbol{\alpha}^\cap(t^\cap) \tag{7.33}$$

Preuve: voir annexe, p. 310.

Ensuite, rappelons-nous de la proposition 7.1.2 et du fait que seuls les TUP plus spécifiques que $\boldsymbol{\gamma}(s)$ (resp. $\boldsymbol{\gamma_1}(s)$, $\boldsymbol{\gamma_0}(s)$), ont une PosA (resp. PosAObl, PosAInt) s. Cette propriété est également valide pour la plupart des TUC.

Soit $\downarrow^\cap t$ le plus petit ensemble inférieur de $\boldsymbol{T}^\cap$ qui contienne $\{t\}$:

$$\downarrow^\cap t \overset{\text{def}}{=} \{t^\cap \in \boldsymbol{T}^\cap \mid t^\cap \overset{\cap}{\lesssim} \{t\}\} \tag{7.34}$$

Proposition 7.3.8. *Pour tout $s \in \mathbf{S}_{\mathscr{T}}$, les égalités suivantes sont vraies :*

$$\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing = \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing \tag{7.35}$$
$$\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}_{\boldsymbol{1}}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing = \downarrow^\cap \boldsymbol{\gamma_1}(s) \setminus \varnothing \tag{7.36}$$
$$\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}_{\boldsymbol{0}}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing = \downarrow^\cap \boldsymbol{\gamma_0}(s) \setminus \varnothing \tag{7.37}$$

Preuve: voir annexe, p. 310.

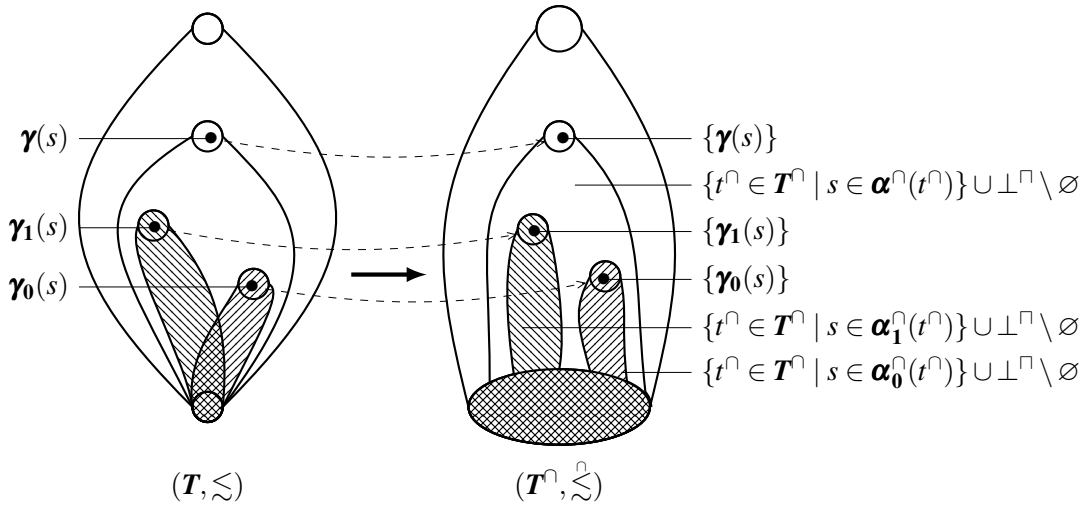


FIGURE 7.17 – Illustration de la proposition 7.1.2 à gauche, et des propositions 7.2.6 et 7.3.8 à droite.

La figure 7.17 illustre les propositions 7.1.2 à gauche, et les propositions 7.3.8 et 7.2.6 à droite.
- À droite, la zone avec hachures \\ et sans hachures // est la zone où les TUC ne sont pas absurdes et ont une PosAObl s.
- À droite, la zone avec hachures // et sans hachures \\ est la zone où les TUC ne sont pas absurdes et ont une PosAInt s.
- À droite, une partie de la zone avec hachures // et \\ correspond aux TUC qui ont la PosA s à la fois obligatoire et interdite.

Remarquons que deux types de TUC ne sont pas pris en compte ici : le TUC $\varnothing$, et tout TUC absurde $t^\cap \in \bot^\sqcap$:
- le TUC $\varnothing$: un problème apparaît par exemple pour $s \in \boldsymbol{S}_{\mathscr{T}}$ tel que $\boldsymbol{\gamma}(s) \simeq \top$. Par construction du préordre $\overset{\cap}{\lesssim}$, $\varnothing \overset{\cap}{\lesssim} \top^\cap$, cependant $\varnothing$ ne possède pas de PosA ($\boldsymbol{\alpha}^\cap(\varnothing) = \bigcup_{t \in \varnothing} \boldsymbol{\alpha}(t) = \varnothing$). Donc $s \notin \boldsymbol{\alpha}^\cap(\varnothing)$.
- Quelques PosA ne sont pas héritées parmi les TUC absurdes. Soit $t^\cap \in \bot^\sqcap$. Un problème apparaît pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$ tel que $s \notin \boldsymbol{\alpha}^\cap(t^\cap)$. Par construction du préordre $\overset{\cap}{\lesssim}$, $t^\cap \overset{\cap}{\lesssim} \bot^\cap$, cependant, par les propositions 7.1.5 et 7.2.1, $s \in \boldsymbol{\alpha}^\cap(\bot^\cap) = \boldsymbol{S}_{\mathscr{T}}$. Ainsi $t^\cap$ n'hérite pas automatiquement de la PosA des TUC plus génériques. Deux exceptions sont $\boldsymbol{T}$ et $\bot^\cap$, qui sont absurdes mais héritent de toutes les PosA ($\boldsymbol{T}$ est composée de tous les TUP, et $\bot^\cap$ par la proposition 7.1.8).

Ces cas dégénérés ont un impact limité, car les TUC absurdes ne doivent en réalité jamais être instanciés. Ainsi le TUC vide et les TUC absurdes, sauf $\boldsymbol{T}$ et $\bot^\cap$, ne sont pas pris en compte dans la plupart des propriétés de cette section. Nous introduisons le terme *TUC régulier* et une notation pour ces TUC, afin d'alléger les formules dans les sections suivantes.

Définition 52 (TUC réguliers)**.** Tout TUC est dit *régulier*, sauf le TUC vide, et tous les TUC absurdes exceptés $\boldsymbol{T}$ et $\bot^\cap$. L'ensemble des TUC réguliers est noté $\boldsymbol{T}^\cap_{regular}$, i.e.,

$$\boldsymbol{T}^\cap_{regular} \overset{\text{def}}{=} \boldsymbol{T}^\cap \setminus \varnothing - (\bot^\sqcap - \{\boldsymbol{T}, \bot^\cap\}) \tag{7.38}$$

Lorsqu'on navigue vers le bas de la hiérarchie des TUC, et qu'ils deviennent de plus en plus spécifiques, l'ensemble des PosA (resp. PosAObl, PosAInt) ne peut que croître. Les PosA (resp. PosAObl, PosAInt) sont donc également héritées parmi les TUC, tant qu'ils restent réguliers.

Proposition 7.3.9. *Les PosA, PosAObl et PosAInt sont héritées parmi les TUC, tant qu'ils restent réguliers. i.e.,* $\forall t_x^\cap \in \mathbf{T}^\cap_{regular}$, *et* $t_y^\cap \in \mathbf{T}^\cap$,

$$t_x^\cap \overset{\cap}{\lesssim} t_y^\cap \implies \boldsymbol{\alpha}^\cap(t_y^\cap) \subseteq \boldsymbol{\alpha}^\cap(t_x^\cap) \tag{7.39}$$

$$t_x^\cap \overset{\cap}{\lesssim} t_y^\cap \implies \boldsymbol{\alpha}_{\boldsymbol{1}}^\cap(t_y^\cap) \subseteq \boldsymbol{\alpha}_{\boldsymbol{1}}^\cap(t_x^\cap) \tag{7.40}$$

$$t_x^\cap \overset{\cap}{\lesssim} t_y^\cap \implies \boldsymbol{\alpha}_{\boldsymbol{0}}^\cap(t_y^\cap) \subseteq \boldsymbol{\alpha}_{\boldsymbol{0}}^\cap(t_x^\cap) \tag{7.41}$$

Preuve: voir annexe, p. 311.

PosA optionnelles. De plus, par la proposition 7.3.8, nous savons que tout TUC régulier possédant une PosAOpt s est dans l'ensemble inférieur $\downarrow^\cap \boldsymbol{\gamma}(s)$, et n'est pas dans les ensembles inférieurs $\downarrow^\cap \boldsymbol{\gamma_1}(s)$ ou $\downarrow^\cap \boldsymbol{\gamma_0}(s)$. Donc la proposition suivante est vraie.

Proposition 7.3.10. *Pour tout* $s \in \mathbf{S}_{\mathscr{T}}$,

$$\begin{aligned}
&\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}_?^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing \\
&= \left(\downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing - \downarrow^\cap \boldsymbol{\gamma_1}(s) \setminus \varnothing - \downarrow^\cap \boldsymbol{\gamma_0}(s) \setminus \varnothing\right) \cup \bot^\sqcap \setminus \varnothing \\
&= \left\{ t^\cap \in \mathbf{T}^\cap \mid t^\cap \overset{\cap}{\lesssim} \{\boldsymbol{\gamma}(s)\} \text{ et } t^\cap \not\overset{\cap}{\lesssim} \{\boldsymbol{\gamma_1}(s)\} \text{ et } t^\cap \not\overset{\cap}{\lesssim} \{\boldsymbol{\gamma_0}(s)\} \right\} \cup \bot^\sqcap \setminus \varnothing
\end{aligned}$$

Preuve: voir annexe, p. 311.

Signatures. Comme pour les TUP, lorsqu'on navigue vers le bas dans la hiérarchie des TUC (réguliers), et qu'ils deviennent plus spécifiques, la signature d'une même PosA ne peut qu'être spécialisée :

Proposition 7.3.11. *Pour tout* $t_x^\cap \in \mathbf{T}^\cap_{regular}$, $t_y^\cap \in \mathbf{T}^\cap$ *et* $s \in \mathbf{S}_{\mathscr{T}}$ *tel que* $s \in \boldsymbol{\alpha}^\cap(t_y^\cap)$,

$$t_x^\cap \overset{\cap}{\lesssim} t_y^\cap \implies \boldsymbol{\varsigma}^\cap_{t_x^\cap}(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}^\cap_{t_y^\cap}(s) \tag{7.42}$$

Preuve: voir annexe, p. 311.

La figure 7.18 illustre l'exemple suivant : Si $t_x^\cap = \{/\text{régicide}\backslash, /\text{poignarder}\backslash\}$ et $t_y^\cap = \{/\text{tuer}\backslash\}$, alors $t_x^\cap$ est plus spécifique que $t_y^\cap$. $/\text{tuer}\backslash$ possède une PosA *tué* qui correspond à la personne tuée, i.e., $\boldsymbol{\varsigma}^\cap_{t_y^\cap}(\textit{tué}) = \{/\text{personne}\backslash\}$. Et la signature de $t_x^\cap$ pour la PosA *tué* est $\boldsymbol{\varsigma}^\cap_{t_x^\cap}(\textit{tué}) = \{/\text{roi}\backslash, /\text{personne}\backslash\}$, qui est plus spécifique que $\boldsymbol{\varsigma}^\cap_{t_y^\cap}(\textit{tué})$. Cela est en accord avec la conceptualisation des signatures proposée dans la section 3.3.3.2.

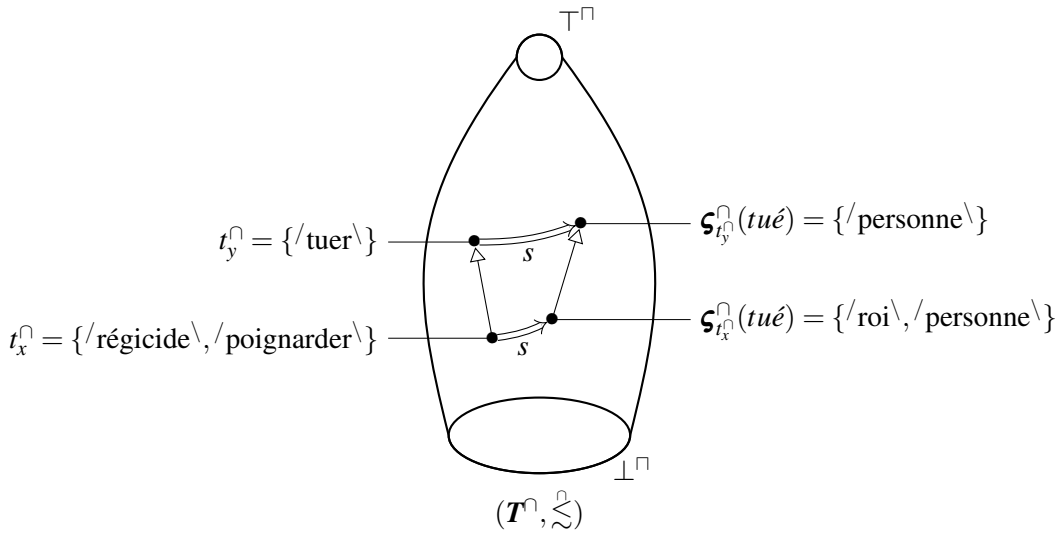


FIGURE 7.18 – Illustration de la proposition 7.3.11.

7.3.4 Classes d'équivalence, clôtures, ordres partiels

Cette section caractérise la relation d'équivalence naturelle sur l'ensemble des TUC (§7.3.4.1), et étudie le sous-ensemble remarquable des *TUC clos* : ceux qui possèdent une cardinalité maximale au sein de leur classe d'équivalence (§7.3.4.2). Nous montrons en particulier que l'opérateur qui associe à chaque TUC le TUC équivalent de cardinalité maximale est un opérateur de clôture sur $\boldsymbol{T}^{\cap}$, et que le préordre $\overset{\cap}{\lesssim}$ définit un ordre partiel sur l'ensemble des TUC clos. Finalement, nous montrons que la cohérence de la structure actancielle est renforcée dans l'ensemble partiellement ordonné des TUC clos (§7.3.4.3).

7.3.4.1 Classes d'équivalence des TUC

Soit $\overset{\cap}{\simeq}$ la relation d'équivalence naturelle définie par $t_x^{\cap} \overset{\cap}{\simeq} t_y^{\cap} \Leftrightarrow \left(t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \wedge t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}\right)$. L'ensemble des classes d'équivalence de TUC définit une partition de $\boldsymbol{T}^{\cap}$. Soit $t^{\cap} \in \boldsymbol{T}^{\cap}$, nous notons $[t^{\cap}] \overset{\text{def}}{=} \{t_x^{\cap} \in \boldsymbol{T}^{\cap} \mid t_x^{\cap} \overset{\cap}{\simeq} t^{\cap}\}$ la classe d'équivalence à laquelle $t^{\cap}$ appartient. Nous noterons usuellement $t^{\sqcap}$ une classe d'équivalence variable.

Définition 53 (Ensemble des classes d'équivalence de TUC)**.** L'*ensemble des classes d'équivalence de TUC* $\boldsymbol{T}^{\sqcap}$ est l'ensemble quotient de $\boldsymbol{T}^{\cap}$ par $\overset{\cap}{\simeq}$, i.e., $\boldsymbol{T}^{\sqcap} \overset{\text{def}}{=} \boldsymbol{T}^{\cap}/\overset{\cap}{\simeq} = \{[t^{\cap}] \mid t^{\cap} \in \boldsymbol{T}^{\cap}\}$. Nous définissons un ordre partiel $\overset{\sqcap}{\leq}$ sur $\boldsymbol{T}^{\sqcap}$ avec $[t_x^{\cap}] \overset{\sqcap}{\leq} [t_y^{\cap}]$ si et seulement si $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$.

Par la proposition 7.2.5, nous savons que la classe d'équivalence de $\top^{\cap}$ est égale à l'ensemble des TUC universels (cf., définition 48), i.e., $\top^{\sqcap} = [\top^{\cap}]$.

Par la proposition 7.2.7, nous savons que la classe d'équivalence de $\bot^{\cap}$ est égale à l'ensemble des TUC absurdes (cf., définition 49), i.e., $\bot^{\sqcap} = [\bot^{\cap}]$;

Proposition 7.3.12. *$\top^{\sqcap}$ et $\bot^{\sqcap}$ sont respectivement les bornes supérieures et inférieures de* $(\mathbf{T}^{\sqcap}, \overset{\sqcap}{\leq})$.

Preuve: voir annexe, p. 312.

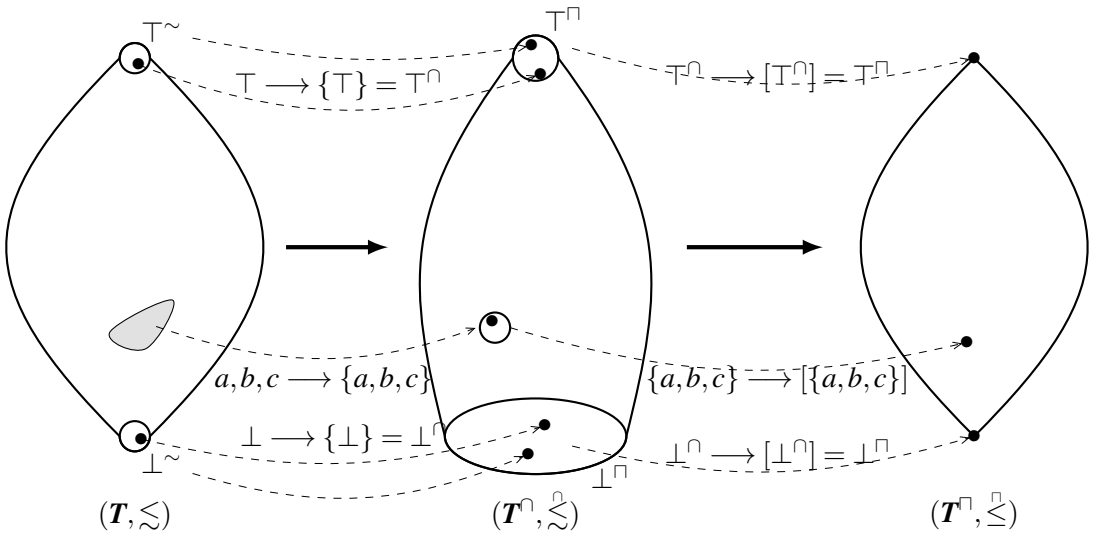


FIGURE 7.19 – Illustration du passage des TUP aux classes d'équivalence de TUC.

La figure 7.19 illustre le passage des TUP aux classes d'équivalence de TUC, avec quelques exemples :

- le TUP $\top$ forme le TUC $\top^\cap$. $\top^\cap$ a pour classe d'équivalence $\top^\sqcap$ par rapport à $\overset{\cap}{\lesssim}$, qui est également la classe d'équivalence du TUC $\top^\sim$. $\top^\sqcap$ est la borne supérieure de l'ensemble partiellement ordonné $(\boldsymbol{T}^\sqcap, t^\sqcap)$.
- le TUP $\bot$ forme le TUC $\bot^\cap$. $\bot^\cap$ a pour classe d'équivalence $\bot^\sqcap$ par rapport à $\overset{\cap}{\lesssim}$, qui est également la classe d'équivalence du TUC $\bot^\sim$. $\bot^\sqcap$ est la borne inférieure de l'ensemble partiellement ordonné $(\boldsymbol{T}^\sqcap, t^\sqcap)$.
- le TUP a,b,c forme le TUC $\{a,b,c\}$. La classe d'équivalence correspondante est $[\{a,b,c\}]$.

Les deux propositions suivantes nous seront utiles dans les preuves des prochaines sections. Tout d'abord, l'ensemble des classes d'équivalence est fermé par l'opération d'union :

Proposition 7.3.13. *Soit $t^\sqcap \in \mathbf{T}^\sqcap$, et $t_x^\cap, t_y^\cap \in t^\sqcap$. Alors $t_x^\cap \cup t_y^\cap \in t^\sqcap$.*

Preuve: voir annexe, p. 312.

Ensuite, ajouter un TUP t à un TUC $t^\cap$ ne lui fait pas changer de classe d'équivalence, tant que t est supérieur à l'un des TUP de $t^\cap$:

Lemme 7.3.14. *Soit $t^\cap \in \mathbf{T}^\cap$ et $t \in \mathbf{T}$ tel que $\exists t' \in t^\cap, t' \lesssim t$. Alors $t^\cap \cup \{t\} \triangleq t^\cap$.*

Preuve: voir annexe, p. 312.

7.3.4.2 Opérateur de clôture sur les TUC

Les TUC permettent un premier mécanisme d'inférence, qui est l'inférence de type. Si un TUC $t_x^\cap$ est inférieur à $t_y^\cap$, alors toute instance de $t_x^\cap$ est également une instance de tout TUP dans $t_y^\cap$. Dans cette section, nous considérons un sous-ensemble remarquable de TUC que nous nommons TUC clos.

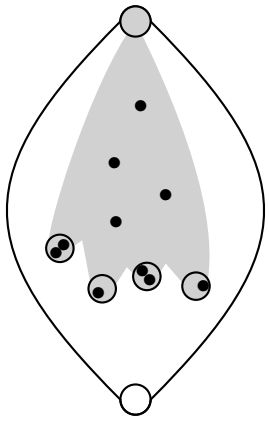
Introduisons un opérateur de clôture $\Uparrow$ qui associe à tout TUC $t^\cap$ l'union de tous les TUC de $[t^\cap]$:

Définition 54 (Opérateur de clôture)**.** L'*opérateur de clôture* $\Uparrow$ défini sur $\boldsymbol{T}^\cap$ associe à chaque TUC $t^\cap$ le TUC $\Uparrow t^\cap \stackrel{\text{def}}{=} \bigcup_{t^\cap \in [t^\cap]} t^\cap$.

Définition 55 (Ensemble des TUC clos)**.** L'*ensemble des TUC clos* est noté $\boldsymbol{T}^\Uparrow$, et est défini comme suit :

$$\boldsymbol{T}^\Uparrow = \{\Uparrow t^\cap \mid t^\cap \in \boldsymbol{T}^\cap\}$$

La figure 7.20 illustre un TUC clos : si $t^\cap$ est un TUC qui consiste en tous les points noirs, alors la région grisée représente l'ensemble des TUP qui appartiennent à $\Uparrow t^\cap$.



$(\boldsymbol{T}, \lesssim)$

FIGURE 7.20 – Illustration d'un TUC clos $\Uparrow t^\cap$ avec $t^\cap \in \boldsymbol{T}^\cap_{regular}$.

Dans la suite de cette section, nous étudions l'opérateur de clôture $\Uparrow$, et l'ensemble des TUC clos $\boldsymbol{T}^\Uparrow$.

Tout d'abord, l'opérateur de clôture préserve la classe d'équivalence. Cela signifie qu'un TUC $t^\cap$ et son TUC clos correspondant $\Uparrow t^\cap$ sont dans la même classe d'équivalence, comme l'illustre la figure 7.21.

Proposition 7.3.15. *La restriction de l'opérateur de clôture $\Uparrow$ à n'importe quelle classe d'équivalence de TUC est un endomorphisme de cette classe, i.e., pour tout $t^\sqcap \in \mathbf{T}^\sqcap$, si $t^\cap \in t^\sqcap$, alors $\Uparrow t^\cap \in t^\sqcap$. Ou de manière équivalente,*

$$(\forall t^\cap \in \mathbf{T}^\cap)[\Uparrow t^\cap \stackrel{\circ}{\simeq} t^\cap]$$

Preuve: voir annexe, p. 312.

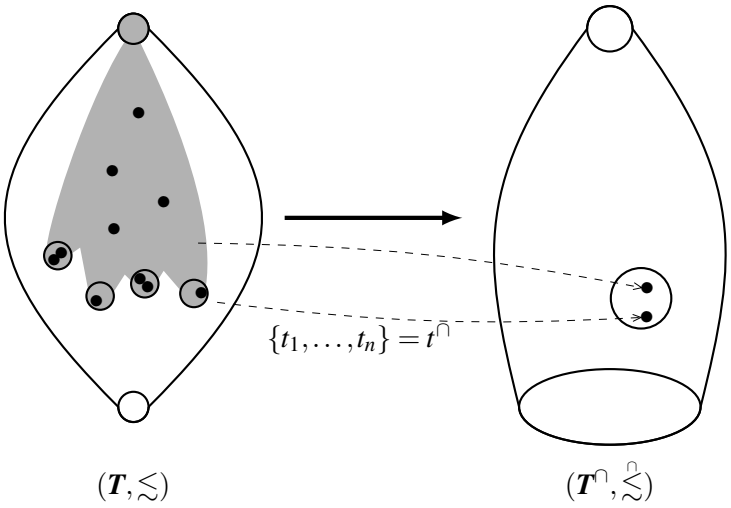


FIGURE 7.21 – Illustration de la proposition 7.3.15 : $\Uparrow t^{\cap}$ est équivalent à $t^{\cap}$.

Ensuite, tous les TUC d'une classe d'équivalence donnée possèdent la même image par l'opérateur $\Uparrow$, comme l'illustre la figure 7.22.

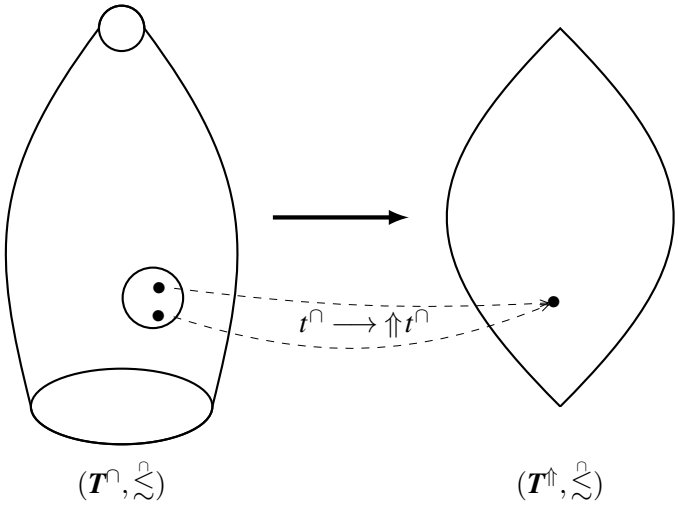


FIGURE 7.22 – Illustration de la proposition 7.3.16 : si $t_x^{\cap} \overset{\cap}{\simeq} t_y^{\cap}$, alors $\Uparrow t_x^{\cap} = \Uparrow t_y^{\cap}$.

Proposition 7.3.16. *Le noyau de l'opérateur de clôture $\Uparrow$ sur* **T** *est la relation d'équivalence $\overset{\cap}{\simeq}$, i.e., si $t_x^{\cap} \overset{\cap}{\simeq} t_y^{\cap}$, alors $\Uparrow t_x^{\cap} = \Uparrow t_y^{\cap}$*

Preuve: voir annexe, p. 312.

Ainsi, il y a une bijection entre l'ensemble des classes d'équivalence de TUC $\boldsymbol{T}^{\cap}$, et l'ensemble des TUC clos $\boldsymbol{T}^{\Uparrow}$. De plus, $\boldsymbol{T}$ est un TUC clos, dans le sens où c'est le TUC qui possède la cardinalité maximale dans sa classe d'équivalence.

Proposition 7.3.17. *L'opérateur de clôture $\Uparrow$ conserve l'ordre, i.e.,*

$$t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \iff \Uparrow t_x^{\cap} \overset{\cap}{\lesssim} \Uparrow t_y^{\cap} \tag{7.43}$$

Preuve: voir annexe, p. 312.

Soit $t^{\cap} \in \boldsymbol{T}^{\cap}$, nous notons $\uparrow t^{\cap}$ l'ensemble supérieur de $\boldsymbol{T}$ généré par les TUP de $t^{\cap}$, i.e.,

$$\uparrow t^{\cap} \overset{\text{def}}{=} \{t \in \boldsymbol{T} \mid \exists t' \in t^{\cap}, t' \lesssim t\} \tag{7.44}$$

Proposition 7.3.18. *L'opérateur de clôture $\Uparrow$ est tel que chacun des points ci-dessous est vrai :*
- *si $t^{\cap} \in \mathbf{T}^{\cap}_{regular}$, alors $\Uparrow t^{\cap} = \uparrow t^{\cap}$;*
- *si $t^{\cap} \in \bot^{\cap}$, alors $\Uparrow t^{\cap} = \mathbf{T}$;*
- *$\Uparrow \varnothing = \Uparrow \top^{\cap}$;*

Preuve: voir annexe, p. 313.

La figure 7.20 illustre le cas le plus intéressant pour la proposition 7.3.18, qui est celui pour lequel $t^{\cap} \in \boldsymbol{T}^{\cap}_{regular}$.

La condition nécessaire de clôture d'un TUC peut donc être formulée comme suit.

Proposition 7.3.19. *Un TUC $t^{\cap}$ est clos si et seulement si : soit il est l'ensemble complet des TUP, soit il est un ensemble supérieur non-absurde et non-vide de TUP.*

Preuve: voir annexe, p. 313.

Finalement l'opérateur $\Uparrow$ définit un opérateur de clôture sur $\boldsymbol{T}$.

Proposition 7.3.20. *$\Uparrow$ est effectivement un opérateur de clôture sur $\mathbf{T}$, i.e., il satisfait les conditions suivantes pour tous TUC $t_x^{\cap}, t_y^{\cap} \in \mathbf{T}^{\cap}$:*
- *$t_x^{\cap} \subseteq \Uparrow t_x^{\cap}$, ($\Uparrow$ est extensif) ;*
- *$t_x^{\cap} \subseteq t_y^{\cap} \Longrightarrow \Uparrow t_x^{\cap} \subseteq \Uparrow t_y^{\cap}$, ($\Uparrow$ est croissant par rapport à $\subseteq$) ;*
- *$\Uparrow \Uparrow t_x^{\cap} = \Uparrow t_x^{\cap}$, ($\Uparrow$ est idempotent) ;*

Preuve: voir annexe, p. 313.

Ainsi un TUC clos $t_{\Uparrow}^{\cap}$ est également clos dans le sens que tout TUC de sa classe d'équivalence est contenu dans $t_{\Uparrow}^{\cap}$.

Finallement, nous pouvons prouver que le préordre $\overset{\cap}{\lesssim}$ définit un ordre partiel sur l'ensemble des TUC clos $\boldsymbol{T}^{\Uparrow}$, i.e., c'est un préordre antisymétrique.

Proposition 7.3.21. *$\overset{\cap}{\lesssim}$ définit un ordre partiel sur l'ensemble $\mathbf{T}^{\Uparrow}$.*

Preuve: voir annexe, p. 314.

Notons $\overset{\cap}{\leq}$ cet ordre partiel, i.e.,

$$(\forall t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\Uparrow})[t_x^{\cap} \overset{\cap}{\leq} t_y^{\cap} \iff t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}] \tag{7.45}$$

7.3.4.3 Cohérence de la structure actancielle pour les TUC clos

Dans la section 7.3.3, seuls les TUC réguliers étaient pris en compte. Puisque tout TUC clos est régulier, nous pouvons prouver une cohérence plus forte de la structure actancielle des TUC clos. Détaillons seulement deux propriétés intéressantes, qui serons utilisées dans le chapitre 9.

Proposition 7.3.22. *Les PosA, PosAObl et PosAInt sont héritées parmi les TUC clos, i.e.,* $\forall t_x^{\cap}, t_y^{\cap} \in \mathbf{T}^{\Uparrow}$,

$$t_x^{\cap} \overset{\cap}{\leq} t_y^{\cap} \implies \boldsymbol{\alpha}^{\cap}(t_y^{\cap}) \subseteq \boldsymbol{\alpha}^{\cap}(t_x^{\cap}) \tag{7.46}$$

$$t_x^{\cap} \overset{\cap}{\leq} t_y^{\cap} \implies \boldsymbol{\alpha}_{\boldsymbol{1}}^{\cap}(t_y^{\cap}) \subseteq \boldsymbol{\alpha}_{\boldsymbol{1}}^{\cap}(t_x^{\cap}) \tag{7.47}$$

$$t_x^{\cap} \overset{\cap}{\leq} t_y^{\cap} \implies \boldsymbol{\alpha}_{\boldsymbol{0}}^{\cap}(t_y^{\cap}) \subseteq \boldsymbol{\alpha}_{\boldsymbol{0}}^{\cap}(t_x^{\cap}) \tag{7.48}$$

Preuve: voir annexe, p. 314.

Proposition 7.3.23. *Pour tout* $t_x^{\cap}, t_y^{\cap} \in \mathbf{T}^{\Uparrow}$ *et* $s \in \mathbf{S}_{\mathscr{T}}$ *tels que* $s \in \boldsymbol{\alpha}^{\cap}(t_y^{\cap})$,

$$t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \implies \boldsymbol{\varsigma}_{t_x^{\cap}}^{\cap}(s) \overset{\cap}{\leq} \boldsymbol{\varsigma}_{t_y^{\cap}}^{\cap}(s) \tag{7.49}$$

Preuve: voir annexe, p. 314.

Conclusion

Nous avons ainsi introduit et caractérisé une structure mathématique qui permet la représentation des types d'unité linguistique munis d'une structure actancielle.

Nous avons introduit dans la section 7.1 un ensemble de types d'unité primitifs (TUP) munis de positions actancielles qui peuvent être obligatoires, interdites ou optionnelles, et munies de signatures. Nous avons proposé une organisation hiérarchique de ces TUP, au sein de laquelle la structure actancielle est effectivement héritée et potentiellement spécialisée, en accord avec la conceptualisation des types d'unité linguistique développée dans le chapitre 3.

Nous avons ensuite étendu notre étude aux types d'unité conjonctifs (TUC) dans la section 7.2. La description de la structure actancielle d'un TUC s'exprime simplement à partir des structures actancielles des TUP qui le composent. Nous avons ensuite introduit un préordre sur l'ensemble des TUC, légèrement plus complexe que l'extension naturelle d'un préordre sur un ensemble à un préordre sur l'ensemble de ses parties. Nous avons montré en quoi ce préordre nous permet de définir les TUC universels et absurdes plus naturellement que pour les TUP, et en quoi il facilite la construction d'une sémantique d'interprétation pour le formalisme des GU, que nous développerons dans le chapitre 9.

Finalement, nous avons introduit un processus de construction itérative du préordre sur l'ensemble des TUC, grâce auquel nous avons pu caractériser les TUC dans la section 7.3.

Nous avons précisé la condition nécessaire et suffisante pour qu'un TUC soit : i) universel, ii) absurde, ou iii) comparable à un autre TUC. Le second résultat est particulièrement intéressant, car il montre que le caractère absurde d'un TUC se propage par héritage et par signature d'un PosAObl.

Grâce à ces conditions nécessaires et suffisantes, nous avons pu montrer que les structures actancielles des TUC sont également héritées et potentiellement spécialisées, en accord avec la conceptualisation des types d'unité linguistique développée dans le chapitre 3. Seuls le type vide et les types absurdes font exception.

Enfin, nous avons caractérisé la relation d'équivalence naturelle sur l'ensemble des TUC, et étudié le sous-ensemble remarquable des TUC qui possèdent une cardinalité maximale au sein de leur classe d'équivalence. Nous avons montré que l'opérateur qui associe à chaque TUC le TUC équivalent de cardinalité maximale est un opérateur de clôture sur $\boldsymbol{T}^{\cap}$, et que le préordre $\overset{\cap}{\lesssim}$ définit un ordre partiel sur l'ensemble des TUC clos. Nous avons finalement montré que la cohérence de la structure actancielle est renforcée dans l'ensemble partiellement ordonné des TUC clos.

Une hiérarchie des types d'unité est donc un ensemble minimal d'objets mathématiques nécessaires à la formation de la base cohérente du formalisme des GU : un ensemble préordonné de TUC munis de structures actancielles. Elle se note $\mathscr{T} = \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Gamma_1}, \boldsymbol{\gamma_1}, \boldsymbol{\Gamma_0}, \boldsymbol{\gamma_0}, C_A, \bot_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}} \rangle$, et se compose d'un ensemble de TUP déclarés T_D, d'un ensemble de SRelA $\boldsymbol{S}_{\mathscr{T}}$, chaque SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$ étant associé à trois TUP auxiliaires :

- la racine de PosA s, $\boldsymbol{\gamma}(s) \in \boldsymbol{\Gamma}$, qui introduit une position actancielle de ce symbole ;
- la racine de PosAObl s, $\boldsymbol{\gamma_1}(s) \in \boldsymbol{\Gamma_1}$, qui rend cette position actancielle obligatoire ;
- la racine de PosAInt s, $\boldsymbol{\gamma_0}(s) \in \boldsymbol{\Gamma_0}$, qui rend cette position actancielle interdite.

Le préordonnancement des TUP est généré par un ensemble de comparaisons de TUP C_A, et le préordonnancement des TUC est construit de sorte que chaque TUC déclaré absurde (ensemble $\bot_A^{\sqcap}$) soit un élément minimal. Finalement, $\{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}}$ définit l'ensemble des signatures des TUP, à partir desquelles l'ensemble des signatures des TUC est calculé.

Le chapitre suivant poursuit la construction du formalisme des Graphes d'Unités par la définition :

- des Graphes d'Unités, qui permettent de représenter les représentations linguistiques ;
- des définitions de TUP, qui permettent de représenter des définitions lexicographiques formelles.

CHAPITRE 8

Des graphes d'unités aux définitions de types d'unité pour représenter les définitions lexicographiques

---

Tout écrivain doit être peintre, autant du moins que le sujet qu'il traite le permet. Or, nos pensées sont susceptibles de différents coloris : séparées, chacune a une couleur qui lui est propre ; rapprochées, elles se prêtent mutuellement des nuances, et l'art consiste à peindre ces reflets.
Étienne Bonnot de Condillac, Traité de l'art d'écrire, 1803.

Sommaire

Introduction

Maintenant que nous avons introduit et caractérisé la hiérarchie des types d'unité, ce chapitre poursuit la construction du formalisme des Graphes d'Unités par la définition des graphes d'unités eux-mêmes, et leur utilisation pour formaliser les définitions lexicographiques.

- la section 8.1 définit une hiérarchie de relations binaires non actancielles, qui permettent de représenter les relations circonstancielles dans les GU ;
- la section 8.2 précise comment les unités peuvent être interconnectées pour construire des GU, adaptés à la conceptualisation introduite dans la section 4.1 ;
- enfin, la section 8.3 introduit la notion de règle, et en dérive les définitions des types d'unité, qui permettent de représenter un sous-ensemble des définitions lexicographiques formelles introduites dans la section 4.2.

8.1 Hiérarchie des symboles de relation circonstancielle

Tout d'abord, bien que nous nous focalisions sur les niveaux de représentation linguistique sémantique de surface et profond, nous souhaitons que le formalisme des Graphes d'Unités puisse être facilement étendu aux niveaux plus surfaciques. Comme précisé dans la section 4.1.1.2, nous introduisons donc un jeu de *symboles de relation circonstancielle*, qui complète l'ensemble des symboles de relation actancielle.

Les relations de dépendance circonstancielles sont binaires, optionnelles, et peuvent être multiples (i.e., une unité peut gouverner zéro ou plusieurs unités à travers des relations de même symbole). Ces relations pourront être utilisées pour représenter les relations circonstancielles au niveau syntaxique profond, mais nous verrons qu'elles pourront également représenter le lien entre une unité lexicale dans une représentation syntaxique profonde, et l'unité sémantique de surface correspondante dans une représentation sémantique de surface, par exemple. Précisons ces deux usages.

Relations syntaxiques profondes. Dans le jeu des relations syntaxiques profondes universelles, les relations **ATTR**(ibutive), **COORD**(inative), et **APPEND**(itive), sont les trois relations circonstancielles, comme précisé dans la section 4.1.2.1. Les niveaux sémantique profond et sémantique de surface ne font pas usage de relations circonstancielles, elles ne devraient donc pas apparaître ni dans les représentations linguistiques à ces niveaux, ni dans les définitions lexicographiques.

Lien entre unités de représentation linguistique de niveaux différents. Nous proposons d'utiliser des relations circonstancielles pour représenter le lien entre une unité dans une représentation linguistique à un niveau de représentation, et l'unité correspondante dans une représentation linguistique associée à un autre niveau de représentation.

Considérons une unité lexicale dans une représentation syntaxique profonde. Si l'unité lexicale est pleine, elle peut être associée à une unité sémantique de surface dans la représentation sémantique de surface. Si maintenant la représentation syntaxique admet plusieurs représentations sémantiques de surface associées (ex : la phrase présente une ambiguïté sémantique), l'unité lexicale pourrait être liée à une unité sémantique de surface dans chacune de ces représentations sémantiques de surface.

Notre choix de représenter ainsi les liens entre unités de représentation linguistique de niveaux différents est arbitraire, mais montre l'extensibilité du formalisme des Graphes d'Unités. Un approfondissement en tandem avec les linguistes de la TST sera nécessaire après maturation du formalisme des GU pour valider notre choix.

Afin de représenter ce jeu de relations circonstancielles, nous proposons donc d'introduire un ensemble de SRelC, hiérarchisés, et munis d'une signature.

Définition d'un ensemble préordonné de symboles de relation circonstancielle. Nous utilisons des symboles pour différencier les différentes relations circonstancielles, et introduisons donc un ensemble fini de SRelC.

Définition 56 (Ensemble de SRelC)**.** Un *ensemble de symboles de relation circonstancielle* est un ensemble fini de symboles de relation binaire noté $\boldsymbol{S}_{\mathscr{C}}$. Tout symbole de relation circonstancielle s est associé à une URI notée $\text{uri}(s)$.

Les SRelC sont classés en ensembles et sous-ensembles. Nous introduisons donc un préordre sur l'ensemble des SRelC, qui est induit par un ensemble de *comparaisons déclarées de SRelC*.

Définition 57 (Ensemble de comparaisons déclarées de SRelC, et préordre sur l'ensemble $\boldsymbol{S}_{\mathscr{C}}$)**.** Soit $\boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}} \subseteq \boldsymbol{T}^{\cap 2}$ un *ensemble de comparaisons déclarées de SRelC*.

$(\boldsymbol{S}_{\mathscr{C}}, \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}})$ est un graphe orienté sur $\boldsymbol{S}_{\mathscr{C}}$. Il induit un préordre $\overset{\mathscr{C}}{\lesssim}$ sur $\boldsymbol{S}_{\mathscr{C}}$, qui est la fermeture réflexo-transitive $\boldsymbol{C}^*_{\boldsymbol{S}_{\mathscr{C}}}$ de $\boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$, avec $(s,s') \in \boldsymbol{C}^*_{\boldsymbol{S}_{\mathscr{C}}}$ si et seulement si s' est un descendant de s, et s est un ascendant de s'.

$$(\forall s, s' \in \boldsymbol{S}_{\mathscr{C}})[s' \overset{\mathscr{C}}{\lesssim} s \iff (s,s') \in \boldsymbol{C}^*_{\boldsymbol{S}_{\mathscr{C}}}]$$

Signature des SRelC. Chacun des SRelC possède une signature, qui spécifie le type des unités qui sont liées par une relation circonstancielle ayant ce symbole.

Définition 58 (Signature des SRelC)**.** L'*ensemble des signatures des SRelC* $\{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}}$ est un ensemble de couples dans $\boldsymbol{T}^{\cap 2}$. Pour tout SRelC s, $\boldsymbol{\sigma}_s \overset{\text{def}}{=} (\mathit{domaine}(s), \mathit{codomaine}(s))$.
L'ensemble des signatures $\{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}}$ doit être tel que :

$$(\forall s_1, s_2 \in \boldsymbol{S}_{\mathscr{C}})[s_1 \overset{\mathscr{C}}{\lesssim} s_2 \implies \mathit{domaine}(s_1) \overset{\cap}{\lesssim} \mathit{domaine}(s_2)] \tag{8.1}$$

$$(\forall s_1, s_2 \in \boldsymbol{S}_{\mathscr{C}})[s_1 \overset{\mathscr{C}}{\lesssim} s_2 \implies \mathit{codomaine}(s_1) \overset{\cap}{\lesssim} \mathit{codomaine}(s_2)] \tag{8.2}$$

Hiérarchie des SRelC. Nous pouvons maintenant introduire la hiérarchie des SRelC.

Définition 59 (Hiérarchie de SRelC)**.** Une *Hiérarchie de SRelC* $\mathscr{C} \overset{\text{def}}{=} \langle \boldsymbol{S}_{\mathscr{C}}, \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}, \mathscr{T}, \{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}} \rangle$, est composée de :

- $\boldsymbol{S}_{\mathscr{C}}$ un ensemble de SRelC ;
- $\boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$ un ensemble de comparaisons déclarées de SRelC ;
- $\mathscr{T} = \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Gamma_1}, \boldsymbol{\gamma_1}, \boldsymbol{\Gamma_0}, \boldsymbol{\gamma_0}, C_A, \bot_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}} \rangle$ une hiérarchie de types d'unité ;
- $\{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}}$ un ensemble de signatures des SRelC.

8.2 Graphes d'Unité (GU)

Les Graphes d'Unités permettent de représenter les représentations linguistiques, comme cela a été illustré dans la section 4.1.2. Nous avons maintenant introduit toutes les conceptualisations et représentations importantes qui vont nous permettre de définir les graphes d'unités :

- nous avons précisé la différence entre les types, les instances, les identifiants, les nœuds, et les marqueurs d'unités dans la section 3.1.1 ;
- nous avons précisé la conceptualisation et la représentation d'une hiérarchie de types d'unité munis d'une structure actancielle (chapitres 3 et 7) ;
- nous avons précisé la conceptualisation et la représentation des symboles de relation circonstancielle, qui complètent le jeu des relations que l'on rencontre dans les représentations linguistiques (§4.1.1.2 et §8.1) ;
- nous avons précisé la conceptualisation des représentations linguistiques (§4.1).

Nous introduirons d'abord la notion de *support de Graphes d'Unité*, empruntée du formalisme des Graphes Conceptuels (§8.2.1). Nous pourrons ensuite définir formellement les GU (§8.2.2), qui possèdent une représentation naturelle sous forme de multigraphes étiquetés et orientés (§8.2.3). Nous préciserons finalement les notions d'homomorphisme de GU, à partir des homomorphismes de leurs graphes sous-jacents (§8.2.4).

8.2.1 Support de graphes d'unités

Comme nous l'avons précisé dans la section 3.1.1, les GU représentent des nœuds d'unité (cf., définition 5), qui sont liés par des relations actancielles ou circonstancielles. Un nœud d'unité représente une unité, et possède une étiquette (cf., définition 6) qui spécifie :

- le type de l'unité représentée (cf., définition 3) ;
- un ensemble de marqueurs associés à des identifiants d'unité (cf., définition 4).

Comme les Graphes Conceptuels, les GU sont définis sur un *support*, qui est composé d'une hiérarchie de types d'unité $\mathscr{T}$ (cf., §7.2.4), une hiérarchie de SRelC $\mathscr{C}$ (cf., §8.1), et un ensemble d'identifiants d'unité $\boldsymbol{M}$ (cf., définition 4).

Définition 60 (Support de graphes d'unités)**.** Un *support de graphes d'unités* est un n-uplet $\mathscr{S} \stackrel{\text{def}}{=} \langle \mathscr{T}, \mathscr{C}, \boldsymbol{M} \rangle$ où :

- $\mathscr{T}$ est une hiérarchie de types d'unité ;
- $\mathscr{C}$ est une hiérarchie de symboles de relation circonstancielle ;
- $\boldsymbol{M}$ est un ensemble d'identifiants d'unité.

8.2.2 Définition des Graphes d'Unités

La figure 8.1 illustre trois représentations de la phrase *John feels no revulsion at the sight of a dead animal.*. Les représentations sémantique de surface et syntaxique profonde sont adaptées de Kahane (2003). La représentation sémantique profonde est obtenue en choisissant pour chaque TUSemP une structure actancielle selon les principes et critères formulés dans la section 3.3.3.

Un GU est une combinaison de nœuds d'unité interconnectés, définie sur un support donné. Il est composé de nœuds d'unité étiquetés, de triplets actanciels, de triplets circonstanciels, et d'une relation d'équivalences déclarées.

L'étiquette d'un nœud d'unité, en extension de la définition 6, est composée :

- d'un TUC, qui permet de spécifier le type de l'unité ;

– et d'un ensemble de marqueurs, qui permet d'identifier l'unité représentée.

L'ensemble des marqueurs des nœuds d'unité est donc l'ensemble des parties de l'ensemble des identifiants d'unité.

Définition 61 (Ensemble des marqueurs de nœuds d'unité)**.** L'*ensemble des marqueurs des nœuds d'unité* est noté $\boldsymbol{M}^{\cap}$ et est égal à l'ensemble des parties de l'ensemble des identifiants d'unité, i.e., $\boldsymbol{M}^{\cap} \stackrel{\text{def}}{=} 2^{\boldsymbol{M}}$. Si un nœud d'unité est marqué $\varnothing$, alors il est dit *générique*, et l'unité représentée est inconnue. D'un autre côté, si un nœud d'unité est marqué $\{m_1, m_2\}$, alors les identifiants d'unité m_1 et m_2 identifient en fait la même unité.

Chaque triplet actanciel lie deux nœuds d'unité avec un SRelA s. L'unité représentée par le second nœud d'unité prend la PosA de symbole s de l'unité représentée par le premier nœud d'unité.

Chaque triplet circonstanciel lie deux nœuds d'unité avec un SRelC s. L'unité représentée par le second nœud d'unité dépend de l'unité représentée par le premier nœud d'unité par rapport à s.

Finalement, une relation d'équivalences déclarées permet de déclarer que deux nœuds d'unité représentent la même unité. C'est ainsi que l'on représentera les relations de coréférence anaphorique par exemple.

Définition 62 (Graphe d'Unité et visualisation)**.** L'ensemble des GU définis sur un support $\mathscr{S}$ est noté $\mathscr{G}(\mathscr{S})$, et chaque GU $G \in \mathscr{G}(\mathscr{S})$ est un quintuplet $G \stackrel{\text{def}}{=} \langle U, \boldsymbol{l}, A, C, Eq \rangle$ où :

- U est l'ensemble des *nœuds d'unité*, en accord avec la définition 5. Ils sont illustrés par des rectangles comme sur la figure 8.1.
- Les nœuds d'unité sont étiquetés en accord avec la définition 6. L'application d'étiquetage $\boldsymbol{l}$ associe à chaque nœud $u \in U$ une étiquette $\boldsymbol{l}(u) = (t^{\cap}, m^{\cap}) \in \boldsymbol{T}^{\cap} \times \boldsymbol{M}^{\cap}$ composée d'un TUC et d'un marqueur de nœud d'unité. Nous notons $t^{\cap} = type(u)$ et $m^{\cap} = marker(u)$. L'étiquette d'un nœud d'unité est notée dans le rectangle qui illustre le nœud d'unité, comme spécifié dans les sections 3.1.2 et 3.3.1, et avec les règles supplémentaires de la section 4.1.1.1.
- A est l'ensemble des *triplets actanciels* $(u, s, v) \in U \times \boldsymbol{S}_{\mathscr{T}} \times U$. Pour tout $a = (u, s, v) \in A$, l'unité représentée par v prend la PosA s de l'unité représentée par u. Nous notons $u = governor(a)$, $s = symbol(a)$ et $v = actant(a)$. Nous notons également $arc(a) = (u, v)$. La visualisation des triplets actanciels est précisée dans la section 4.1.1.2.
- C est l'ensemble des *triplets circonstanciels* $(u, s, v) \in U \times \boldsymbol{S}_{\mathscr{C}} \times U$. Pour tout $c = (u, s, v) \in C$, l'unité représentée par u gouverne l'unité représentée par v par rapport à s. Réciproquement, l'unité représentée par v dépend de l'unité représentée par u par rapport à s. Nous notons $u = governor(c)$, $s = symbol(c)$ et $v = circumstantial(c)$. Nous notons également $arc(c) = (u, v)$. La visualisation des triplets circonstanciels est précisée dans la section 4.1.1.2.
- $Eq \subseteq U^2$ est la *relation d'équivalences déclarées*. Pour tout $(u_1, u_2) \in U^2$, $(u_1, u_2) \in Eq$ signifie que u_1 et u_2 représentent la même unité. La relation Eq n'est pas une relation d'équivalence sur les nœuds d'unité [1]. Il s'agit d'un sous-ensemble de la relation d'équivalence qui sera elle calculée en en faisant la fermeture symétrico-réflexo-transitive. Nous distinguons ainsi les connaissances explicites et implicites. La visualisation des triplets circonstanciels est précisée dans la section 4.1.1.2.

1. Une relation d'équivalence est une relation réflexive, symétrique, et transitive.

Rappelons que nous ne représentons pas la structure communicative des représentations linguistiques, comme on l'a justifié dans la section 1.2.2, ni les fonctions lexicales. Nous reportons leur étude aux travaux futurs de cette thèse pour les raisons suivantes :

- Nous nous concentrons sur les définitions lexicographiques, et nous avons montré qu'elles doivent être représentées au niveau sémantique profond. Or, les fonctions lexicales n'opèrent qu'au niveau syntaxique profond.
- La structure communicative est auxiliaire dans les représentations linguistiques. Nous décidons de nous concentrer ici sur la structure principale.

Les GU ainsi définis vont nous permettre de représenter :

- les représentations linguistiques à différents niveaux du modèle ST ;
- les hypothèses et conclusions des règles lexicales et grammaticales ;
- les définitions lexicographiques formelles.

GU absurde premier. Il est possible que les connaissances représentées dans un GU le rende absurde, par exemple, si un nœud d'unité est typé par un TUC absurde. Le GU absurde premier est le plus simple de ces GU absurdes.

Définition 63 (GU absurde premier)**.** Le *GU absurde premier*, défini sur n'importe quel support de graphes d'unités, est défini par $G_\perp = (\{u\}, \boldsymbol{l}, \varnothing, \varnothing, \varnothing)$, avec $type(u) = \perp$ et $marker(u) = \varnothing$.

Fusion de nœuds d'unité dans un GU. Une opération élémentaire sur les GU est de fusionner deux nœuds d'unité.

Définition 64 (Fusion de nœuds d'unité dans un GU)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle$ un GU. Soient $u, v \in U$ deux nœuds d'unité de G. La *fusion de u et v* dans G est définie comme suit :

1. ajouter un nouveau nœud d'unité w, avec $type(w) = type(u) \cup type(v)$ et $marker(w) = marker(u) \cup marker(v)$;
2. remplacer u et v par w dans tout triplet actanciel ou circonstanciel dans $A \cup C$, et dans tout élément de Eq ;
3. supprimer u et v de U.

8.2.3 Graphe sous-jacent d'un graphe d'unités

Les GU ont une représentation naturelle sous forme de multigraphes étiquetés et orientés, composés de nœuds d'unité, d'arcs étiquetés par des SRelA, d'arcs étiquetés par des SRelC, et d'arcs étiquetés par '=' pour la relation d'équivalences déclarées.

Définition 65 (Graphe sous-jacent d'un GU)**.** Le *graphe sous-jacent d'un GU G*, noté $graph(G)$ est un multigraphe étiqueté et orienté où :

- Pour tout $u \in U$, u est un nœud de $graph(G)$ étiqueté par $\boldsymbol{l}(u)$;
- Pour tout $a \in A$, $arc(a)$ est un arc de $graph(G)$ étiqueté par $symbol(a)$;
- Pour tout $c \in C$, $arc(c)$ est un arc de $graph(G)$ étiqueté par $symbol(c)$;
- pour tout $(u_1, u_2) \in Eq$, (u_1, u_2) est un arc de $graph(G)$ étiqueté '='.

Ce sont aux graphes sous-jacents des GU que l'on peut simplement associer une représentation graphique, comme dans la section 4.1. La figure 8.1 représente trois représentations linguistiques de la même phrase *John feels no revulsion at the sight of a dead animal*. Dans la représentation

syntaxique profonde, les nœuds d'unité u_n et u_d sont typés par des singletons, et seuls les nœuds d'unité u_{j1} et $uj2$ ne sont pas génériques et ont un marqueur commun : $i01$. A contient notamment $(u_s, \mathbf{I}, u_{j2})$ et $(u_s, \mathbf{II}, u_d)$, où **I** et **II** sont des SRelA. C est composé de $(u_r, \mathbf{ATTR}, u_n)$ et $(u_a, \mathbf{ATTR}, u_d)$, où **ATTR** est une SRelC. Dans l'ensemble Eq il y a (u_{j1}, u_{j2}).

Puisque nous distinguons les unités, les nœuds unités, et les marqueurs d'unités, plusieurs mécanismes peuvent impliquer que deux nœuds d'unité u_1 et u_2 représentent la même unité, ou que deux marqueurs d'unités m_1 et m_2 identifient la même unité. Pour les GU définis ci-dessus, les mécanismes suivants sont en jeu :

- Si u_1 et u_2 sont déclarés équivalents, i.e., $(u_1, u_2) \in Eq$, alors tous les marqueurs dans $marker(u_1) \cup marker(u_2)$ identifient la même unité.
- Si u_1 et u_2 possèdent un marqueur commun, alors ils représentent tous deux la même unité.
- Puisque seule une unité peut prendre une PosA donnée, si (u, r, v_1) et (u, r, v_2) sont tous les deux des triplets actanciels, alors v_1 et v_2 représentent la même unité.
- Si deux nœuds d'unité sont déclarés équivalents, alors ils peuvent être fusionnés selon la définition 64, et les cas cités précédemment peuvent à nouveau s'appliquer.

Ces différents mécanismes donnent un avant-goût des possibilités d'inférence dont on souhaite munir les GU. La sémantique formelle du formalisme des GU sera détaillée dans le chapitre 9. En attendant, précisons la notion de GU fini.

GU finis et infinis.

Définition 66 (GU Fini)**.** Un GU G est fini si et seulement si son graphe sous-jacent $graph(G)$ est fini, et infini dans le cas contraire.

Proposition 8.2.1. *Un GU G est fini si et seulement s'il possède un ensemble fini de nœuds d'unité.*

Preuve: voir annexe, p. 314.

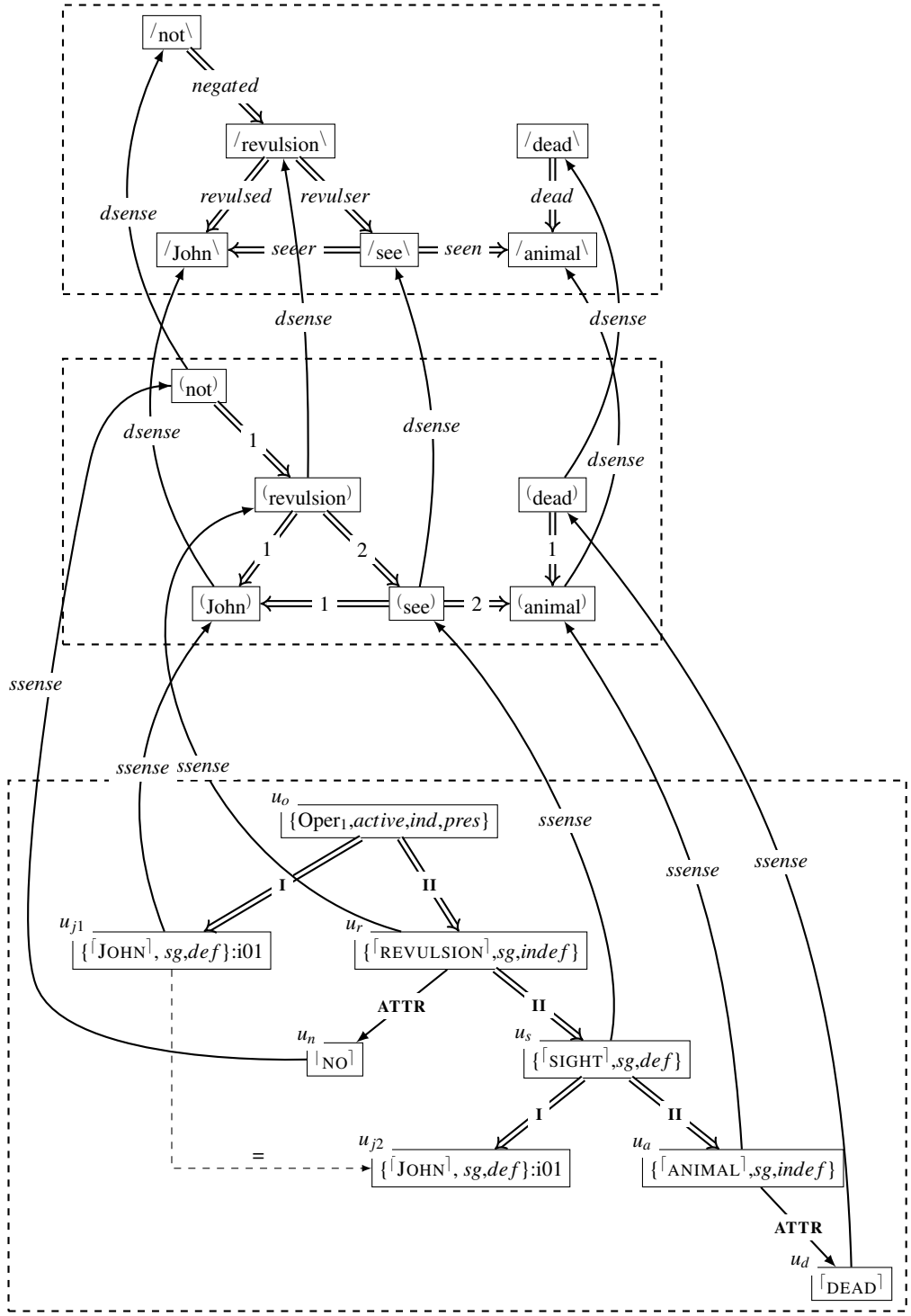


FIGURE 8.1 – Exemple de GU : différentes représentations linguistiques de la phrase *John feels no revulsion at the sight of a dead animal.*

8.2.4 Applications entre GU

Chaque GU a un multigraphe étiqueté et orienté sous-jacent. Nous adaptons la notion classique d'homomorphisme de graphes à ces objets.

Rappelons que pour les graphes non étiquetés, un homomorphisme de H vers G est une application des nœuds de H vers les nœuds de G qui conserve les arcs. Nous introduisons donc la notion d'homomorphisme de GU, en nous basant sur les homomorphismes de leurs graphes sous-jacents. Introduisons tout d'abord la notion d'application entre GU. Une application entre GU correspond à une application entre leurs graphes sous-jacents.

Soient $H = \langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h \rangle$ et $G = \langle U^g, \boldsymbol{l}^g, A^g, C^g, Eq^g \rangle$ deux GU définis sur le même support.

Définition 67 (Application entre GU)**.** Une application f du GU H vers le GU G, notée $f : H \to G$, correspond à une application de leurs graphes sous-jacents, i.e., une application $f : U^h \to U^g$ des nœuds d'unité de H vers les nœuds d'unité de G.

Homomorphisme. Il existe un homomorphisme entre GU s'il existe un homomorphisme entre leurs multigraphes étiquetés et orientés sous-jacents. Afin de définir un tel homomorphisme, nous devons choisir un préordre sur les étiquettes des nœuds et des arcs. Nous choisissons donc les préordres suivants :

- l'inclusion pour les marqueurs de nœuds d'unité ;
- le préordre $\overset{\cap}{\lesssim}$ pour les types de nœuds d'unité ;
- l'égalité pour les triplets actanciels ;
- le préordre $\overset{\mathscr{C}}{\lesssim}$ pour les triplets circonstanciels ;
- l'égalité pour la relation d'équivalences déclarées.

Définition 68 (Homomorphisme de GU)**.** Il existe un *homomorphisme* de H vers G si et seulement s'il existe une application $\pi : H \to G$ telle que chacun des points ci-dessous est vrai :

$$\forall u \in U^h, marker^h(u) \subseteq marker^g(\pi(u)) \tag{8.3}$$
$$\forall u \in U^h, type^g(\pi(u)) \overset{\cap}{\lesssim} type^h(u) \tag{8.4}$$
$$(u,s,v) \in A^h \Rightarrow (\pi(u), s, \pi(v)) \in A^g \tag{8.5}$$
$$(u,s,v) \in C^h \Rightarrow \exists c \in C^g, arc(c) = (\pi(u), \pi(v)) \text{ et } symbol(c) \overset{\mathscr{C}}{\lesssim} s \tag{8.6}$$
$$(u,v) \in Eq^h \Rightarrow (\pi(u), \pi(v)) \in Eq^g \tag{8.7}$$

Par exemple, la figure 8.2 illustre un homomorphisme de H vers G. On suppose dans cet exemple que les comparaisons suivantes sont vraies :

$$\{\lceil \text{COW} \rceil\} \overset{\cap}{\lesssim} \{\text{NOUN}\} \tag{8.8}$$
$$\{\lceil \text{TO GRAZE} \rceil, present, progressive\} \overset{\cap}{\lesssim} \{\text{VERB}\} \tag{8.9}$$
$$\{\lceil \text{TO GRAZE} \rceil, present, progressive\} \overset{\cap}{\lesssim} \{tense\} \tag{8.10}$$
$$\{/\text{cow}\backslash\} \overset{\cap}{\lesssim} \{/\text{animal}\backslash\} \tag{8.11}$$
$$\{/\text{to graze}\backslash\} \overset{\cap}{\lesssim} \{/\text{to eat}\backslash\} \tag{8.12}$$

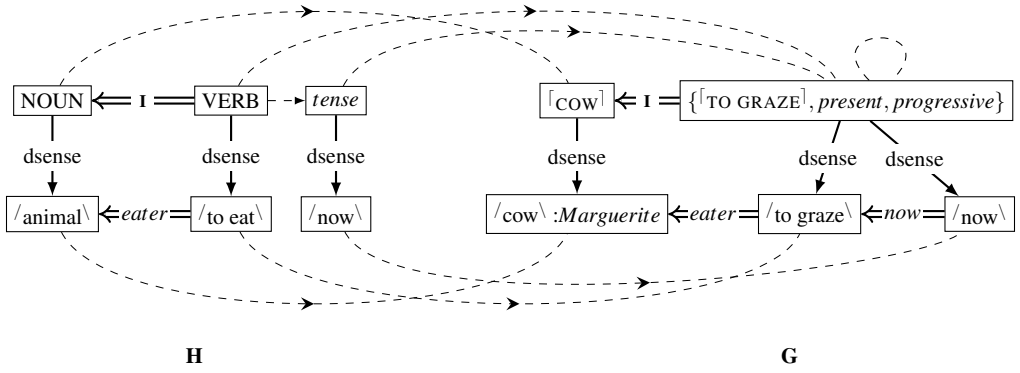


FIGURE 8.2 – Illustration d'un homomorphisme de H vers G.

Hom-Équivalence. Deux GU sont hom-équivalents s'il existe un homomorphisme du premier vers le second, et un homomorphisme du second vers le premier.

Définition 69 (Hom-équivalence entre GU)**.** Soient $G = \langle U^g, \boldsymbol{l}^g, A^g, C^g, Eq^g \rangle \in \mathscr{G}(\mathscr{S})$ et $H = \langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h \rangle \in \mathscr{G}(\mathscr{S})$ deux GU définis sur le même support. G et H sont *hom-équivalent* s'il existe un homomorphisme de G vers H et un homomorphisme de H vers G.

La figure 8.3 illustre deux GU hom-équivalents qui sont différents. On suppose toujours dans cet exemple que les comparaisons 8.8 à 8.12 sont vraies. Les nœuds d'unité u_1 et u_2 de H se projettent tous les deux sur le nœud d'unité v de G, cependant le nœud d'unité v de G se projette sur le nœud d'unité u_1 de H seulement.

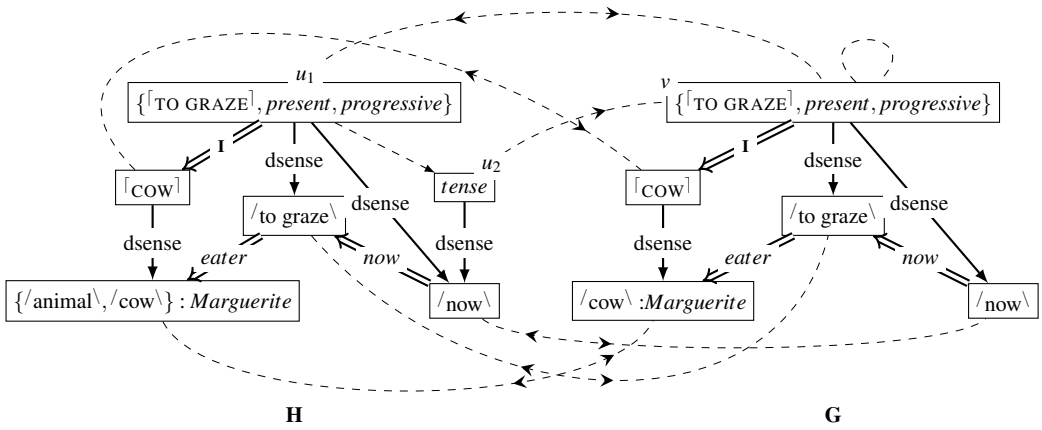


FIGURE 8.3 – Illustration d'une hom-équivalence entre H et G.

Isomorphisme. Une hom-équivalence est un isomorphisme si les deux applications sont inverses l'une de l'autre.

Définition 70 (Isomorphisme de GU)**.** Soient $G = \langle U^g, \boldsymbol{l}^g, A^g, C^g, Eq^g \rangle \in \mathscr{G}(\mathscr{S})$ et $H = \langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h \rangle \in \mathscr{G}(\mathscr{S})$ deux GU définis sur le même support. Une application $\pi : G \to H$ est un *isomorphisme* si et seulement si π est un homomorphisme bijectif.

La figure 8.4 ci-dessous illustre deux GU isomorphes qui sont différents.

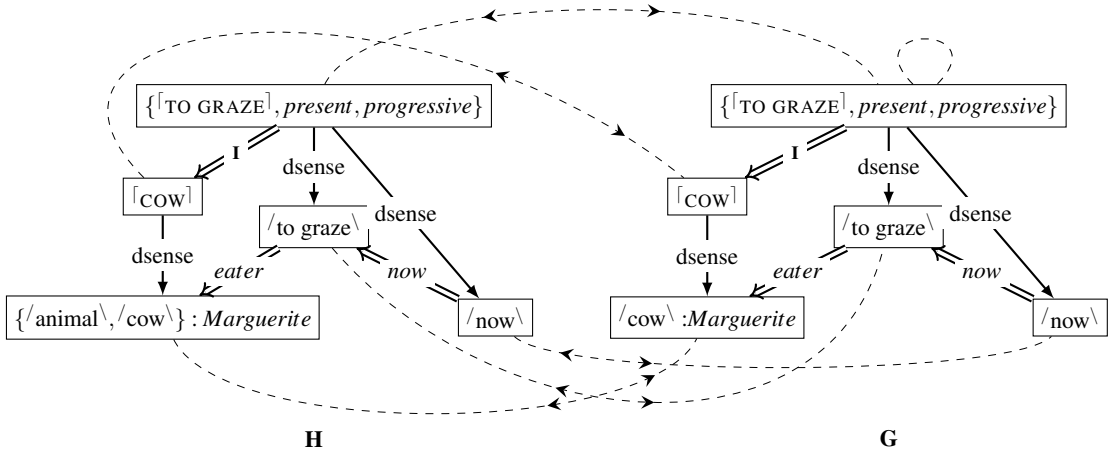


FIGURE 8.4 – Illustration d'un isomorphisme entre H et G.

8.3 Règles et définitions de types d'unité primitifs

Maintenant que nous avons introduit les GU et leurs applications, nous pouvons introduire les notions fondamentales de règles, et de définitions de types d'unité primitifs.

Les règles peuvent être utilisées de différents manières dans le formalisme des GU :
- Les règles permettent par exemple de définir des règles d'inférence qui explicitent des connaissances dans les GU, comme nous le verrons dans le chapitre 9.
- Elles devraient permettent également de représenter les correspondances entre une structure actancielle d'un TUSemP, et une structure actancielle de son TUSemS associé. Nous donnerons un exemple simple dans la section 8.3.1.2, mais ne nous focaliserons pas sur cette application dans cette thèse.

Nous définirons donc la notion de règle dans la section 8.3.1.

Ensuite, la section 8.3.2 introduit la notion de définition de TUP, qui correspond grossièrement à deux règles contraposées impliquant un GU qui représente un TUP et sa structure actancielle. Nous verrons que ces règles sont d'un intérêt tout particulier pour la représentation des définitions lexicographiques de la TST, et nous étendrons donc la définition de hiérarchie des types d'unité pour prendre en compte un ensemble de définitions de TUP.

8.3.1 Règles

Avant d'introduire les règles en tant que telles (§8.3.1.2), définissons tout d'abord la notion de λ-GU, qui est un GU dont certains nœuds sont distingués (§8.3.1.1).

8.3.1.1 λ-GU

Les nœuds d'unité d'un GU dont le marqueur est vide sont dits génériques.

Définition 71 (Généricité)**.** Dans un GU $G \in \mathscr{G}(\mathscr{S})$, un nœud d'unité u est dit *générique* si et seulement si $marker(u) = \varnothing$.
G est dit *générique* si et seulement si tous ses nœuds d'unité sont génériques.

Un λ-GU est un GU dont certains nœuds d'unité génériques sont distingués.

Définition 72 (λ-GU et généricité)**.** Un λ-*GU* $L = \{u_1, \ldots, u_n\}G$ défini sur un support $\mathscr{S}$, est composé d'un GU $G = \langle U, \boldsymbol{I}, A, C, Eq\rangle \in \mathscr{G}(\mathscr{S})$, et d'un ensemble de nœuds d'unité génériques de G, $\{u_1, \ldots, u_n\}$, dits *distingués*.
n est la λ-*taille* de G.
L est dit *entièrement distingué* si et seulement si tous ses nœuds d'unité sont distingués.

Les λ-GU correspondent en réalité à des GU généralisés, et un GU peut être considéré comme étant un λ-GU de λ-taille 0. Nous étendons la définition 64 à la fusion d'un λ-GU C dans un λ-GU G, qui est définie par une fonction partielle [2] de l'ensemble des nœuds d'unité distingués de C vers les nœuds d'unité de G.

Définition 73 (Fusion d'un λ-GU dans un λ-GU selon une fonction partielle)**.** Soient C et G deux λ-GU définis sur un support $\mathscr{S}$:
- $C = \{u_1^c, \ldots, u_n^c\}\langle U^c, \boldsymbol{I}^c, A^c, C^c, Eq^c\rangle$;
- $G = \{u_1^g, \ldots, u_m^g\}\langle U^g, \boldsymbol{I}^g, A^g, C^g, Eq^g\rangle$.

Soit η une fonction partielle de $\{u_1^c, \ldots, u_n^c\}$ vers U^g. La fusion de C dans G selon η est notée $merge(G, C, \eta)$, et est obtenue comme suit :

1. ajouter C à G ;
2. pour tout $(u^c, u^g) \in \eta$, fusionner u^c et u^g selon la définition 64. Le nœud résultant est distingué si et seulement si $u^g \in \{u_1^g, \ldots, u_m^g\}$.

Si G est de λ-taille 0, et si la fonction η est totale, alors le graphe résultant $merge(G, C, \eta)$ est de λ-taille 0.

2. Le degré de généralité de cette définition nous permet, dans des travaux en cours, de représenter plus facilement la notion de dérivation sémantique de la TST, qui est une facette importante des fonctions lexicales.

8.3.1.2 Définition des règles

Une règle est représentée par deux λ-GU et une bijection entre leurs nœuds d'unité distingués.

Définition 74 (Règle de GU)**.** Une *règle de GU* définie sur un support $\mathscr{S}$ est un triplet $R \stackrel{\text{def}}{=} \langle H, C, \kappa \rangle$ où :
- $H = \{u_1^h, \ldots, u_n^h\}H'$ est un λ-GU défini sur $\mathscr{S}$ appelé *prémisse*, ou *hypothèse* ;
- $C = \{u_1^c, \ldots, u_n^c\}C'$ est un λ-GU défini sur $\mathscr{S}$ appelé *conclusion* ;
- κ est une bijection de $\{u_1^h, \ldots, u_n^h\}$ vers $\{u_1^c, \ldots, u_n^c\}$.

Une règle $\langle H, C, \kappa \rangle$ est dite *applicable* à un GU G si et seulement s'il existe un homomorphisme de H vers G.

Définition 75 (Applicabilité d'une règle)**.** Soient $R = \langle H, C, \kappa \rangle$ une règle, et G un GU définis sur un support $\mathscr{S}$. R est *applicable* à G si et seulement s'il existe un homomorphisme de H vers G.

Soit π un tel homomorphisme de H vers G. L'application de R à G selon π est le GU obtenu en fusionnant C dans G selon $\pi \circ \kappa^{-1}$.

Définition 76 (Application d'une règle)**.** Soit $R = \langle H, C, \kappa \rangle$ une règle applicable à un GU G, et soit π un homomorphisme de H vers G. L'application de R à G selon π est notée $\lambda(G, R, \pi)$, et est le GU obtenu en fusionnant C dans G selon $\pi \circ \kappa^{-1}$, comme la définition 73 le décrit.

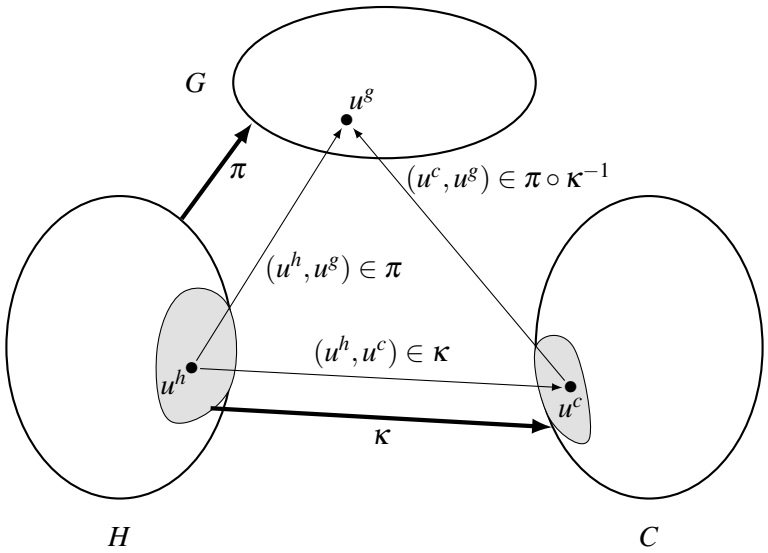


FIGURE 8.5 – Illustration d'une règle $R = \langle H, C, \kappa \rangle$ applicable à un graphe G.

La correspondance entre les structure actancielles du TUSemP $^{/}$to eat$^{\backslash}$, et du TUSemS $^{(}$to eat$^{)}$ a été précisée sur la figure 3.8. Nous pouvons alors générer automatiquement une règle de correspondance (des sens vers les textes), comme illustré par la figure 8.6. Par assemblage de telles règles, nous pourrions représenter les correspondances entre des représentations linguistiques à des niveaux adjacents.

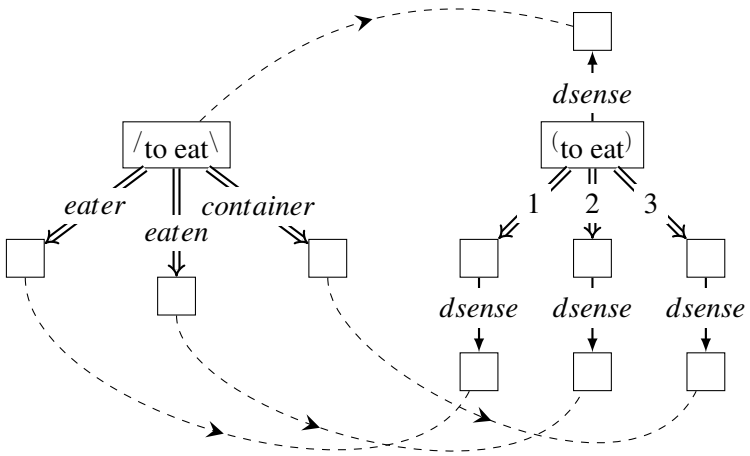


FIGURE 8.6 – Illustration d'une règle de correspondance entre le TUSemP /to eat\ et le TUSemS (to eat).

Notons que chaque PosA peut être ou ne pas être exprimée dans les représentations linguistiques, et qu'en réalité trois règles élémentaires différentes devraient être générées pour la correspondance entre /to eat\ et (to eat) : une pour chaque correspondance élémentaire de PosA. La règle illustrée par la figure 8.6 représente donc déjà un assemblage de ces trois règles. De plus, il est possible que certaines règles soient exclusives, comme pour représenter les PosA scindées par exemple : la PosA 2 du TUSemS (outil) correspond soit à la PosA *profession*, soit à la PosA *activité* de /outil\, par exemple.

8.3.2 Définitions de types d'unité primitifs

Informellement, une définition de TUP définit une équivalence entre deux λ-GU définis sur le même support. L'un des λ-GU est très simple et représente le TUP défini avec une partie de sa structure actancielle, il est appelé *empreinte* de t. L'autre λ-GU est appelé *expansion* de t.

Les empreintes sont d'abord introduites dans la section 8.3.2.1, puis la section 8.3.2.2 introduit la notion de définition des TUP. Nous augmentons alors la définition de la hiérarchie des types d'unité avec un ensemble de définitions de types d'unité (§8.3.2.3), et étudions dans quelle mesure les définitions de TUP permettent de représenter les définitions lexicographiques de la TST telles qu'on les a conceptualisées dans la section 4.2 (§8.3.3).

8.3.2.1 Empreinte d'un TUC

Une *empreinte* d'un TUC $t^\cap$ est un λ-GU qui représente des informations sur la structure actancielle de $t^\cap$. Il s'agit d'un λ-GU complètement distingué, avec un nœud d'unité central de type $t^\cap$. Les autres nœuds représentent chacun des informations sur une PosA particulière de $t^\cap$. Prenons le cas d'un nœud d'unité v qui représente les informations d'une PosA $s \in \boldsymbol{\alpha}^\cap(t^\cap)$. Alors v apparaît seulement dans le triplet actanciel (u,s,v), et est typé par la signature de $t^\cap$ pour s, i.e., $type(v) = \boldsymbol{\varsigma}^\cap_{t^\cap}(s)$.

Nous notons $[[m,n]]$ l'ensemble des entiers $m \leq i \leq n$.

Définition 77 (Empreinte d'un TUC)**.** Une *empreinte d'un TUC* $t^\cap$ est un λ-GU complètement distingué, de graphe d'unité $\langle U, \boldsymbol{l}, A, C, Eq\rangle \in \mathscr{G}(\mathscr{S})$, avec :
- $U = \{u, v_1, \ldots, v_n\}$, avec $n \geq 0$;
- u est appelé le *nœud central* de l'empreinte de t, et $\boldsymbol{l}(u) = (t^\cap, \varnothing)$;
- il existe une application injective $\alpha : [[1,n]] \longrightarrow \boldsymbol{\alpha}^\cap(t^\cap)$, telle que pour tout $i \in [[1,n]]$:
  $a_i = (u, \alpha(i), v_i)$;
  $\boldsymbol{l}(v_i) = (\boldsymbol{\varsigma}^\cap_{t^\cap}(\alpha(i)), \varnothing)$;
- $C = Eq = \varnothing$;

n est la *e-taille* de l'empreinte de $t^\cap$. Les cas suivants correspondent à des empreintes particulières :
- Si $n = 0$, alors G est l'*empreinte vide* de $t^\cap$.
- Si $n = 1$, alors G est l'*empreinte élémentaire* de $t^\cap$ pour la PosA $\alpha(1)$.

Une empreinte partielle d'un TUC $t^\cap$ représente commodément un ensemble des PosA de $t^\cap$ avec leur signature, mais elle ne peut pas représenter la nature de ces PosA. Si une empreinte partielle de $t^\cap$ met en scène l'une de ses PosA interdites, alors elle est dite absurde.

Définition 78 (Empreinte absurde)**.** Une empreinte d'un TUC $t^\cap$ est *absurde* si $codomaine(\alpha) \cap \boldsymbol{\alpha}^\cap_{\mathbf{0}}(t^\cap) \neq \varnothing$.

8.3.2.2 Notion de définition de type d'unité primitif

Nous pouvons finalement formaliser la notion de *définition d'un TUP*, afin d'en inclure un ensemble dans la hiérarchie des types d'unité. Une définition d'un TUP t déclare une équivalence entre deux λ-GU définis sur le même support. L'un est une empreinte de t, et l'autre est appelé *expansion* de t. Nous ignorons les triplets circonstanciels dans ces deux λ-GU, puisqu'ils n'ont pas leur place dans les définitions lexicographiques formelles.

Définition 79 (Définition d'un type d'unité primitif)**.** Une *définition* D_t d'un TUP t est un triplet $D_t \stackrel{\text{def}}{=} \langle D_t^-, D_t^+, \kappa\rangle$ où :
- $D_t^- = \{u_t^-, v_1^-, \ldots, v_n^-\}(U^-, \boldsymbol{l}^-, A^-, \varnothing, \varnothing)$ est une empreinte de t ayant pour nœud d'unité central u_t^- ;
- $D_t^+ = \{u_t^+, v_1^+, \ldots, v_n^+\}(U^+, \boldsymbol{l}^+, A^+, \varnothing, \varnothing)$ est appelé l'*expansion* de t ;
- κ est une bijection de $\{u_t^-, v_1^-, \ldots, v_n^-\}$ vers $\{u_t^+, v_1^+, \ldots, v_n^+\}$, telle que $\kappa(u_t^-) = u_t^+$, et pour tout i, $\kappa(v_i^-) = v_i^+$;
- le type de u_t^+ est appelé le *genre proche* de t, et est noté $genus(t)$, avec $(genus(t), t) \in C_A$ (cf., définition 31).
- $\boldsymbol{l}^+(\kappa(v_i^-)) = \boldsymbol{l}^-(v_i^-)$ pour tout $v_i^- \in \{v_1^-, \ldots, v_n^-\}$;

La figure 8.7 est une définition lexicographique du TUP ⌜PEIGNE⌝ : un instrument qu'une personne X utilise pour démêler les fibres d'un objet Y.

Intuitivement, une définition correspond à deux règles réciproques.

Définition 80 (Règles d'expansion et de contraction d'un TUP t)**.** Une définition de TUP $D_t = (D_t^-, D_t^+, \kappa)$ correspond à deux règles :

def+ La *règle d'expansion* $R_t^+ \stackrel{\text{def}}{=} (D_t^-, D_t^+, \kappa)$;

def- La *règle de contraction* $R_t^- \stackrel{\text{def}}{=} (D_t^+, D_t^-, \kappa^{-1})$;

Ainsi, si l'on rencontre l'empreinte d'un TUP défini t dans un GU, alors on peut inférer son expansion, et vice versa.

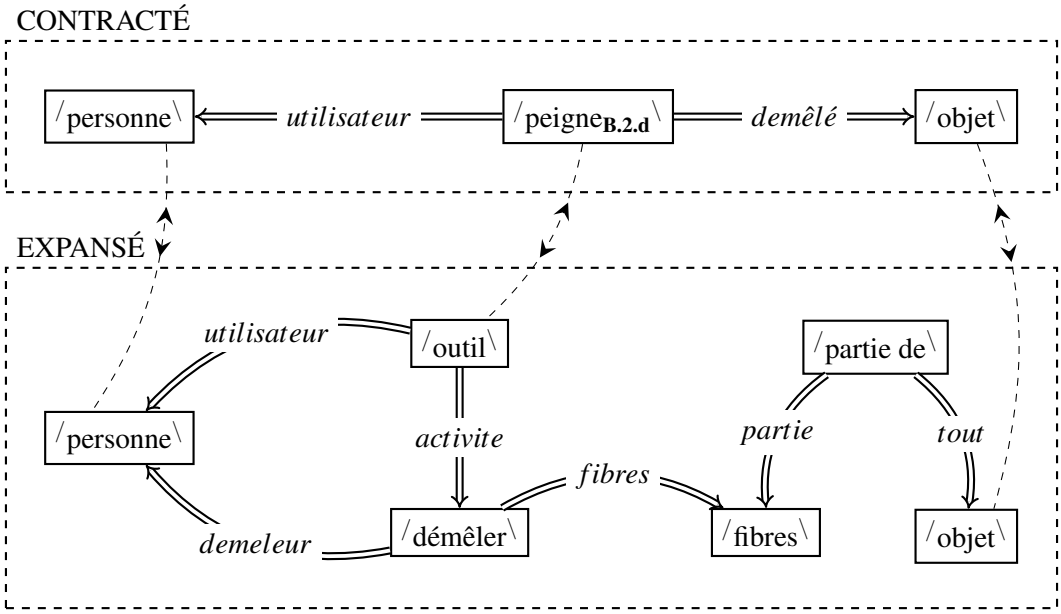


FIGURE 8.7 – Définition du TUP /peigne$_{\mathbf{B.2.d}}$\.

8.3.2.3 Injection des définitions dans la hiérarchie des types d'unité

Nous pouvons maintenant réviser la définition de la hiérarchie des types d'unité $\mathscr{T}$ pour prendre en compte un ensemble de définitions de TUP $\mathscr{D}$.

Définition 81 (Hiérarchie de types d'unité avec définitions)**.** Une hiérarchie des types d'unité, notée $\mathscr{T}$, est un tuple $\mathscr{T} \stackrel{\text{def}}{=} \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\gamma}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_0}, C_A, \perp_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}}, \mathscr{D}\rangle$ qui permet de construire un ensemble préordonné de types d'unité $\boldsymbol{T}^{\cap}$ munis d'une structure actancielle, et avec un ensemble de définitions de TUP $\mathscr{D}$.

Dans la suite de cette thèse, le terme hiérarchie de type d'unités référera à cette définition, sauf mention contraire.

8.3.3 Adéquation du formalisme des Graphes d'Unités pour la représentation des définitions lexicographiques formelles

Nous avons présenté une conceptualisation des définitions lexicographiques formelles dans la section 4.2, et nous montrons ici comment on peut construire une définition de TUP à partir d'une définition lexicographique formelle.

Notre point de départ est une définition lexicographique formelle conceptualisée suivant la définition 22 :

$$D(/\mathrm{L}\backslash) = \langle p_0, P, A, type, nature, marker\rangle$$

Nous nous concentrons sur le sous-cas pour lequel tous les nœuds de participant sont de nature obligatoire. La définition suivante introduit une correspondance entre la définition lexicographique formelle de $\lceil \mathrm{L} \rceil$, et la définition du TUP $/\mathrm{L}\backslash$.

Définition 82 (Aller-retour entre une définition lexicographique formelle et une définition de TUP)**.** Soit $D(/\mathrm{L}\backslash) = \langle p_0, P, A, type, nature, marker\rangle$ la définition lexicographique formelle de $\lceil \mathrm{L} \rceil$. Nous supposons que :

- la fonction *nature* prend ses valeurs dans le singleton {'!'} ;
- Aucun arc de la définition lexicographique formelle n'est dans une configuration inconsistante.

$D(^{/}\mathrm{L}^{\backslash})$ est équivalente à une définition du TUP $^{/}\mathrm{L}^{\backslash}$, $D_{/\mathrm{L}\backslash} = (D^{-}_{/\mathrm{L}\backslash}, D^{+}_{/\mathrm{L}\backslash}, \kappa)$.
Soit $D^{-}_{/\mathrm{L}\backslash} = \{u^{-}_{/\mathrm{L}\backslash}, v^{-}_{1}, \ldots, v^{-}_{n}\}(U^{-}, l^{-}, A^{-}, \varnothing, \varnothing)$
Soit $D^{+}_{/\mathrm{L}\backslash} = \{u^{+}_{/\mathrm{L}\backslash}, v^{+}_{1}, \ldots, v^{+}_{n}\}(U^{+}, l^{+}, A^{+}, \varnothing, \varnothing)$
Nous définissons une correspondance entre $D(^{/}\mathrm{L}^{\backslash})$ et $D_{/\mathrm{L}\backslash}$ comme suit :

- p_0 correspond à la fois à $u^{-}_{/\mathrm{L}\backslash}$ et à $u^{+}_{/\mathrm{L}\backslash}$.
- il existe une bijection κ^{-} entre l'ensemble $P^{-} \stackrel{\text{def}}{=} \{p_0\} \cup \{p \in P | marker(p) \neq \varnothing\}$ (i.e., le nœud de participant principal, et l'ensemble des nœuds de participant qui représentent une PosA de symbole $marker(p)$), et U^{-}. Nous imposons $\kappa^{-}(p_0) = u^{-}_{/\mathrm{L}\backslash}$.
  Nous définissons alors $D^{-}_{/\mathrm{L}\backslash}$ tel que :
  - $type(u^{-}_{/\mathrm{L}\backslash}) = \{^{/}\mathrm{L}^{\backslash}\}$, et $marker(u^{-}_{/\mathrm{L}\backslash}) = \varnothing$.
  - pour tout $p \neq p_0$, $type(\kappa^{-}(p)) = type(p)$, et $marker(\kappa^{-}(p)) = \varnothing$.
  - $A^{-} = \{(u^{-}_{/\mathrm{L}\backslash}, m, \kappa^{-}(p)) | p \in P^{-}, s \in marker(p)\}$
- il existe une bijection κ^{+} entre l'ensemble P et l'ensemble U^{+}. Nous imposons $\kappa^{+}(p_0) = u^{-}_{/\mathrm{L}\backslash}$.
  Nous définissons alors $D^{+}_{/\mathrm{L}\backslash}$ tel que :
  - $type(\kappa^{+}(p)) = type(p)$, et $marker(\kappa^{+}(p)) = \varnothing$.
  - $A^{-} = \{(\kappa^{+}(p_1), s, \kappa^{+}(p_2)) | (p_1, s, p_2) \in A$ et $nature((p_1, s, p_2)) = !\}$.
- la bijection κ correspond alors à $\kappa = \kappa^{+} \circ \kappa^{--1}$.

La représentation des définitions lexicographiques formelles qui possèdent des nœuds de participant optionnel ou interdit fait l'objet des travaux futurs de cette thèse. Nous montrerons dans le chapitre 11 que leur représentation nécessite de complexifier la définition des règles et des définitions de TUP. Par contre, lorsque nous proposerons dans le chapitre 10 d'opérationnaliser le formalisme des GU sur le Web des données, nous proposerons une modélisation des définitions lexicographiques formelles, quelle que soit la nature de leurs nœuds participants.

Conclusion

Nous avons poursuivi la construction du formalisme des Graphes d'Unités, qui nous permet de formaliser de manière adéquate la plupart des connaissances de la TST auxquelles nous nous intéressons dans cette étude.

Nous avons précisé le rôle des relations circonstancielles dans les GU. Pour la représentation de la théorie Sens-Texte, ces relations servent par exemple à représenter les relations circonstancielles au niveau syntaxique profond, mais également à représenter le lien entre une unité lexicale dans une représentation syntaxique profonde, et l'unité sémantique de surface correspondante dans une représentation sémantique de surface. Nous avons donc défini une hiérarchie de relations binaires circonstancielles munies de signatures.

Nous avons introduit et caractérisé les GU dans la section 8.2 : les GU sont définis sur un support, composé d'une hiérarchie de types d'unité, d'une hiérarchie de symboles de relation circonstancielle, et d'un ensemble d'identifiants d'unité.

Un GU est une combinaison de nœuds d'unité interconnectés, définie sur un support donné. Il est composé de nœuds d'unité étiquetés, de triplets actanciels, de triplets circonstanciels, et d'une relation d'équivalences déclarées.

Nous avons introduit une représentation d'un GU sous la forme d'un multigraphes étiqueté et orienté sous-jacent. Nous avons alors entrevu quelques mécanismes d'inférence que l'on peut effectuer sur les GU. Dans le chapitre suivant, nous recenserons ces mécanismes d'inférence dans une base de règles axiomatiques d'inférence, qui va nous permettre de raisonner dans le formalisme des GU.

Nous avons finalement défini différentes applications intéressantes entre les GU, notamment l'homomorphisme entre GU, défini à partir de l'homomorphisme entre leurs GU sous-jacents.

Enfin, la section 8.3 a introduit la notion de règle dans le formalisme des GU. Nous avons montré que les règles peuvent représenter les correspondances entre les structures actancielles des types d'unité de niveaux linguistiques adjacents. Nous avons alors introduit la notion de définition de TUP, comme la combinaison de deux règles réciproques, entre :

– un GU, nommé empreinte, représentant le type défini et une partie de sa structure actancielle ;
– un GU, nommé expansion, représentant le graphe définitoire du type défini.

Nous avons alors augmenté la notion de hiérarchie des types d'unité pour y inclure un ensemble de définitions de TUP.

Nous avons conclu ce chapitre par l'étude de l'adéquation des définitions de TUP pour représenter les définitions lexicographiques formelles telles que conceptualisées dans le chapitre 4. Nous avons montré qu'il est possible de définir une correspondance entre la définition lexicographique formelle d'un type de lexie, et la définition du TUSemP associé, dans le cas où tous les nœuds de participant de la définition lexicographique formelle sont de nature obligatoire.

La représentation des définitions lexicographiques formelles est donc incomplète pour le moment dans le formalisme des GU, et nous montrerons dans le chapitre 11 que leur représentation nécessite de complexifier la définition des règles et des définitions de TUP. Lorsque nous proposerons dans le chapitre 10 d'opérationnaliser le formalisme des GU sur le Web des données, nous proposerons une modélisation des définitions lexicographiques formelles selon leur conceptualisation introduite dans le chapitre 4, et donc quelle que soit la nature de leurs nœuds participants.

Ce chapitre achève donc la partie de ce mémoire dévouée au choix d'un formalisme de représentation de connaissances adapté à la conceptualisation étendue des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques de la TST. Nous nous intéressons dans les chapitres suivants à l'opérationnalisation du formalisme des GU, selon deux points de vue différents :

– pour le raisonnement logique (chapitre 9) ;
– pour l'interopérationnalisation sur le Web des données (chapitre 10).

Quatrième partie

Vers une opérationnalisation du formalisme des Graphes d'Unités

CHAPITRE 9

Raisonnement dans le formalisme des Graphes d'Unités

***S**ocrate :*
"Mais moi, Hippias, je ne mets pas en doute que tu sois plus savant que moi ; j'ai cependant l'habitude, quand on dit quelque chose, d'y réfléchir, et tout particulièrement si celui qui parle me paraît savant, et comme j'aspire à apprendre ce qu'il dit, je l'interroge dans tous les sens et j'examine ses paroles de tous les côtés, en les comparant entre elles, afin d'apprendre."
Platon, Hippias mineur

Sommaire

Introduction

Nous avons donc étendu la conceptualisation des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques dans la TST. Nous avons ensuite introduit dans les chapitres 7 et 8 le formalisme des Graphes d'Unités, pour représenter ces connaissances. Le présent chapitre et le suivant s'intéressent maintenant à opérationnaliser ce formalisme.

Dans ce chapitre nous nous intéressons plus particulièrement à la question de recherche suivante :

Comment raisonner dans le formalisme des Graphes d'Unités ?

Nous divisons cette question en plusieurs sous-questions, qui structurent ce chapitre de la manière suivante.

Quelle sémantique formelle associer aux GU, et comment caractériser l'implication sémantique d'un GU par un autre ?
La section 9.1 introduit une sémantique formelle pour le formalisme des GU, basée sur la théorie des modèles et l'algèbre relationnelle.

Peut-on définir un ensemble de règles axiomatiques d'inférence pour expliciter les connaissances dans un GU ?
La section 9.2 introduit une telle base de règles, définie pour un support de GU donné.

Comment utiliser cette base de règles axiomatiques d'inférence pour développer des algorithmes de déduction ?
La section 9.3 fait usage de cette base de règles axiomatiques d'inférence pour définir la déduction d'un GU par un autre. Nous nous intéresserons plus particulièrement aux conditions de décidabilité de la déduction, et montrerons que le processus de déduction est correct vis-à-vis de la sémantique formelle, et complet dans certains cas.

9.1 Une sémantique d'interprétation pour les graphes d'unités

Dans cette section, nous associons au formalisme des GU une sémantique formelle basée sur la théorie des modèles et l'algèbre relationnelle.

Nous définirons la notion de modèle d'un support de GU (§9.1.1). Cette définition dépend de la définition de modèle d'une hiérarchie de types d'unité, et d'une hiérarchie de symboles de relation circonstancielle.

Nous préciserons d'abord la notion de modèle d'une hiérarchie de types d'unité sans définition de TUP (§9.1.2), et un modèle d'une hiérarchie de symboles de relation circonstancielle (§9.1.3).

Nous pourrons alors introduire la condition pour qu'un modèle satisfasse un GU, et la notion de conséquence sémantique d'une manière classique en sémantique des modèles : étant donnés deux GU G et H définis sur le même support, G implique H si tout modèle du support qui satisfait G satisfait aussi H (§9.1.4).

Finalement, nous définirons la satisfaction d'une règle, puis d'une définition de TUP par un modèle, et réviserons enfin la définition du modèle d'une hiérarchie des types d'unité pour prendre en compte un ensemble de définitions de TUP (§9.1.5).

9.1.1 Modèle d'un Support

Introduisons tout d'abord la définition de modèle d'un support. Soit D un ensemble non-vide nommé l'*univers*, ou *domaine* du modèle, qui contient l'ensemble des *unités*.

Définition 83 (Modèle d'un support)**.** Soit $\mathscr{S} = (\mathscr{T}, \mathscr{C}, \boldsymbol{M})$ un support.
Un modèle de $\mathscr{S}$ est un couple $M = (D, \delta)$.
D est appelé l'*univers* ou le *domaine* de M, et contient un ensemble d'*unités*. D contient au moins une unité, plus une unité spéciale notée $\bullet$, qui représente *rien*.
δ dénote la *fonction d'interprétation*, et doit être telle que :

- M est un modèle de $\mathscr{T}$;
- M est un modèle de $\mathscr{C}$;
- $\forall m \in \boldsymbol{M}, \delta(m) \in D$;

Nous introduirons dans les prochaines sections les notions nécessaires de modèle d'une hiérarchie de types d'unité, et de modèle d'une hiérarchie de SRelC.

9.1.2 Modèle d'une hiérarchie de types d'unité sans définition de TUP

La définition du modèle d'une hiérarchie de types d'unité nécessite la définition de la satisfaction d'un modèle par une définition de TUP, qui nécessite elle-même la définition de modèle d'une hiérarchie de types d'unité. Nous désactivons cette récursion en introduisant dans un premier temps la définition de modèle d'une hiérarchie de types d'unité sans définition de TUP.

Nous définirons l'interprétation des TUP (§9.1.2.1), puis l'étendrons aux TUC (§9.1.2.2), pour enfin définir le modèle d'une hiérarchie des types d'unité sans définition de TUP (§9.1.2.3).

9.1.2.1 Interprétation des TUP

Nous nous inspirons de la structure de la section 7.1 afin de définir l'interprétation des TUP et de leur structure actancielle.

Interprétation générale des TUP L'interprétation d'un TUP t est une relation sur le domaine D, notée $\delta(\{t\})$, avec l'ensemble des attributs $\{0\} \cup \boldsymbol{\alpha}(t)$:

- 0 apporte à t la sémantique d'une classe ;
- l'ensemble des attributs $\boldsymbol{\alpha}(t)$ apporte à t la sémantique complémentaire d'une relation.

Chaque tuple r de $\delta(\{t\})$ peut être identifié par une application, toujours notée r, de l'ensemble des attributs $\{0\} \cup \boldsymbol{\alpha}(t)$ vers l'univers D. r décrit comment une unité de type t est liée à ses actants. $r(0)$ est l'unité elle-même, et pour tout $s \in \boldsymbol{\alpha}(t)$, $r(s)$ est l'unité qui remplit la PosA s de $r(0)$. Si $r(s) = \bullet$, alors *il n'y a pas d'unité* qui remplit la PosA s de $r(0)$. Une unité donnée doit être décrite au maximum une fois dans $\delta(\{t\})$, donc 0 est une clé unique pour $\delta(\{t\})$.

Définition 84 (Interprétation des TUP)**.** L'interprétation d'un TUP $t \in \boldsymbol{T}$ est une relation $\delta(\{t\})$ avec un attribut primaire noté 0 ($0 \notin \boldsymbol{S}_{\mathscr{T}}$), et un attribut pour chacun de ses PosA dans $\boldsymbol{\alpha}(\{t\})$. 0 est une clé unique dans l'interprétation de tout TUP, i.e.,

$$\forall t \in \boldsymbol{T}, \forall r_1, r_2 \in \delta(\{t\}), \quad r_1(0) = r_2(0) \Rightarrow r_1 = r_2 \tag{9.1}$$

Proposition 9.1.1. *L'arité de l'interprétation de t est* $1 + valence(t)$.

$$\forall t \in \mathbf{T}, \delta(\{t\}) \subseteq D^{1+valence(t)} \tag{9.2}$$

Preuve: voir annexe, p. 314.

Maintenant, si une unité $i \in D$ est de type t_1, et que $t_1 \lesssim t_2$, alors par définition de la relation de spécialisation $\lesssim$ (cf., définition 31), i est également de type t_2. Si le TUP avait seulement l'interprétation d'une classe, alors cette condition pourrait être exprimée ainsi :

$$\delta(\{t_1\}) \subseteq \delta(\{t_2\})$$

Cependant, la description de i en tant que relation dans $\delta(\{t_2\})$ doit également être la même que la description de i dans $\delta(\{t_1\})$. Si les TUP avaient seulement la sémantique de relations de même arité et de mêmes attributs, cette condition pourrait être exprimée ainsi :

$$\delta(\{t_1\}) \subseteq \delta(\{t_2\})$$

Cependant, t_1 et t_2 ont tous deux la sémantique de classes et de relations, et peuvent avoir différentes structures actancielles. Comme nous savons que t_1 hérite des PosA de t_2 (cf., prop. 7.1.3),

nous savons aussi que les attributs que $\delta(\{t_1\})$ et $\delta(\{t_2\})$ ont en commun sont $\{0\} \cup \boldsymbol{\alpha}(t_2)$. Notons $\pi_A R$ la projection de la relation R sur l'ensemble d'attributs A. Ainsi l'interprétation de $\lesssim$ peut être définie comme suit :

Définition 85 (Interprétation de la relation de spécialisation sur l'ensemble des TUP)**.** La projection de $\delta(\{t_1\})$ sur les attributs de $\delta(\{t_2\})$ doit être une sous-relation de $\delta(\{t_2\})$, i.e.,

$$\forall t_1 \lesssim t_2, \pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)}\delta(\{t_1\}) \subseteq \delta(\{t_2\}) \tag{9.3}$$

Une conséquence directe de cette définition est que des TUP équivalents partagent la même interprétation.

Proposition 9.1.2. *Les TUP équivalents partagent la même interprétation, i.e.,*

$$\forall t_1 \simeq t_2, \delta(\{t_1\}) = \delta(\{t_2\}) \tag{9.4}$$

Preuve: voir annexe, p. 314.

Interprétation de la structure actancielle des TUP Maintenant, comme précisé par la définition 28, la racine de PosAObl d'un SRelA s a sa PosA s obligatoire. Donc la colonne s de $\delta(\{\boldsymbol{\gamma_1}(s)\})$ ne doit pas contenir $\bullet$. Au contraire, la racine de PosAInt s a sa PosA s interdite, donc la colonne s de $\delta(\{\boldsymbol{\gamma_0}(s)\})$ ne doit contenir que $\bullet$. Comme pour tout $s \in \boldsymbol{\alpha}(t)$, la projection $\pi_s\delta(\{t\})$ représente l'ensemble des unités qui remplissent la PosA s des unités de type t, alors l'interprétation des racines de PosAObl et de PosAInt est définie comme suit.

Définition 86 (Interprétation des racines de PosAObl et de PosAInt)**.** La colonne s de $\delta(\{\boldsymbol{\gamma_1}(s)\})$ ne contient pas $\bullet$, et la colonne s de $\delta(\{\boldsymbol{\gamma_0}(s)\})$ ne contient que $\bullet$ (ou est vide), i.e.,

$$\forall s \in \boldsymbol{S}_{\mathscr{T}}, \pi_s\delta(\{\boldsymbol{\gamma_1}(s)\}) \subseteq D \setminus \bullet \tag{9.5}$$

$$\forall s \in \boldsymbol{S}_{\mathscr{T}}, \pi_s\delta(\{\boldsymbol{\gamma_0}(s)\}) \subseteq \{\bullet\} \quad \text{i.e., } = \{\bullet\} \text{ ou } = \varnothing. \tag{9.6}$$

Ceci peut être généralisé à l'interprétation de tout TUP t. Si t possède une PosAObl s, alors la colonne s de $\delta(\{t\})$ ne contient pas $\bullet$. Au contraire, si t possède une PosAInt s, alors la colonne s de $\delta(\{t\})$ ne contient que $\bullet$.

Proposition 9.1.3. *Soit $t \in \mathbf{T}$. Les deux assertions suivantes sont vraies :*

- *Si $s \in \boldsymbol{\alpha_1}(t)$, alors la colonne s de $\delta(\{t\})$ ne contient pas $\bullet$:*

$$\forall t \in \mathbf{T}, \forall s \in \boldsymbol{\alpha_1}(t), \pi_s\delta(\{t\}) \subseteq D \setminus \bullet \tag{9.7}$$

- *Si $s \in \boldsymbol{\alpha_0}(t)$, alors la colonne s de $\delta(\{t\})$ ne contient que $\bullet$ (ou est vide) :*

$$\forall t \in \mathbf{T}, \forall s \in \boldsymbol{\alpha_0}(t), \pi_s\delta(\{t\}) \subseteq \{\bullet\} \tag{9.8}$$

Preuve: voir annexe, p. 314.

Finalement, pour toutes les unités de type t et pour toute PosA de t, l'unité qui remplit la PosA s doit être soit $\bullet$, soit une unité du type $\boldsymbol{\varsigma}_t(s)$:

Définition 87 (Interprétation des signatures de TUP)**.** La signature des TUP est interprétée comme suit :

$$\forall t \in \boldsymbol{T}, \forall s \in \boldsymbol{\alpha}(t), \pi_s\delta(\{t\}) \setminus \bullet \subseteq \pi_0\delta(\boldsymbol{\varsigma}_t(s)) \tag{9.9}$$

Notons que cette définition nécessite la définition de l'interprétation des TUC (une signature est un TUC), qui est introduite dans la section 9.1.2.2.

Interprétation du TUP universel premier et du TUP absurde premier Comme précisé par la définition 29, $\top$ est le type de toute unité, et $\bot$ n'est le type d'aucune unité. Puisque nous avons introduit l'unité spéciale *rien* $\bullet$, $\top$ est le type de toute unité sauf $\bullet$. Puisque la projection $\pi_0\delta(\{t\})$ sur l'attribut primaire 0 représente l'ensemble des unités qui ont le type t, l'interprétation de $\top$ et $\bot$ peut être définie comme suit.

Définition 88 (Interprétation du TUP universel premier et du TUP absurde premier)**.** Chaque unité sauf $\bullet$ est une instance de $\top$, et aucune unité n'est instance de $\bot$, i.e.,

$$\pi_0\delta(\{\top\}) = D \setminus \bullet \tag{9.10}$$
$$\pi_0\delta(\{\bot\}) = \varnothing \tag{9.11}$$

Avec la définition simple de l'interprétation de $\bot$, nous pouvons prouver que tout TUP absurde (cf., section 7.1.4) possède une interprétation vide, ce qui correspond à notre intuition.

Proposition 9.1.4. *Tout TUP absurde possède une interprétation vide, i.e.,*

$$\forall t \in A^{\sim}, \delta(\{t\}) = \varnothing \tag{9.12}$$

Preuve: voir annexe, p. 315.

9.1.2.2 Interprétation des TUC

Interprétation générale des TUC et TUC déclarés absurdes L'interprétation des TUP est étendue à une interprétation des TUC en utilisant l'opérateur de jointure.

Définition 89 (Interprétation d'un TUC)**.** L'interprétation d'un TUC $t^{\cap}$ est la jointure de l'interprétation de ses TUP constituants, sauf pour la TUC $\varnothing$ qui possède la même interprétation que $\top^{\cap}$, i.e.,

$$\forall t^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing, \delta(t^{\cap}) = \bowtie_{t \in t^{\cap}} \delta(\{t\}) \tag{9.13}$$

$$\delta(\varnothing) = \delta(\{\top\}) \tag{9.14}$$

Il est trivial de vérifier que cette définition est cohérente pour les TUC primitifs :

$$\forall t \in \boldsymbol{T}, \bowtie_{t \in \{t\}} \delta(\{t\}) = \delta(\{t\})$$

L'interprétation de $\varnothing$ est le seul point problématique ici. En fait, $\varnothing$ est équivalent à $\top^{\cap}$ (cf., définition 47 et proposition 7.2.4), donc dans l'idéal nous aimerions prouver que leur interprétation est égale. Cependant, $\varnothing$ ne possède aucune PosA d'après la définition 42, alors que $\top^{\cap}$ peut en avoir. Nous permettons à ces deux TUC d'avoir la même interprétation en définissant arbitrairement l'interprétation de $\varnothing$ comme étant égale à celle de $\top^{\cap}$.

Nous vérifions dans les propositions suivantes que l'extension des interprétations des TUP aux interprétations des TUC est cohérente.

Proposition 9.1.5. *L'interprétation d'un TUC non vide $t^\cap \in \mathbf{T}^\cap \setminus \varnothing$ est une relation d'arité $1 + valence(t^\cap)$, et d'attributs $\{0\} \cup \boldsymbol{\alpha}^\cap(t^\cap)$, i.e.,*

$$\forall t^\cap \in \mathbf{T}^\cap \setminus \varnothing, \delta(t^\cap) \subseteq D^{1+valence(t^\cap)} \tag{9.15}$$

Preuve: voir annexe, p. 315.

Proposition 9.1.6. *0 est une clé unique pour l'interprétation de tout TUC $t^\cap \in \mathbf{T}^\cap$, i.e.,*

$$\forall t^\cap \in \mathbf{T}^\cap, \forall r_1, r_2 \in \delta(t^\cap), r_1(0) = r_2(0) \Rightarrow r_1 = r_2 \tag{9.16}$$

Preuve: voir annexe, p. 315.

Proposition 9.1.7. *Aucun TUC n'est le type de •, i.e.,*

$$\forall t^\cap \in \mathbf{T}^\cap, \pi_0\delta(t^\cap) \subseteq D \setminus \varnothing \tag{9.17}$$

Preuve: voir annexe, p. 315.

Soit s un attribut commun aux relations R_1 et R_2. L'opérateur de jointure a la propriété intéressante suivante :

$$\pi_s\delta(R_1 \bowtie R_2) \subseteq \pi_s R_1 \cap \pi_s R_2$$

Nous montrons que cette propriété est renforcée pour l'attribut 0, de par la cohérence de l'interprétation de la relation de spécialisation sur l'ensemble des TUP avec leur structure actancielle, et puisque 0 est une clé unique.

Proposition 9.1.8.

$$\pi_0\delta(t^\cap) = \bigcap_{t \in t^\cap} \pi_0\delta(\{t\}) \tag{9.18}$$

Preuve: voir annexe, p. 316.

Interprétation des PosA obligatoires et interdites Nous montrons que l'interprétation des PosA obligatoires et interdites des TUP s'étend de manière cohérente aux TUC.

Proposition 9.1.9. *Les deux points suivants sont vrais :*

$$\forall t^\cap \in \mathbf{T}^\cap, \forall s \in \boldsymbol{\alpha}_{\boldsymbol{1}}^\cap(t^\cap), \pi_s\delta(t^\cap) \subseteq D \setminus \bullet \tag{9.19}$$

$$\forall t^\cap \in \mathbf{T}^\cap, \forall s \in \boldsymbol{\alpha}_{\boldsymbol{0}}^\cap(t^\cap), \pi_s\delta(t^\cap) \subseteq \{\bullet\} \tag{9.20}$$

Preuve: voir annexe, p. 316.

Interprétation des signatures Nous montrons que l'interprétation des signatures des TUP s'étend de manière cohérente aux TUC.

Proposition 9.1.10. *La définition 87 peut être étendue aux TUC, i.e.,*

$$\forall t^\cap \in \mathbf{T}^\cap, \forall s \in \boldsymbol{\alpha}^\cap(t^\cap), \pi_s\delta(t^\cap) \setminus \bullet \subseteq \pi_0\delta(\boldsymbol{\varsigma}_{t^\cap}^\cap(s)) \tag{9.21}$$

Preuve: voir annexe, p. 317.

TUC absurdes Finalement, nous imposons que tout TUC déclaré absurde $t^{\cap} \in \bot_A^{\sqcap}$ possède une interprétation vide.

Définition 90 (Interprétation d'un TUC déclaré absurde)**.** Les TUC déclarés absurdes possèdent une interprétation vide, i.e.,

$$\forall t^{\cap} \in \bot_A^{\sqcap}, \delta(t^{\cap}) = \varnothing \tag{9.22}$$

Nous montrons ainsi que l'interprétation du préordre sur les TUC s'étend de manière cohérente à l'interprétation du préordre sur les TUC, pourvu qu'ils soient réguliers (i.e., l'ensemble $\boldsymbol{T}^{\cap}_{regular}$, cf., définition 52).

Proposition 9.1.11. *La définition 85 peut être étendue aux TUC réguliers, i.e.,*

$$\forall t_1^{\cap} \in \mathbf{T}^{\cap}_{regular}, \forall t_2^{\cap} \in \mathbf{T}^{\cap} \setminus \varnothing, t_1^{\cap} \overset{\cap}{\lesssim} t_2^{\cap} \implies \pi_{\{0\} \cup \boldsymbol{\alpha}^{\cap}(t_2^{\cap})} \delta(t_1^{\cap}) \subseteq \delta(t_2^{\cap}) \tag{9.23}$$

Si $t_2^{\cap} = \varnothing$, alors l'ensemble des attributs de son interprétation est l'ensemble des attributs de $\delta(\top^{\cap})$. Ainsi :

$$\forall t_1^{\cap} \in \mathbf{T}^{\cap} - \bot^{\sqcap}, \pi_{\{0\} \cup \boldsymbol{\alpha}^{\cap}(\top^{\cap})} \delta(t_1^{\cap}) \subseteq \delta(\varnothing) \tag{9.24}$$

Preuve: voir annexe, p. 317.

Cette proposition se limite aux TUC réguliers. En effet, Le TUC vide, pose problème puisque l'ensemble des attributs de $\delta(\varnothing)$ n'est pas $\{0\} \cup \boldsymbol{\alpha}^{\cap}(\varnothing)$. De plus, tout TUC absurde $t_1^{\cap}$ peut poser problème puisque l'ensemble des PosA de $t_1^{\cap}$ peut être différent de l'ensemble des PosA d'un TUC $t_2^{\cap}$ plus générique (cf., proposition 7.3.9).

Finalement, nous pouvons montrer que tout TUC absurde possède bien une interprétation vide.

Proposition 9.1.12. *Tout TUC absurde possède une interprétation vide, i.e.,*

$$\forall t^{\cap} \in \bot^{\sqcap}, \delta(t^{\cap}) = \varnothing \tag{9.25}$$

Preuve: voir annexe, p. 317.

9.1.2.3 Modèle d'une hiérarchie de types d'unité

Nous pouvons maintenant définir le modèle d'une hiérarchie de types d'unité sans définition de TUP.

Définition 91 (Modèle d'une hiérarchie de types d'unité sans définition de TUP)**.** Soit une hiérarchie de types d'unité sans définition de TUP $\mathscr{T} = \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Gamma_1}, \boldsymbol{\gamma_1}, \boldsymbol{\Gamma_0}, \boldsymbol{\gamma_0}, C_A, \bot_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}} \rangle$. Un modèle de $\mathscr{T}$ est un couple $M = (D, \delta)$ tel que la fonction d'interprétation δ satisfait les définitions 84 à 90.

9.1.3 Modèle d'une hiérarchie de symboles circonstanciels

Afin que le modèle d'une hiérarchie de types d'unité soit également un modèle d'une hiérarchie de SRelC, la fonction d'interprétation δ doit être étendue et contrainte comme suit.

Définition 92 (Interprétation des SRelC)**.** La fonction d'interprétation δ associe à chaque SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$ une relation binaire $\delta(s)$ avec deux attributs : *gov* qui signifie *gouverneur*, et *circ* qui signifie *circonstant*.

$$\forall s \in \boldsymbol{S}_{\mathscr{C}}, \delta(s) \subseteq (D \setminus \bullet)^2, \text{ une relation d'attributs } \{gov, circ\}; \tag{9.26}$$

Parallèlement aux relations binaires de la sémantique des modèle des relations binaires du formalisme des GC, si une SRelC s_1 est plus spécifique qu'une autre SRelC s_2, alors l'interprétation de s_1 soit être inclue dans l'interprétation de s_2.

Définition 93 (Interprétation de la relation de spécialisation sur les SRelC)**.** L'interprétation d'un sous-SRelC s_1 doit être inclue dans l'interprétation se son super-SRelC s_2.

$$\forall s_1, s_2 \in \boldsymbol{S}_{\mathscr{C}}, s_1 \overset{\mathscr{C}}{\lesssim} s_2 \Rightarrow \delta(s_1) \subseteq \delta(s_2) \tag{9.27}$$

Finalement, le type des unités qui sont liées via une relation de SRelC s doit correspondre à la signature de s.

Définition 94 (Interprétation des signatures de SRelC)**.** La signature des SRelC est interprétée comme suit :

$$\forall s \in \boldsymbol{S}_{\mathscr{C}}, \pi_{gov}\delta(s) \subseteq \pi_0\delta(domaine(s)); \tag{9.28}$$
$$\forall s \in \boldsymbol{S}_{\mathscr{C}}, \pi_{circ}\delta(s) \subseteq \pi_0\delta(codomaine(s)); \tag{9.29}$$

Nous pouvons maintenant définir le modèle d'une hiérarchie de SRelC.

Définition 95 (Modèle d'une hiérarchie de symboles de relation circonstancielle)**.** Soit une hiérarchie de SRelC $\mathscr{C} = \langle \boldsymbol{S}_{\mathscr{C}}, \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}, \mathscr{T}, \{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}} \rangle$. Un modèle de $\mathscr{C}$ est un modèle $M = (D, \delta)$ de $\mathscr{T}$ tel que la fonction d'interprétation δ satisfait les définitions 92 à 94.

9.1.4 Modèle satisfaisant un GU et conséquence sémantique

Maintenant que le modèle d'un support de GU est complètement défini et caractérisé, nous pouvons définir le modèle d'un GU. Un modèle d'un GU est un modèle du support sur lequel il est défini, augmenté d'une application d'*affectation* sur les nœuds d'unité qui affecte à chaque nœud d'unité un élément de $D \setminus \bullet$.

Définition 96 (Modèle d'un GU)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle$ un GU défini sur un support $\mathscr{S}$. Un modèle de G est un triplet (D, δ, β) où :
- (D, δ) est un modèle de $\mathscr{S}$;
- β, nommé *affectation*, est une application de U vers $D \setminus \bullet$.

Afin de satisfaire le GU, l'application d'affectation β doit satisfaire un ensemble de conditions. Tout d'abord, si un nœud d'unité $u \in U$ possède un marqueur $m \in marker(u)$, alors l'affectation de u doit correspondre à l'interprétation de m.

$$\forall u \in U, \forall m \in marker(u), \beta(u) = \delta(m) \tag{9.30}$$

Ensuite, l'affectation d'un nœud d'unité u doit appartenir à l'ensemble des unités qui ont le type $type(u)$.

$$\forall u \in U, \beta(u) \in \pi_0 \delta(type(u)) \tag{9.31}$$

Pour tout triplet actanciel $(u, s, v) \in A$, et comme $\{\boldsymbol{\gamma}(s)\}$ est le TUC qui introduit une PosA s, l'interprétation $\delta(\{\boldsymbol{\gamma}(s)\})$ doit refléter le fait que l'unité représentée par v remplit la position actancielle s de l'unité représentée par u.

$$\forall (u, s, v) \in A, (\beta(u), \beta(v)) \in \pi_{0,s} \delta(\{\boldsymbol{\gamma}(s)\}) \tag{9.32}$$

De manière similaire, pour tout triplet circonstanciel $(u, s, v) \in C$, l'interprétation de s doit refléter le fait que l'unité représentée par v dépend de l'unité représentée par u par rapport à s.

$$\forall (u, s, v) \in C, (\beta(u), \beta(v)) \in \delta(s) \tag{9.33}$$

Finalement, si deux nœuds d'unité sont déclarés équivalents, alors l'unité qu'ils représentent est la même, et leur affectation doit donc être la même.

$$\forall (u_1, u_2) \in Eq, \beta(u_1) = \beta(u_2) \tag{9.34}$$

Nous pouvons maintenant définir la notion de satisfaction d'un GU par un modèle.

Définition 97 (Modèle satisfaisant un GU)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle$ un GU défini sur un support $\mathscr{S}$, et (D, δ, β) un modèle de G. (D, δ, β) est un *modèle satisfaisant* G, noté $(D, \delta, \beta) \vDash G$, si β est une affectation qui satisfait les conditions 9.30 à 9.34.

En utilisant la notion de modèle d'un support et de modèle de GU, il est possible de définir l'implication entre GU comme suit.

Définition 98 (Implication et équivalence)**.** Soient H et G deux GU définis sur un support $\mathscr{S}$.

- G *implique* H, ou H est une *conséquence sémantique* de G, noté $G \models H$, si et seulement si pour tout modèle (D, δ) de $\mathscr{S}$ et pour tout affectation β^g des nœuds d'unité de G telle que $(D, \delta, \beta^g) \models G$, il existe une affectation β^h des nœuds d'unité de H telle que $(D, \delta, \beta^h) \models H$.
- H et G sont *sémantiquement équivalents*, noté $H \equiv G$, si et seulement si $H \models G$ et $G \models H$.

Ainsi, les informations portées par un GU correspondent à des informations partielles sur le modèle, et nous nécessiterons des algorithmes pour expliciter les connaissances dans le modèle tout en respectant ces contraintes.

Finalement, nous pouvons définir les GU absurdes comme étant ceux qui ne sont satisfaits par aucun modèle.

Définition 99 (UG absurde)**.** Soit G un GU défini sur un support $\mathscr{S}$. G est dit *absurde* si et seulement si aucun modèle ne satisfait G.

Montrons que tout GU qui possède un nœud d'unité de type absurde est absurde.

Proposition 9.1.13. *Soit $G = \langle U, \mathbf{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$. S'il existe un nœud d'unité $u \in U$ tel que $type(u) \in \bot^{\sqcap}$, alors G est absurde.*

Preuve: voir annexe, p. 319.

Le GU absurde premier $G_\bot$ est donc bien absurde. Par ailleurs, puisqu'un GU absurde G ne possède aucun modèle, il implique tout autre GU H.

9.1.5 Modèle d'une hiérarchie de types d'unité avec définitions de TUP

Maintenant que nous avons défini l'implication sémantique, nous revenons au modèle de la hiérarchie des types d'unité afin d'y imposer la satisfaction d'un ensemble de définitions de TUP.

Définissons d'abord la notion de satisfaction d'une règle. Intuitivement, un modèle satisfait une règle si pour toute affectation β^h qui rend l'hypothèse satisfaite, il existe une affectation β^c telle que a) la conclusion est satisfaite, et b) l'affectation d'un nœud d'unité distingué de l'hypothèse est égale à l'affectation du nœud d'unité distingué correspondant dans la conclusion.

Définition 100 (Satisfaction d'une règle par un modèle)**.** Soit $R = (H, C, \kappa)$ une règle définie sur un support $\mathscr{S}$, avec $H = \{u_1^h, \ldots, u_n^h\}\langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h\rangle$ et $C = \{u_1^c, \ldots, u_n^c\}\langle U^c, \boldsymbol{l}^c, A^c, C^c, Eq^c\rangle$. Soit $M = (D, \delta)$ un modèle de $\mathscr{S}$. *M satisfait la règle R*, noté $(D, \delta) \models R$, si et seulement si pour toute affectation β^h telle que (D, δ, β^h) satisfait H, il existe une affectation β^c telle que les deux points suivants sont vrais :

- le modèle (D, δ, β^c) satisfait C ;
- $\forall u^h \in \{u_1^h, \ldots, u_n^h\}, \beta^c(\kappa(u^h)) = \beta^h(u^h)$;

Il est alors direct de définir la satisfaction d'une définition de TUP, puisqu'elle correspond à deux règles réciproques.

Définition 101 (Satisfaction d'une définition de TUP par un modèle)**.** Soit $D_t = (D_t^-, D_t^+, \kappa)$ une définition d'un TUP t défini sur un support $\mathscr{S}$. Soit $M = (D, \delta)$ un modèle de $\mathscr{S}$. *M satisfait la définition D_t*, noté $(D, \delta) \models D_t$ si et seulement si les deux points suivants sont vrais :

- M satisfait la règle (D_t^-, D_t^+, κ) ;
- M satisfait la règle $(D_t^+, D_t^-, \kappa^{-1})$;

Ainsi, un modèle d'une hiérarchie d'unité $\mathscr{T}$ est un modèle de la hiérarchie d'unité sans TUP correspondante, qui satisfait de surcroît l'ensemble des règles d'expansion et de contraction des définitions de TUP que $\mathscr{T}$ contient.

9.2 Une base de règles pour expliciter les connaissances des GU

Dans le formalisme des graphes d'unités, un GU représente des connaissances explicites, et d'autres connaissances implicites peuvent être rendues explicites. Nous avons déjà vu dans la section 8.2.3 certains mécanismes d'inférence qui permettent d'expliciter les connaissances dans les GU.

Nous nous proposons donc de représenter ces différents mécanismes d'inférence à l'aide d'un ensemble de règles axiomatiques. Cette section est organisée suivant les différentes facettes des graphes d'unités :

- la section 9.2.1 définit un ensemble de règles qui concernent la relation d'équivalences déclarées et les étiquettes des nœuds d'unité ;
- la section 9.2.2 étudie les mécanismes d'inférence possibles avec les types d'unité et leurs structures actancielles ;
- la section 9.2.3 introduit les règles d'inférence qui concernent les relations circonstancielles.

Augmenté de l'ensemble des règles d'expansion et de contraction associées aux définitions de TUP (cf., définition 80), cet ensemble de règles définit une base de règles axiomatiques d'inférence, que nous récapitulerons dans la section 9.2.4.

9.2.1 Règles sur la relation d'équivalences déclarées et les étiquettes

Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un graphe d'unité.

Fermeture réflexo-symétrico-transitive de la relation d'équivalences déclarées Tout d'abord, la relation d'équivalences déclarées peut être complétée pour réellement former une relation d'équivalence entre les nœuds d'unité, à l'aide des règles suivantes :

eq-ref tout nœud d'unité $u \in U$ peut être déclaré équivalent à lui-même ;

eq-sym si u est déclaré équivalent à v, alors v peut être déclaré équivalent à u ;

eq-trans si u_1 est déclaré équivalent à u_2, et que u_2 est déclaré équivalent à u_3, alors u_1 peut être déclaré équivalent à u_3.

Ces règles sont illustrées sur la figure 9.1.

Marqueurs des nœuds d'unité et relation d'équivalences déclarées Ensuite, nous savons que deux nœuds d'unité ayant un marqueur commun représentent la même unité. De tels nœuds peuvent alors être déclarés équivalents. De plus, un marqueur peut se propager suivant la relation d'équivalences déclarées. Ainsi pour chaque identifiant d'unité $m \in \boldsymbol{M}$, les règles suivantes explicitent ces connaissances :

mrk-eq si $m \in marker(u_1)$ et $m \in marker(u_2)$, alors les nœuds d'unité u_1 et u_2 peuvent être déclarés équivalents ;

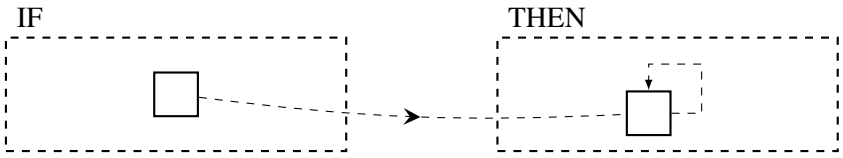
eq-mrk si $m \in marker(u_1)$ et $(u_1, u_2) \in Eq$, alors on peut ajouter m aux marqueurs du nœud d'unité u_2.

Ces règles sont illustrées sur la figure 9.2.

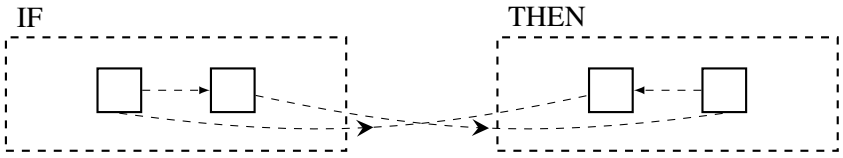
Propagation des types d'unité De la même manière, un type d'unité peut se propager suivant la relation d'équivalences déclarées. Pour chaque TUP $t \in \boldsymbol{T}$, la règle suivante explicite cette connaissance :

eq-typ si $t \in type(u_1)$ et $(u_1, u_2) \in Eq$, alors on peut ajouter t aux types du nœud d'unité u_2.

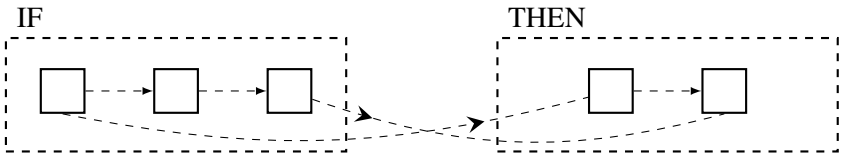
Cette règle est illustrée par la figure 9.3.



(a) Règle **eq-ref**

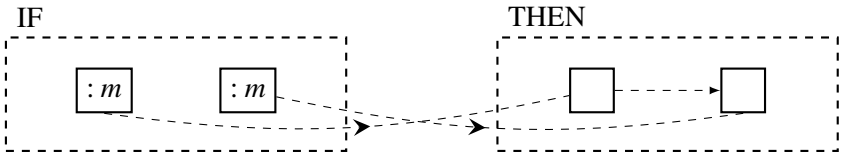


(b) Règle **eq-sym**

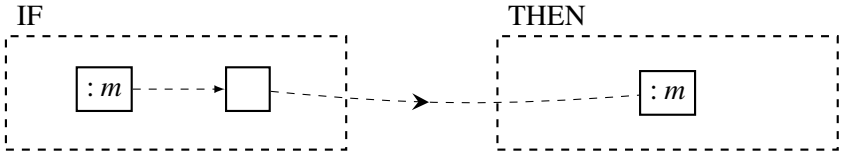


(c) Règle **eq-trans**

FIGURE 9.1 – Règles d'inférence : fermeture réflexo-symétrico-transitive de la relation d'équivalences déclarées

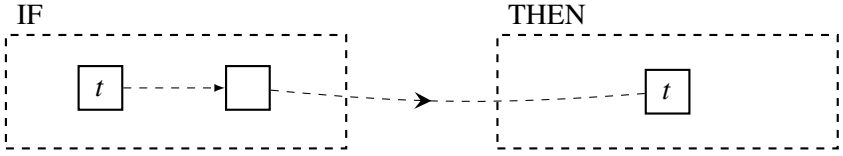


(a) Règle **mrk-eq**, pour tout $m \in \boldsymbol{M}$



(b) Règle **eq-mrk**, pour tout $m \in \boldsymbol{M}$

FIGURE 9.2 – Règles d'inférence : marqueurs des nœuds d'unité et relation d'équivalences déclarées



(a) Règle **eq-typ**, pour tout $t \in \boldsymbol{T}$

FIGURE 9.3 – Règles d'inférence : propagation des types d'unité

9.2.2 Règles sur les types d'unité et leurs structures actancielles

Soit $G = \langle U, \boldsymbol{I}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un graphe d'unité.

Comparaisons des TUP Le TUP universel premier $\top$ est le type de toute unité. Ensuite, la connaissance minimale qui permet de construire une hiérarchie des types d'unité est l'ensemble des comparaisons déclarées de TUP C_A. Quelques comparaisons nécessaires supplémentaires étendent cet ensemble en un ensemble de comparaisons de TUP $\boldsymbol{C_T}$ (cf., définition 32). Si $(t_2, t_1) \in \boldsymbol{C_T}$, alors toute unité de type t_1 est également de type t_2. Finalement, tout nœud d'unité dont le type contient un TUC déclaré absurde rend le GU absurde.

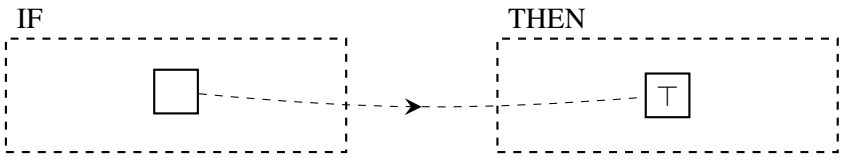
Pour tout nœud d'unité $u \in U$, les trois règles suivantes explicitent ces connaissances :

typ-top on peut ajouter $\top$ à $type(u)$.

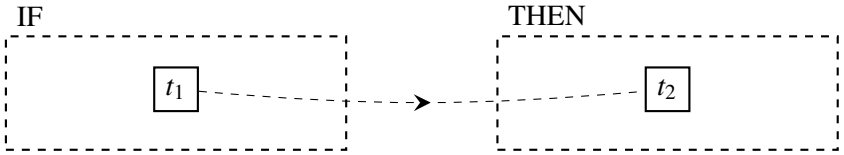
typ-comp pour tout $(t_2, t_1) \in \boldsymbol{C_T}$, si $t_1 \in type(u)$, alors on peut ajouter t_2 aux types du nœud d'unité u.

typ-absurd pour tout $t_a^{\cap} \in \bot^{\sqcap}$ et $t_a^{\cap} \subseteq type(u)$, alors on peut ajouter $\bot$ à $type(u)$.

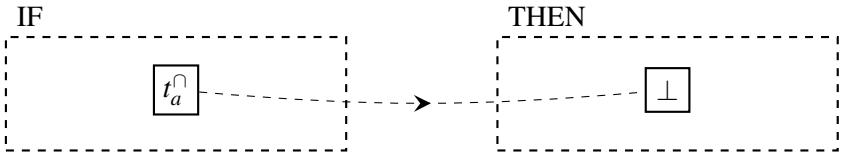
Ces règles sont illustrées sur la figure 9.4.



(a) Règle **typ-top**



(b) Règle **typ-comp**, pour tout $(t_2, t_1) \in \boldsymbol{C_T}$



(c) Règle **typ-absurd**, pour tout $t_a^{\cap} \in \bot_A^{\sqcap}$

FIGURE 9.4 – Règles d'inférence : comparaisons des TUP

Relations actancielles et relation d'équivalences déclarées Pour tout SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$, si un nœud d'unité u_1 est déclaré équivalent à un autre nœud d'unité u_2 impliqué dans une relation actancielle s avec un nœud d'unité v, alors u_1 est également impliqué dans une relation actancielle avec v. Pour tout SRelA. De plus, une seule unité peut prendre une PosA d'une autre unité.

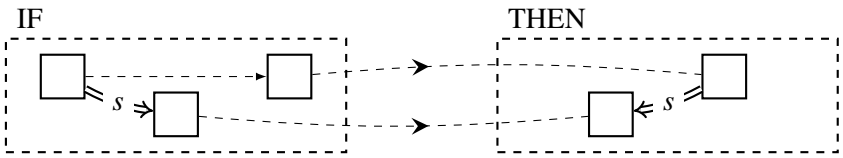
Pour tout SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$, les trois règles suivantes explicitent ces connaissances :

a-eq-g pour tout $(u_1, s, v) \in A$ et $(u_1, u_2) \in Eq$, on peut ajouter (u_2, s, v) à A ;

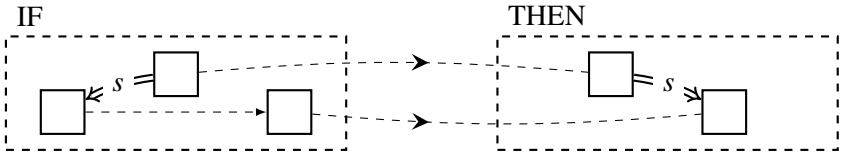
a-eq-a pour tout $(u, s, v_1) \in A$ et $(v_1, v_2) \in Eq$, on peut ajouter (u, s, v_2) à A ;

a-fp pour tout (u, s, v_1) et $(u, s, v_2) \in A$, on peut ajouter (v_1, v_2) à Eq.

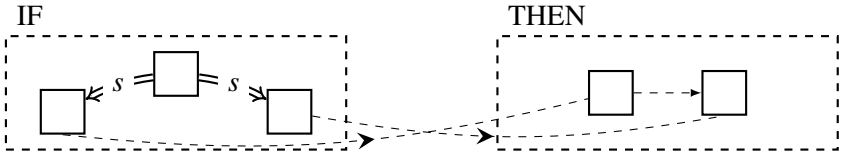
Ces règles sont illustrées sur la figure 9.5.



(a) Règle **a-eq-g**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$



(b) Règle **a-eq-a**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$



(c) Règle **a-fp**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$

FIGURE 9.5 – Règles d'inférence : relations actancielles et relation d'équivalences déclarées

Racines de PosA, de PosAObl, et de PosAInt Pour tout SRelA, si un nœud u_1 possède une PosA s, alors il est de type $\boldsymbol{\gamma}(s)$. S'il est de type $\boldsymbol{\gamma_1}(s)$, alors il possède une PosA s. Enfin, s'il est de type $\boldsymbol{\gamma_0}(s)$ et qu'il possède néanmoins une PosA, alors il est absurde.

Pour tout SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$, les trois règles suivantes explicitent ces connaissances :

a-aslot-root pour tout $(u,s,v) \in A$, on peut ajouter $\boldsymbol{\gamma}(s)$ à $type(u)$;

a-oblaslot-root pour tout $u \in U$ et $s \in \boldsymbol{S}_{\mathscr{T}}$, si $\boldsymbol{\gamma_1}(s) \in type(u)$, on peut expliciter un nouveau nœud d'unité v à U et ajouter (u,s,v) à A ;

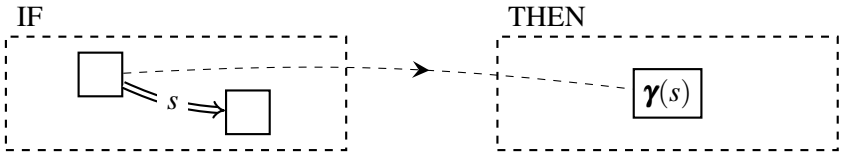
a-proaslot-root pour tout $(u,s,v) \in A$, si $\boldsymbol{\gamma_0}(s) \in type(u)$, alors on peut ajouter $\bot$ à $type(u)$.

Ces règles sont illustrées sur la figure 9.6.

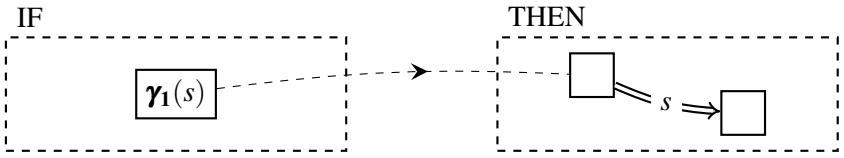
Signatures Enfin, pour tout TUP $t \in \boldsymbol{T}$, les unités qui prennent une PosA $s \in \boldsymbol{\alpha}(t)$ sont du type $\boldsymbol{\varsigma}_t(s)$. La règle suivante explicite ces connaissances :

a-sig pour tout $(u,s,v) \in A$, si $t \in type(u)$ alors on peut ajouter $\boldsymbol{\varsigma}_t(s)$ à $type(v)$.

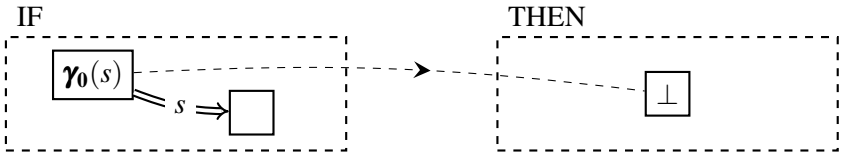
Cette règle est illustrée par la figure 9.7a.



(a) Règle **a-aslot-root**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$

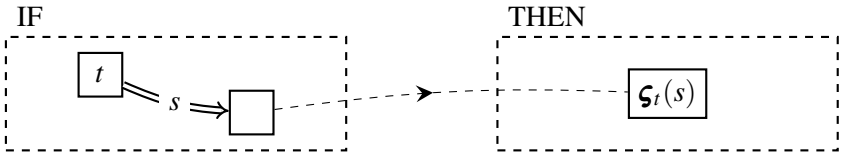


(b) Règle **a-oblaslot-root**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$



(c) Règle **a-proaslot-root**, pour tout $s \in \boldsymbol{S}_{\mathscr{T}}$

FIGURE 9.6 – Règles d'inférence : racines de PosA, de PosAObl, et de PosAInt



(a) Règle **a-sig**, pour tout $t \in \boldsymbol{T}$ et $s \in \boldsymbol{\alpha}(t)$

FIGURE 9.7 – Règles d'inférence : signatures

9.2.3 Règles sur les relations circonstancielles

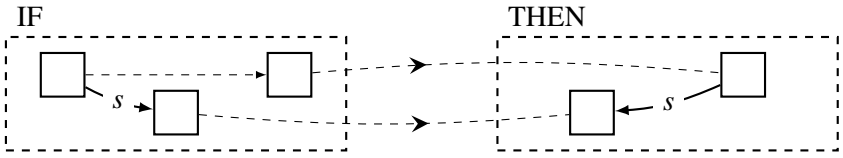
Soit $G = \langle U, \boldsymbol{I}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un graphe d'unité.

Relations circonstancielles et relation d'équivalences déclarées Pour tout SRelC $c \in \boldsymbol{S}_{\mathscr{T}}$, si un nœud d'unité u_1 est déclaré équivalent à un autre nœud d'unité u_2 impliqué dans une relation circonstancielle s avec un nœud d'unité v, alors u_1 est également impliqué dans une relation circonstancielle avec v. Pour tout SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$, les deux règles suivantes explicitent ces connaissances :

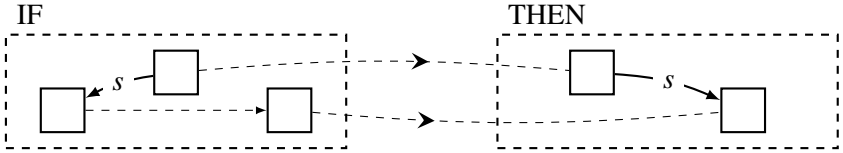
c-eq-g pour tout $(u_1, s, v) \in C$ et $(u_1, u_2) \in Eq$, on peut ajouter (u_2, s, v) à C ;

c-eq-a pour tout $(u, s, v_1) \in C$ et $(v_1, v_2) \in Eq$, on peut ajouter (u, s, v_2) à C ;

Ces règles sont illustrées sur la figure 9.8.



(a) Règle **c-eq-g**, pour tout $s \in \boldsymbol{S}_{\mathscr{C}}$



(b) Règle **c-eq-c**, pour tout $s \in \boldsymbol{S}_{\mathscr{C}}$

FIGURE 9.8 – Règles d'inférence : relations circonstancielles et relation d'équivalences déclarées

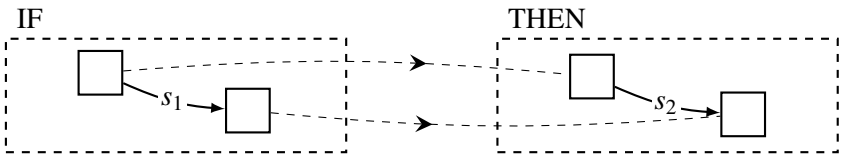
Comparaisons et signatures des SRelC Pour tout triplet $(u, s_1, v) \in C$, s'il existe une comparaison déclarée de SRelC $(s_2, s_1) \in \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$, alors on peut ajouter le triplet $\langle u, s_2, v \rangle$ à C. De plus, le nœud d'unité u est du type $domaine(s)$ et le nœud d'unité v est du type $codomaine(s)$.

Pour tout SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$, les deux règles suivantes explicitent ces connaissances :

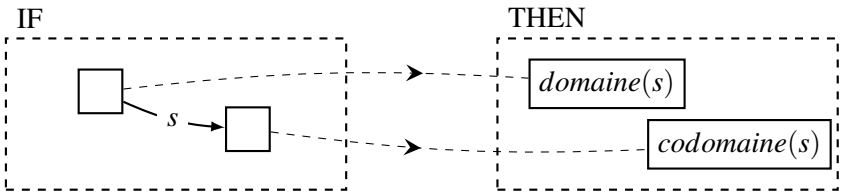
c-comp pour tout $(s_2, s_1) \in \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$, si $(u, s_1, v) \in C$, alors on peut ajouter (u, s_2, v) à C.

c-sig pour tout $(u, s, v) \in C$, on peut ajouter $domaine(s)$ à $type(u)$, et $codomaine(s)$ à $type(v)$;

Ces règles sont illustrées sur la figure 9.9.



(a) Règle **c-comp**, pour tout $(s_2, s_1) \in \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$



(b) Règle **c-sig**, pour tout $s \in \boldsymbol{S}_{\mathscr{C}}$

FIGURE 9.9 – Règles d'inférence : comparaisons et signatures des SRelC

9.2.4 Base de règles axiomatiques d'inférence d'un support

Nous pouvons maintenant définir, pour un support de GU, l'ensemble des règles axiomatiques d'inférence.

Définition 102 (Base de règles axiomatiques d'inférence d'un support)**.** Pour un support de GU $\mathscr{S} = \langle \mathscr{T}, \mathscr{C}, \boldsymbol{M} \rangle$, la *base de règles axiomatiques d'inférence* $\mathscr{R}(\mathscr{S})$ est composée de :

- **eq-ref**, **eq-sym**, **eq-trans** ;
- pour tout identifiant d'unité $m \in \boldsymbol{M}$, **mrk-eq**, **eq-mrk** ;
- pour tout TUP $t \in \boldsymbol{T}$, **eq-typ** ;
- **typ-top** ;
- pour tout couple de TUP comparés $(t_2, t_1) \in \boldsymbol{C}_{\boldsymbol{T}}$, **typ-comp** ;
- pour tout TUC déclaré absurde $t_a^{\sqcap} \in \bot_A^{\sqcap}$, **typ-absurd** ;
- pour tout SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$, **a-eq-g**, **a-eq-a**, **a-fp**, **a-aslot-root**, **a-oblaslot-root**, **a-proaslot-root** ;
- pour tout PosA de tout TUP $a \in \boldsymbol{\alpha}(t)$, **a-sig** ;
- pour tout SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$, **c-eq-g**, **c-eq-c**, **c-sig** ;
- pour tout couple de SRelC déclarés comparables $(s_2, s_1) \in \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$, **c-comp**.
- pour toute définition de TUP $D_t \in \mathscr{D}$, **def+** et **def-**.

Nous étudierons dans la section suivante comment faire usage de cette base de règles axiomatiques d'inférence pour raisonner avec les GU. D'un autre côté, nous nous intéresserons dans le chapitre prochain à simuler ces règles avec les formalismes du Web Sémantique (avec SPARQL).

9.3 Dérivation et déduction basée sur les règles

Maintenant que nous avons défini une sémantique formelle pour le formalisme des GU, et que nous avons introduit une base de règles axiomatiques d'inférence, cette section étudie dans quelle mesure ces règles peuvent être utilisées pour raisonner dans le formalisme des GU.

Le problème de décision premier du formalisme des GU est : *Considérant deux GU G et H définis sur le même support $\mathscr{S}$, les connaissances de G impliquent-elles les connaissances de H ?*

Nous introduirons donc dans un premier temps les notions de dérivation et de déduction pour les GU (§9.3.1). Nous étudierons ensuite des conditions suffisantes de décidabilité du processus de déduction, selon que la hiérarchie des types d'unité contienne des définitions de TUP ou non (§9.3.2). Finalement, nous montrerons que le processus de déduction est correct vis-à-vis de la sémantique formelle (§9.3.3), et complet dans certains cas.

9.3.1 Dérivation et déduction par chaînage avant

Nous ajoutons une garde aux règles afin d'éviter qu'elles produisent plusieurs fois la même connaissance. Nous révisons donc la définition 75 pour les règles axiomatiques d'inférence.

Définition 103 (Applicabilité d'une règle axiomatique d'inférence)**.** Soit $G \in \mathscr{G}(\mathscr{S})$, $R = \langle H, C, \kappa \rangle \in \mathscr{R}(\mathscr{S})$ une règle axiomatique d'inférence, et $C = \{u_1^c, \ldots, u_n^c\}C'$ la conclusion de R.
R est applicable à un GU G selon une application π, si et seulement si :

- π est un homomorphisme de H vers G ;
- il n'existe pas d'homomorphisme π_c de C vers G tel que $\pi_{c|\{u_1^c,\ldots,u_n^c\}} = \pi \circ \kappa^{-1}$.

Soit $G \in \mathscr{G}(\mathscr{S})$ un GU. Soit G' le GU obtenu par l'application de l'une des règles de $\mathscr{R}(\mathscr{S})$. G' explicite certaines connaissances qui sont implicites dans G. Nous définissons ainsi la dérivation immédiate des GU. La dérivation d'un GU est le GU produit d'une séquence d'application de règles de $\mathscr{R}(\mathscr{S})$.

Définition 104 (Dérivation immédiate et dérivation de GU)**.** Soit $\mathscr{S}$ un support de GU, $G, G' \in \mathscr{G}(\mathscr{S})$, et $\mathscr{R}(\mathscr{S})$ la base de règles axiomatiques d'inférences.
G' dérive immédiatement de G s'il est obtenu de G par l'application de l'une des règles de $\mathscr{R}(\mathscr{S})$. G' est une dérivation de G s'il existe une séquence $G_0(=G), \ldots, G_n(=G')$ telle que pour tout $0 \leq i < n$, G_{i+1} dérive immédiatement de G_i.

Nous pouvons alors définir la déduction dans le formalisme des GU.

Définition 105 (Déduction dans le formalisme des GU)**.** Soit $\mathscr{S}$ un support de GU. Soit $\mathscr{R}(\mathscr{S})$ sa base de règles axiomatiques d'inférence. Soient deux GU $H, G \in \mathscr{G}(\mathscr{S})$.
H peut être déduit de G, si et seulement s'il existe un homomorphisme π de H vers une dérivation G' de G ;

GU absurde par déduction Nous introduisons l'ensemble des GU *absurdes par déduction*, comme étant l'ensemble des GU à partir desquels le GU absurde premier $G_\perp$ peut être déduit (cf., définition 63).

Définition 106 (GU absurde par déduction)**.** Soit G un GU défini sur un support $\mathscr{S}$. G est dit *absurde* si et seulement si le GU absurde premier $G_\perp$ peut être déduit de G.

Une étude précise des GU absurdes fera l'objet d'un travail futur de cette thèse, en particulier :
- nous conjecturons qu'un GU absurde par déduction puisse-être dérivé à l'infini, à condition que le support $\mathscr{S}$ possède au moins un SRelA s, et si par exemple $\boldsymbol{\varsigma}_\perp(s) = \perp^\cap$.
- nous conjecturons que les notions de GU absurde et de GU absurde par déduction sont équivalentes.

Dans le reste de ce chapitre, nous nous limitons aux GU qui ne sont pas absurdes (ni absurdes par déduction).

9.3.2 Conditions suffisantes de décidabilité de la déduction : existence d'une clôture finie

Dans cette section, nous cherchons une condition suffisante sur le support $\mathscr{S}$, qui assurerait que le problème de déduction soit décidable. Nous souhaitons plus particulièrement que la condition soit acceptable pour les linguistes de la TST.

Puisque le problème de déduction est défini par un homomorphisme, et que la recherche d'homomorphisme est décidable sur l'ensemble des graphes finis, alors nous cherchons une condition nécessaire et suffisante sur un support $\mathscr{S}$, qui nous assure que tout GU G non absurde défini sur $\mathscr{S}$ possède une expansion finie.

Nous définirons dans un premier temps la notion de clôture des GU (§9.3.2.1). Ensuite, dans l'ensemble des règles axiomatiques d'inférence d'un support de GU $\mathscr{S}$, seules deux ensembles de règles peuvent ajouter des nœuds d'unité lorsqu'elles sont appliquées : **a-oblaslot-root**, et **def+**. La section 9.3.2.2 propose donc une condition suffisante sur la hiérarchie des types d'unité sans définitions de TUP, et nous étudierons enfin l'apport des définitions de TUP dans la section 9.3.2.3.

9.3.2.1 Graphe d'unité clos et modèle canonique

Considérant la garde que nous imposons à l'application des règles axiomatiques d'inférence, une limite d'expansion G' est atteinte si aucune de ces règles ne peut s'appliquer à G'. Si tel est le cas, aucune règle axiomatique d'inférence ne peut expliciter de connaissances, et G' est dit *clos*.

Définition 107 (Graphe d'unité clos)**.** Un GU $G \in \mathscr{G}(\mathscr{S})$ est clos si et seulement si aucune des règles de la base de règles axiomatiques d'inférence $\mathscr{R}(\mathscr{S})$ n'est applicable.

Proposition 9.3.1. *Dans un GU clos $G \in \mathscr{G}(\mathscr{S})$, la relation d'équivalences déclarées Eq est une relation d'équivalence.*

Preuve: voir annexe, p. 319.

L'ensemble des classes d'équivalence de nœuds d'unité définit une partition de U. Soit $u \in U$, nous notons $[u]$ la classe d'équivalence à laquelle u appartient, i.e., $[u] \stackrel{\text{def}}{=} \{u' \in U \mid (u,u') \in Eq\}$.

Définition 108 (Ensemble des classes d'équivalence de nœuds d'unité d'un GU clos)**.** Soit $G \in \mathscr{G}(\mathscr{S})$ un GU clos. L'*ensemble des classes d'équivalence de nœuds d'unité* est le quotient de l'ensemble U par Eq, i.e., $U/Eq = \{[u] \mid u \in U\}$

Nous savons qu'une condition suffisante à la décidabilité du problème de déduction d'un graphe H par un graphe G est que G possède un GU dérivé G' clos. Nous dirons dans un tel cas que G possède une *expansion finie*.

Finalement, nous aurons besoin dans la section 9.3.3.2 de définir un modèle canonique pour les GU clos. La définition suivante introduit un tel modèle.

Définition 109 (Modèle canonique d'un GU clos)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un GU clos. Le modèle canonique M de G est le modèle $M = (D, \delta, [\cdot])$ défini comme suit :

Le domaine est l'ensemble des classes d'équivalence de nœuds d'unité de G, plus l'élément *rien* :

$$D = U/Eq \cup \{\bullet\}$$

L'affectation d'un nœud d'unité est sa classe d'équivalence :

$$\beta(u) = [u]$$

L'interprétation d'un identifiant d'unité $m \in \boldsymbol{M}$ est définie comme suit :
- $\delta(m) = [u]$ s'il existe $u \in U$ tel que $m \in marker(u)$;
- $\delta(m) = \bullet$ sinon.

L'interprétation d'un TUP $t \in \boldsymbol{T}$ est une relation $\delta(\{t\})$, qui possède un tuple pour chaque classe d'équivalence $[u] \in U/Eq$ dont les nœuds d'unité ont un type qui contient t. Soit r un tel tuple défini pour une classe d'équivalence $[u]$. On définit r pour n'importe quel membre de cette classe d'équivalence $u \in [u]$ comme suit :
- $r(0) = [u]$;
- pour tout $s \in \boldsymbol{\alpha}(t)$,
  - S'il existe un triplet actanciel $(u, s, v) \in A$, alors $r(s) = [v]$.
  - Sinon, $r(s) = \bullet$.

Nous construisons l'interprétation des TUC selon la définition 89.

L'interprétation d'un SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$ est une relation binaire $\delta(s)$ définie comme suit :

$$\forall s \in \boldsymbol{S}_{\mathscr{C}}, \delta(s) = \{([u], [v]) \mid \exists (u, s, v) \in C\}$$

Proposition 9.3.2. *Le modèle canonique M d'un GU clos G satisfait effectivement G, i.e., $(D, \delta, [\cdot]) \vDash G$.*

Preuve: voir annexe, p. 319.

9.3.2.2 Condition sur la hiérarchie des types d'unité sans définitions de TUP

Dans cette section, nous considérons seulement les hiérarchies de types d'unité qui ne possèdent pas de définition de TUP. Dans l'ensemble des règles axiomatiques d'inférence, seules les règles **a-oblaslot-root** ajoutent un nœud d'unité au GU. C'est donc une suite infinie d'applications de ces règles qui mène à une expansion infinie.

L'exemple simple de la figure 9.10 illustre une expansion infinie pour un GU fini. Supposons que l'on déclare qu'un TUP $/\text{person}\backslash$ possède une PosA obligatoire *mother* de signature $\boldsymbol{\varsigma}_{/\text{person}\backslash}(mother) = /\text{woman}\backslash$, et qu'évidemment, $/\text{woman}\backslash \lesssim /\text{person}\backslash$. Considérons le GU $G = (\{u\}, \boldsymbol{l}, \varnothing, \varnothing, \{(u,u)\})$, tel que $marker(u) = Peter$ et $type(u) = \{/\text{person}\backslash\}$.

La règle **u-typ** rendra u de type $\{\top, {}^{/}\text{person}^{\backslash}, \boldsymbol{\gamma}(mother), \boldsymbol{\gamma_1}(mother)\}$, et on sait que l'unité représentée par u devrait avoir une unité de type ${}^{/}\text{woman}^{\backslash}$ qui remplit sa PosA obligatoire *mother*. Donc la règle **a-oblaslot-root** s'applique, et on peut ajouter un nœud d'unité v pour représenter cet actant, avec $(u, mother, v) \in A$. La règle **u-typ** rend alors v de type $\{\top, {}^{/}\text{person}^{\backslash}, {}^{/}\text{woman}^{\backslash}, \boldsymbol{\gamma}(mother), \boldsymbol{\gamma_1}(mother)\}$, et la règle **a-oblaslot-root** est à nouveau applicable sur v. Ainsi $cl(G)$ possède une chaîne infinie de nœuds d'unité de type $\{\top, {}^{/}\text{person}^{\backslash}, \boldsymbol{\gamma}(mother), \boldsymbol{\gamma_1}(mother)\}$ et qui remplissent la PosA *mother* l'un de l'autre.

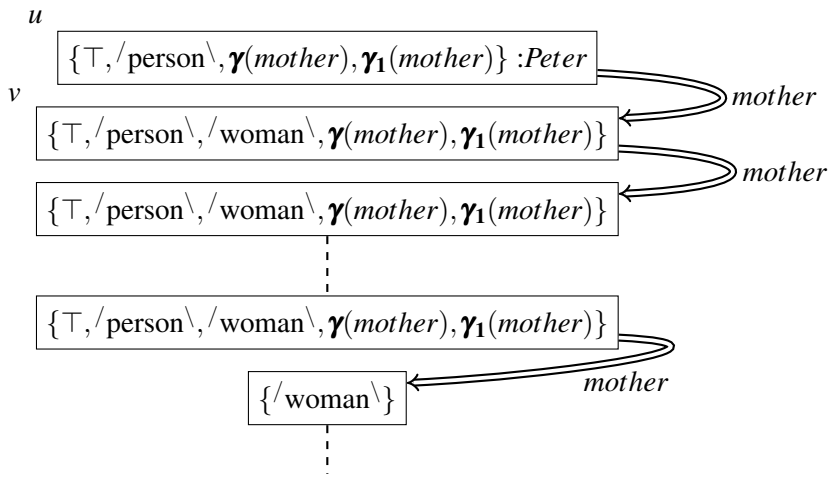


FIGURE 9.10 – Illustration d'une expansion infinie d'un GU, causée par les signatures

Divisons le processus d'expansion en deux expansions partielles :

- L'expansion partielle cl_1 prend en compte toutes les règles sauf **a-oblaslot-root** (et **def+** et **def-**). Elle résulte en un nouveau GU $cl_1(G)$.
- L'expansion partielle cl_2 ne prend en compte que la règle **a-oblaslot-root**. Elle résulte en un nouveau GU $cl_2(G)$.

Le processus suivant est une alternative au processus d'expansion de GU, dans le cas où il n'y a pas de définition de TUP.

Définition 110 (Processus itératif d'expansion d'un GU)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq\rangle \in \mathscr{G}(\mathscr{S})$ un GU non absurde. Considérons la suite $(G_i)_{0\leq i}$ telle que $G_0 = G$, et pour tout i,

$$G_{i+1} = cl_1 \circ cl_2(G_i) \tag{9.35}$$

Soit G un GU fini non absurde. Trivialement, si $G_{i+1} = G_i$, alors G_i est clos, et l'expansion est finie. Donc l'expansion de G est infinie si la suite $(G_i)_{0\leq i}$ ne converge pas. Supposons que $cl_2(G_i) \neq G_i$. Alors la règle **a-oblaslot-root** a ajouté un nœud d'unité v et un triplet (u, s, v) dans G_i. Montrons que le type de v dans G_{i+1} est la clôture de la signature de $type(u)$ pour s.

Proposition 9.3.3. *Soit $G = \langle U, \mathbf{l}, A, C, Eq\rangle \in \mathscr{G}(\mathscr{S})$ un GU non absurde.*
Si $cl_2(G_i)$ ajoute un nœud d'unité v et un arc (u, s, v) dans G_i, alors le type de v dans $cl_1 \circ cl_2(G_i)$ est $\Uparrow \boldsymbol{\varsigma}^{\cap}_{type(u)}(s)$.

Preuve: voir annexe, p. 320.

Ensuite, montrons que le type des nœuds d'unité déjà présents dans G_i ne change pas dans G_{i+1}.

Proposition 9.3.4. *Soit $(G_i = \langle U_i, \mathbf{l}_i, A_i, C_i, Eq_i \rangle)_{i \geq 0}$ la suite du processus itératif de clôture d'un GU G non absurde. Considérons $i \geq 1$. Si $u \in U_i \cap U_{i+1}$, alors $type_i(u) = type_{i+1}(u)$.*

Preuve: voir annexe, p. 321.

Finalement, le lemme suivant précise que le mécanisme qui implique l'apparition d'un nœud d'unité dans G_{i+1} est récursif.

Lemme 9.3.5. *Soit $(G_i = \langle U_i, \mathbf{l}_i, A_i, C_i, Eq_i \rangle)_{i \geq 0}$ la suite du processus itératif de clôture d'un GU G non absurde. Considérons $i \geq 2$. S'il existe un nœud d'unité $u_{i+1} \in U_{i+1} - U_i$, alors il existe un nœud d'unité $u_i \in U_i - U_{i-1}$ et un SRelA $s_i \in \mathbf{S}_{\mathscr{T}}$ tels que $\boldsymbol{\gamma}(s_i) \in type_i(u_i)$, et $(u_i, s_i, u_{i+1}) \in A_{i+1} - A_i$.*

Preuve: voir annexe, p. 321.

Nous pouvons finalement conclure sur la condition nécessaire et suffisante pour qu'un GU non absurde possède une expansion finie, si la hiérarchie des types d'unité n'a pas de définition de TUP. Considérons le graphe orienté :

$$G_{\mathscr{T}} = (\boldsymbol{T}^{\Uparrow}, \left\{ (t^{\cap}, \Uparrow \boldsymbol{\varsigma}^{\cap}_{t^{\cap}}(s)) \right\}_{t^{\cap} \in \boldsymbol{T}^{\Uparrow} \setminus \boldsymbol{T}, s \in \boldsymbol{\alpha}^{\cap}_{\mathbf{1}}(t^{\cap})}) \tag{9.36}$$

Théorème 9.3.6 (Condition nécessaire et suffisante pour une expansion finie, sans définition de TUP)**.** *Soit $\mathscr{S} = (\mathscr{T}, \mathscr{C}, \mathbf{M})$ un support, la hiérarchie des types d'unité $\mathscr{T}$ étant sans définition de TUP, et $\mathbf{S}_{\mathscr{T}} \neq \varnothing$. Tout GU non absurde possède une expansion finie si et seulement si $G_{\mathscr{T}}$ est acyclique.* □

Preuve: voir annexe, p. 321.

La figure 9.11 illustre le graphe $G_{\mathscr{T}}$ de l'exemple de la figure 9.10. Ce graphe possède une boucle élémentaire, et GU G possède donc une expansion infinie.

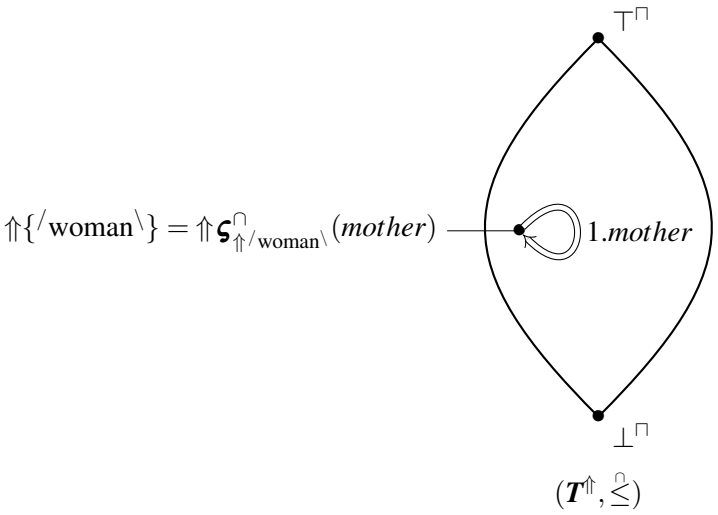


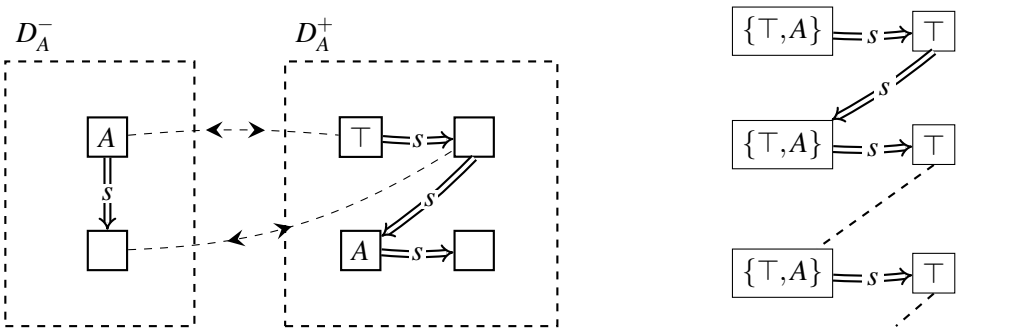
FIGURE 9.11 – Illustration d'un cycle dans le graphe $G_{\mathscr{T}}$.

La restriction que nous mettons ainsi en évidence pour les signatures dans la hiérarchie des types d'unité semble acceptable pour les lexicographes de la TST. En effet, jusqu'à présent les signatures des PosASém sont des étiquettes sémantiques, qui ne possèdent pas de structure actancielle. Notre formalisme permet de gagner en expressivité en associant une structure actancielle à ces étiquettes sémantiques, tout en définissant un critère à respecter pour assurer la décidabilité de la déduction.

9.3.2.3 Condition sur une hiérarchie des types d'unité avec définitions de TUP

Considérons maintenant les hiérarchies de types d'unité avec définitions de TUP. Dans l'ensemble des règles axiomatiques d'inférence, seules les règles **a-oblaslot-root** et **def+** ajoutent des nœuds d'unité au GU lorsqu'elles sont appliquées. C'est alors une suite infinie d'applications de ces règles qui mène à une expansion infinie.

L'exemple simple de la figure 9.12 illustre une expansion infinie pour un GU, avec pour seule cause la définition de TUP illustrée par la figure 9.12a. Dans cet exemple, $T_D = \{A\}$, $C_A = \varnothing$, $\boldsymbol{S}_{\mathscr{T}} = \{s\}$, et $\boldsymbol{\varsigma}_A(s) = \varnothing$.



(a) Définition du TUP A.

(b) Exemple d'expansion infinie

FIGURE 9.12 – Illustration d'une expansion infinie d'un GU, causée par un cycle définitoire de TUP

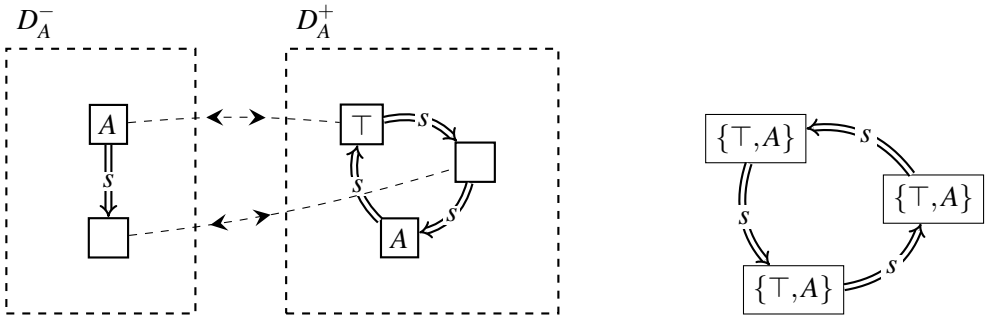
Dans cet exemple problématique, A entre dans la définition de A. Nous sommes donc en présence d'un cycle définitoire élémentaire. Si l'on devait dégager une condition suffisante à ce qu'aucun GU non absurde ne possède une expansion infinie, nous nous orienterions vers un critère d'acyclicité dans l'ensemble des définitions : il ne doit pas exister de cycle définitoire $(D_0, \ldots, D_n = D_0)$ tel que pour tout i, il existe un nœud d'unité u dans D_i^+ dont le type contienne le genus de D_{i+1}. Cette condition d'acyclicité est intéressante à deux points de vue pour les travaux futurs de cette thèse :

- C'est cette condition qu'impose Leclère (1998) à une base de définitions de concepts et de relations pour les Graphes Conceptuels. Par ailleurs, d'autres travaux concernés par le raisonnement à base de règles ont montré que ce type d'acyclicité est une condition suffisante pour assurer la décidabilité du problème de décision (Baget et Salvat, 2006; Baget *et al.*, 2009) ;
- Il s'agit d'une condition que les lexicographes s'imposent habituellement dans la conception des dictionnaires.

Par la combinaison de ces deux points, nous conjecturons que l'acyclicité définitoire, en conjonction de l'acyclicité du graphe $G_{\mathscr{T}}$, est une condition acceptable par les lexicographes, et suffisante pour assurer la finitude des expansions, et donc pour assurer la décidabilité du problème de décision.

Cependant, nous présentons ci-dessous deux exemples de transgression de ces restrictions, pour lesquels tout GU non absurde possède néanmoins une expansion finie.

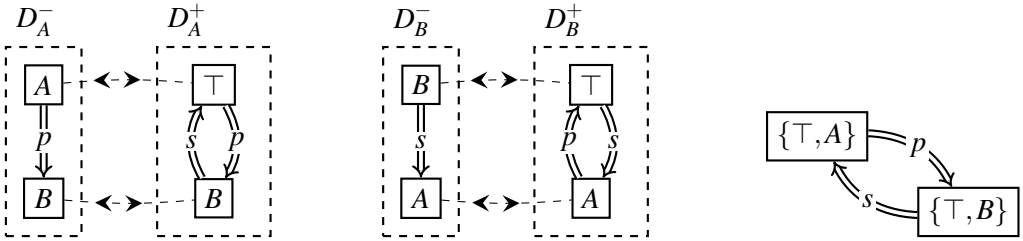
Le premier exemple est illustré sur la figure 9.13, et présente un cycle définitoire élémentaire : A est défini en fonction de A. Le GU D_A^- de la figure 9.13a possède l'expansion finie représentée sur la figure 9.13b.



(a) Définition du TUP A. (b) Exemple d'expansion finie

FIGURE 9.13 – Illustration d'une expansion finie d'un GU, malgré un cycle définitoire de TUP

Le second exemple est illustré sur la figure 9.14. Il présente un cycle définitoire de longueur 2 : A est défini en fonction de B (fig. 9.14a), et B en fonction de A (fig. 9.14b). De plus, le graphe $G_{\mathscr{T}}$ possède un cycle, puisque $\varsigma_A(p) = B$, et $\varsigma_B(s) = A$. Néanmoins, les GU D_A^- et D_B^- possèdent une expansion finie représentée sur la figure 9.14b.



(a) Définition du TUP A. (b) Définition du TUP B. (c) Exemple d'expansion finie

FIGURE 9.14 – Illustration d'une expansion finie d'un GU, malgré un cycle définitoire de TUP, et un cycle dans le graphe $G_{\mathscr{T}}$

Contrairement à la section précédente où nous avions mis en lumière une condition nécessaire est suffisante pour assurer la finitude des expansions, ces deux exemples prouvent que l'acyclicité de $G_{\mathscr{T}}$ et l'acyclicité définitoire ne seraient qu'une condition suffisante pour assurer la finitude des expansions, et qu'il serait possible de trouver une condition moins restrictive, qui satisferait d'autant plus les lexicographes.

9.3.3 Correction et complétude du chaînage avant

Nous montrons finalement dans cette section que :
- La déduction est correcte par rapport à la conséquence sémantique (§9.3.3.1) ;
- La déduction est complète par rapport à la conséquence sémantique, pour les GU qui possèdent une expansion finie (§9.3.3.2).

9.3.3.1 Preuve de la correction

Soient $G,H \in \mathscr{G}(\mathscr{S})$. Montrons que si H est déductible de G, alors $G \vDash H$. La démonstration de ce théorème comporte deux étapes :

1. S'il existe un homomorphisme de H vers G', alors $G' \vDash H$ (lemme 9.3.7) ;
2. Si G' est une dérivation de G, alors $G \vDash G'$ (lemme 9.3.8).

Ainsi si H est déductible de G, alors il existe un homomorphisme π de H vers une dérivation G' de G, et $G \vDash G' \vDash H$.

Lemme 9.3.7. *Soient $G,H \in \mathscr{G}(\mathscr{S})$ deux GU. S'il existe un homomorphisme de H vers G, alors $G \vDash H$.*

Preuve: voir annexe, p. 321.

Lemme 9.3.8. *Soient $G,G' \in \mathscr{G}(\mathscr{S})$. Si G' est une dérivation de G, alors $G \vDash G'$.*

Preuve: voir annexe, p. 322.

Théorème 9.3.9. *La déduction est correcte par rapport à la sémantique formelle, i.e., pour tout $G,H \in \mathscr{G}(\mathscr{S})$ si H peut être déduit de G, alors $G \vDash H$.* □

Preuve: voir annexe, p. 325.

9.3.3.2 Preuve de la complétude, dans le cas d'une expansion finie

Soient $G,H \in \mathscr{G}(\mathscr{S})$. Montrons que si $G \vDash H$, et que G possède une expansion finie, alors H peut être déduit de G. On montre plus particulièrement qu'il existe un homomorphisme π de H vers une expansion finie G' de G. La démonstration de ce théorème comporte deux étapes :

1. Si G' est clos, et $G' \vDash H$, alors il existe un homomorphisme de H vers G' (lemme 9.3.10) ;
2. Si G' est une dérivation de G, alors $G' \vDash G$ (lemme 9.3.11.

Ainsi si $G \vDash H$, et que G possède une expansion finie G', on a $G' \vDash G \vDash H$, et comme G' est clos, il existe un homomorphisme de H vers G'.

Lemme 9.3.10. *Soient $G,G' \in \mathscr{G}(\mathscr{S})$. Si G' est une dérivation de G, alors $G' \vDash G$.*

Preuve: voir annexe, p. 325.

Lemme 9.3.11. *Soit $G,H \in \mathscr{G}(\mathscr{S})$. Si G est clos, et $G \vDash H$, alors il existe un homomorphisme de H vers G.*

Preuve: voir annexe, p. 326.

Théorème 9.3.12. *La déduction est complète par rapport à la sémantique formelle pour les GU qui possèdent une expansion finie, i.e., pour tout $G,H \in \mathscr{G}(\mathscr{S})$, si G possède une expansion finie, et que $G \models H$, alors H est déductible de G* □

Preuve: voir annexe, p. 326.

Conclusion

Nous avons introduit les bases du raisonnement dans le formalisme des GU.

Nous avons associé au support et aux GU une sémantique formelle basée sur la théorie des modèles et l'algèbre relationnelle. L'originalité de notre approche est l'utilisation d'un élément particulier du domaine : l'unité rien, notée $\bullet$, qui permet d'expliciter qu'aucune unité ne prend une PosA donnée. Nous avons alors défini l'implication sémantique d'un GU par un autre, et le caractère absurde d'un GU. La fonction d'interprétation doit satisfaire un ensemble de conditions pour que le modèle satisfasse un GU. Les informations portées par un GU correspondent alors à des informations partielles du modèle, et nous nécessiterons des algorithmes pour expliciter les connaissances dans le modèle tout en respectant ces contraintes.

Nous avons ensuite défini un ensemble de règles de GU, qui explicitent chacune des connaissances dans les GU. Nous avons ainsi défini une base de règles axiomatiques d'inférence pour le formalisme des GU.

Nous avons étudié comment cette base de règles peut être utilisée pour raisonner avec les homomorphismes de GU. Nous avons ainsi défini la déduction d'un GU par un autre, et le caractère absurde par déduction d'un GU.

La section 9.3.2 a étudié les conditions de décidabilité du problème de déduction dans le formalisme des GU. Nous avons d'abord étudié le cas particulier des GU clos, sur lesquels aucune des règles axiomatiques d'inférence n'a d'effet. Nous avons alors justifié que si tout GU non absurde possède un GU dérivé fini et clos (une expansion finie), alors le problème de déduction est décidable. Nous avons alors mis en lumière une condition nécessaire est suffisante sur un support sans définition de TUP pour que tout GU non absurde possède une expansion finie, et nous avons discuté de conjectures dans le cas où le support posséderait des définitions de TUP.
Finalement, nous avons montré dans la section 9.3.3 que le problème de déduction est correct vis-à-vis de la sémantique formelle, et complet dans le cas où les GU possèdent une expansion finie.

Ce chapitre a fait l'objet de deux publications dans des conférences internationales : (Lefrançois et Gandon, 2013b) et (Lefrançois et Gandon, 2013d). Les directions futures de ce travail sur le raisonnement dans le formalisme des GU incluent :

- une étude plus précise de la complexité des tâches de raisonnement ;
- le développement et l'implémentation d'algorithmes efficaces pour accomplir ces tâches, en particulier chaînage avant et chaînage arrière ;

- un approfondissement de l'étude des conditions de finitude de l'expansion des GU dans le cas où le support contiendrait des définitions de TUP, et l'évaluation auprès des linguistes de la pertinence de ces conditions.
- il est possible que le raisonnement reste décidable sous certaines conditions et pour certaines questions précises, malgré une expansion infinie. Un point de départ est les travaux de Cuenca Grau *et al.* (2013), qui proposent des conditions d'acyclicités pour assurer la finitude du chaînage avant.
- La complétude de la déduction pourrait être prouvée indépendamment du fait que le problème soit décidable ou pas.

Le chapitre suivant s'intéresse maintenant à la modélisation possible du formalisme des GU avec les formalismes du Web Sémantique. Nous nous intéresserons en particulier à représenter la base de règles axiomatiques d'inférence en SPARQL.

CHAPITRE 10

Opérationnalisation sur le Web des Données

*_____ **L**a connaissance non partagée n'a pas vraiment son utilité.*
Daniel Desbiens, Maximes d'aujourd'hui, 2008

Sommaire

Introduction

Le formalisme des GU permet donc de représenter des ressources lexicales d'un nouveau type, en particulier les prédicats linguistiques et les définitions lexicographiques formelles. Nous avons justifié dans la section 2.3.1 l'importance de proposer une transformation de ce formalisme vers le modèle RDF, afin de profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances. De plus, nous avons mis en lumière dans la section 2.4.2 l'intérêt de lier le modèle obtenu au modèle **ontolex** de représentation des connaissances lexicales du groupe communautaire Ontology-Lexica.

Ce chapitre répond donc à la question de recherche suivante :

Comment opérationnaliser le formalisme des Graphes d'Unités sur le Web des données lexicales ?

La suite de ce chapitre est organisée de la manière suivante. Nous choisissons le métamodèle et l'architecture globale de notre modèle dans la section 10.1. La section 10.2 développe ensuite le modèle générique des Graphes d'Unités. La section 10.3 étudie plus spécifiquement la modélisation des définitions lexicographiques formelles introduites dans le chapitre 4. Enfin, nous proposons dans la section 10.4 une première instanciation de ce modèle pour la TST, que nous alignons avec le modèle **ontolex**.

10.1 Choix de l'architecture globale du modèle

Nous devons choisir un métamodèle et une architecture globale pour notre modèle. Nous justifions tout d'abord le choix du métamodèle OWL 2 RL (§10.1.1). Nous justifierons ensuite la nécessité de développer deux modèles hétérogènes mais interopérables (§10.1.2). Enfin, nous proposerons une architecture pyramidale des modèles, du modèle générique des GU au modèle spécifique d'une représentation linguistique particulière (§10.1.3).

Notons que les exemples de RDF dans ce chapitre sont écrits avec la syntaxe RDF 1.1 TriG, qui étend la syntaxe RDF Turtle et permet la représentation des graphes nommés de RDF 1.1. Chacun des exemples fait implicitement usage des préfixes usuels du Web Sémantique ci-dessous, et des préfixes du modèle **ontolex** et du vocabulaire **isocat**. Les règles SPARQL font également implicitement usage de ces préfixes usuels.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix isocat: <http://www.isocat.org/datcat/> .
@prefix ontolex: <http://www.w3.org/ns/ontolex#> .
@prefix ontolex-synsem: <http://www.w3.org/ns/ontolex-synsem#> .
```

10.1.1 Choix du métamodèle OWL 2 RL

Nous souhaitons assurer l'interopérabilité de notre modèle avec d'autres ontologies du Web des données.

Bien que nous ayons montré dans la section 6.1 que les logiques de description ne permettent pas de représenter les définitions lexicographiques formelles de manière naturelle, elles permettent assez bien de représenter les prédicats linguistiques. Nous nous proposons donc d'utiliser OWL comme base pour notre étude. Nous voyons deux avantages supplémentaires importants à ce choix :

- la communauté du Web Sémantique est familière avec OWL et sa sémantique ;
- le modèle **ontolex** proposé par le groupe Ontology-Lexica est une ontologie OWL.

Gardons néanmoins à l'esprit que le formalisme des Graphes d'Unités possède une sémantique différente de celle des logiques de description, et qu'il s'opérationnalise à l'aide d'une base de règles axiomatiques d'inférence. Nous choisissons donc de limiter notre usage de OWL de sorte que notre modèle passe les tests de conformité de l'un des sous-langages de OWL, et compléter la capacité de raisonnement à l'aide d'un ensemble de règles SPARQL d'inférence.

Le sous-langage qui se prête le mieux à ces exigences est OWL 2 RL [1], dont la sémantique est partiellement axiomatisée par une base de règles nommée OWL 2 RL/RDF [2]. Un travail préliminaire a donc été de réécrire cette base de règles en SPARQL. Nous nous sommes inspirés du travail de Holger Knublauch [3]. Le résultat de ce travail préliminaire est présenté en annexe C.

Ainsi le modèle que nous proposons est acceptable pour les raisonneurs des logiques de description, et nous étendons la base de règles OWL 2 RL/RDF en un ensemble homogène de règles SPARQL utilisable par un raisonneur du Web sémantique à base de règles tel que Corese (cf., Corby *et al.*, 2004).

10.1.2 Deux modèles interopérables

Nous choisissons donc de nous conformer au fragment syntaxique OWL 2 RL, qui contraint la syntaxe de OWL 2 DL. Nous devons donc nous conformer à la séparation T-box vs. A-box des logiques de description, qui impose grossièrement deux niveaux de description : le niveau des classes, et le niveau des instances.

Lorsque l'on décrit un dictionnaire, il est naturel de considérer que les types d'unité et leurs positions actancielles sont des instances. C'est d'ailleurs le choix des différents modèles existants sur le Web des données lexicales, en particulier le modèle **ontolex** du groupe communautaire Ontology-Lexica. Afin de faciliter l'alignement avec le modèle **ontolex**, nous proposons donc une première modélisation selon ce point de vue, que nous nommons **gu-langue**.

A l'inverse, lorsqu'on décrit des graphes d'unités, il est naturel de considérer que les unités sont les instances, que les types d'unité sont les classes, et que les unités sont interconnectées par des relations actancielles et circonstancielles. Nous proposons donc une seconde modélisation selon ce point de vue, que nous nommons **gu-usage**.

1. OWL 2 RL - http://www.w3.org/TR/owl2-profiles/#OWL_2_RL
2. OWL 2 RL/RDF rules - http://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules
3. OWL 2 RL in SPARQL - http://topbraid.org/spin/owlrl-all.html

Le tableau 10.1 résume l'architecture proposée, et donne un aperçu des différentes natures des objets du formalisme des GU, selon le point de vue de la langue (dictionnaires), ou de l'usage (énoncés).

	gu-langue	**gu-usage**
Type d'unité	`a owl:Thing`	`a owl:Class`
Symbole de relation actancielle	`a owl:Thing`	`a owl:ObjectProperty`
Symbole de relation circonstancielle	-	`a owl:ObjectProperty`
Unité	-	`a owl:Thing`

TABLE 10.1 – Les deux modélisations **gu-langue** et **gu-usage** co-existantes dans la représentation en OWL du formalisme des GU.

Bien que ces deux modèles semblent incompatibles, ils peuvent co-exister au sein d'un même modèle grâce au concept de la *surcharge* [4] introduite dans OWL 2. Il est en effet possible d'utiliser une même URI pour deux objets de natures distinctes. Il est donc possible d'utiliser une même URI pour décrire un type d'unité comme un instance dans le modèle **gu-langue**, et comme une classe dans le modèle **gu-usage**. De même il est possible d'utiliser une même URI pour décrire un SRelA comme une instance dans le modèle **gu-langue**, et comme une propriété objet dans le modèle **gu-usage**. Gardons néanmoins à l'esprit que l'on manipule en réalité deux objets différents, bien qu'ils aient la même URI.

Afin d'assurer la cohérence et l'interopérabilité entre les deux modèles, nous définirons un ensemble de règles dites *de transfert*. Ces règles définissent un morphisme d'ontologies au sens de Flouris *et al.* (2008). Il s'agit d'une opération visant à résoudre les hétérogénéités entre les ontologies afin de permettre leur interopérabilité, dont la particularité est de proposer une fonction entre les vocabulaires *et* les axiomes des ontologies.

La sémantique du formalisme des GU sera capturée par un ensemble de règles SPARQL :
- les règles SPARQL OWL 2 RL/RDF écrites en SPARQL, accessibles à l'URL `<http://ns.inria.org/ug/v1/owl2rl.rul>` ;
- des règles complémentaires génériques pour le modèle **gu-langue**, accessibles à l'URL `<http://ns.inria.org/ug/v1/ug-language.rul>`, et détaillées dans la section 10.2 ;
- des règles complémentaires génériques pour le modèle **gu-usage**, accessibles à l'URL `<http://ns.inria.org/ug/v1/ug-usage.rul>`, et détaillées dans la section 10.2 ;
- des règles de transfert entre les deux modèles, accessibles à l'URL `<http://ns.inria.org/ug/v1/ug-transfer.rul>`, et détaillées dans la section 10.2 ;
- les règles spécifiques d'expansion et de contraction des types d'unité formellement définis. Un exemple de telles règles est donné dans la section 10.2.3.3. Nous proposerons un modèle des définitions formelles dans la section 10.3.1, et proposerons une méthode générique pour générer des règles d'expansion et de contraction dans la section 10.3.3.

4. La surcharge se dit *punning* (jeu de mots) dans la terminologie anglaise de OWL.

La figure 10.1 résume nos choix de modélisation, et donne un aperçu du modèle que nous présentons dans ce chapitre.

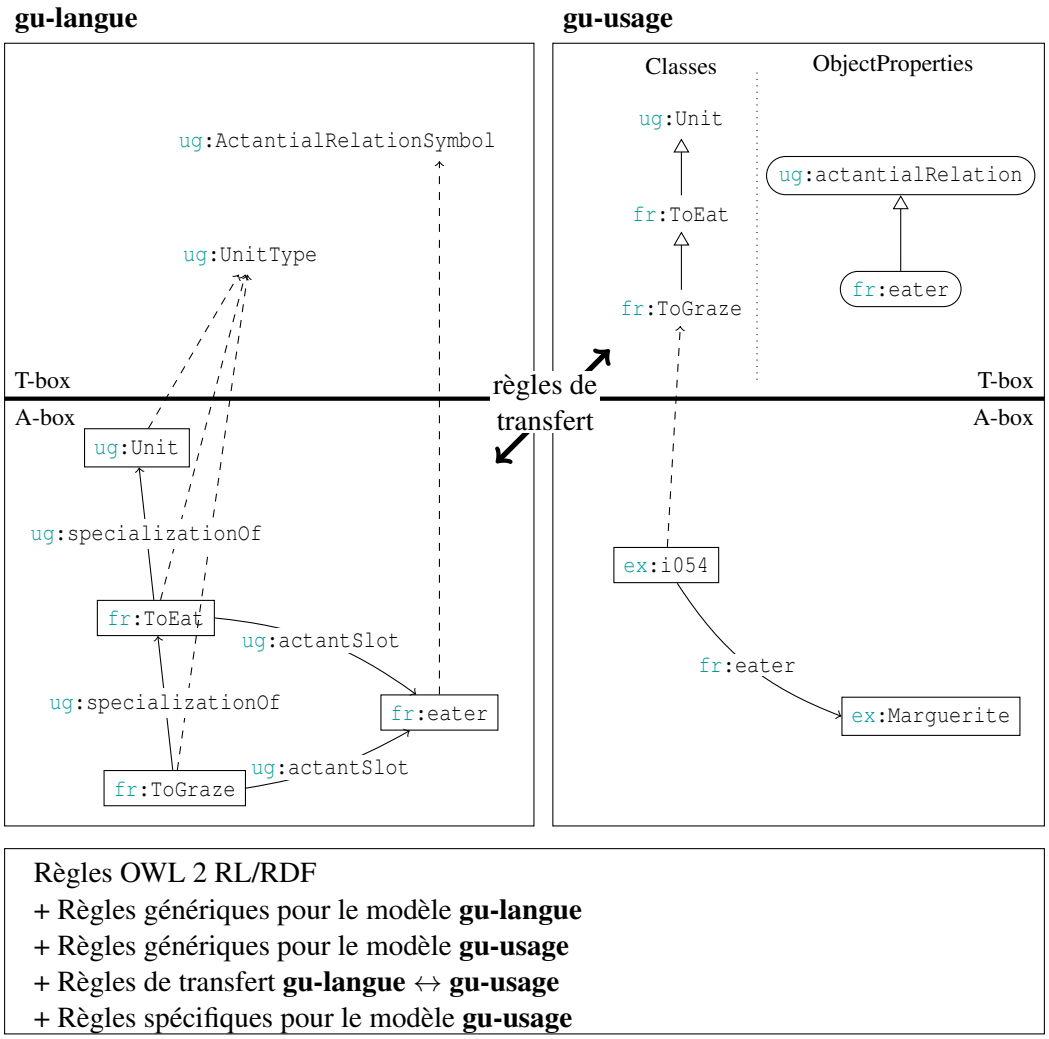


FIGURE 10.1 – Architecture globale du modèle.

10.1.3 Du modèle des graphes d'unités aux représentations linguistiques

Nous développons donc dans un premier temps une ontologie OWL 2 RL pour le modèle des Graphes d'Unités. Cette ontologie est détaillée dans la section 10.2, et accessible à l'URL <http://ns.inria.org/ug/v1#>.

```
1 <http://ns.inria.org/ug/v1#> a owl:Ontology ;
2     dc:title "The Unit Graphs ontology (UG)"@en .
3
4 ug:UnitType a owl:Class ;
5     rdfs:label "UnitType" ;
6     rdfs:comment "The class of unit types."@en ;
7     ...
8
```

```
9 ...
```

Nous introduisons l'espace de nommage <http://ns.inria.org/ug/v1#>, avec le préfixe ug:. Les exemples de ce chapitre utilisent tous implicitement ce préfixe.

```
1 @prefix ug: <http://ns.inria.org/ug/v1#> .
```

Nous souhaitons instancier le modèle des Graphes d'Unités pour la TST en particulier. Nous proposons une modularisation en plusieurs ontologies, suivant deux axes :

- l'indépendance ou non de la langue ;
- l'indépendance ou non des dictionnaires.

La figure 10.2 résume une modularisation pyramidale des ontologies que nous décrivons dans la suite de cette section.

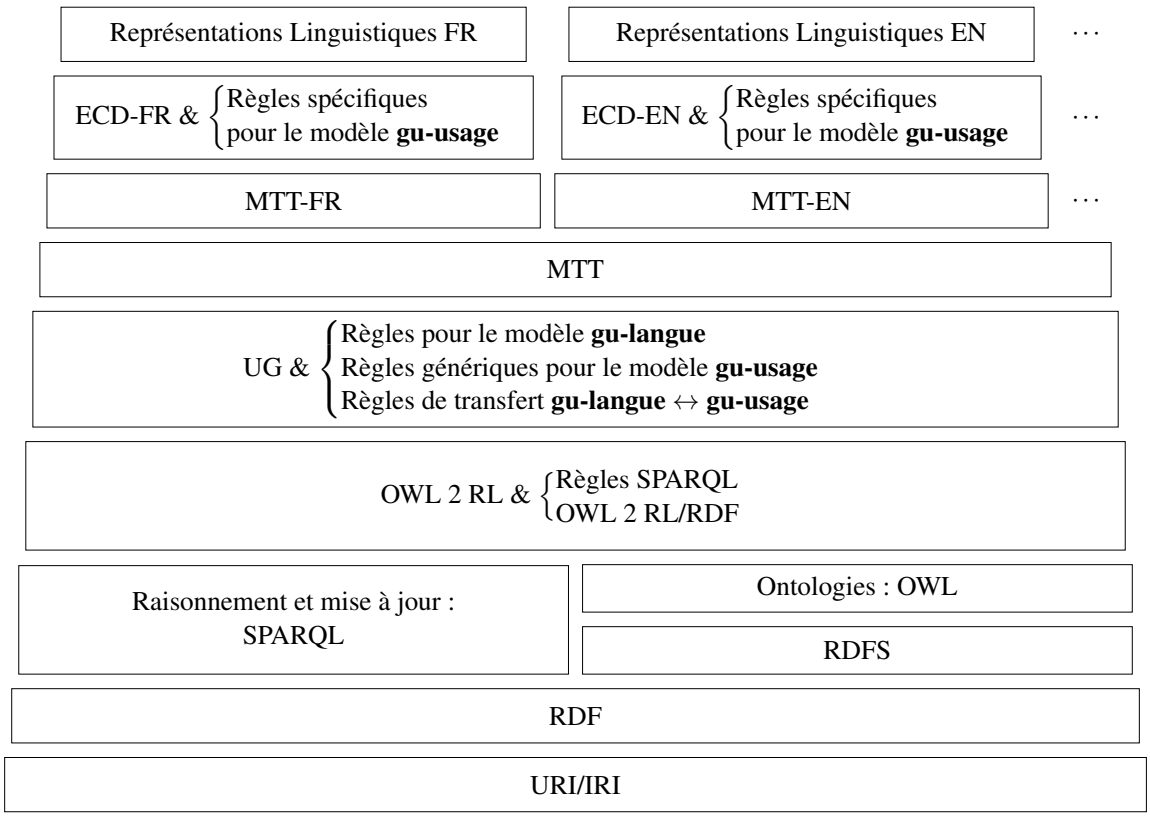


FIGURE 10.2 – Du modèle des graphes d'unités aux représentations linguistiques : la pyramide des langages et des ontologies proposées.

Au-dessus du modèle générique des GU, nous proposons une ontologie pour le modèle Sens-Texte. C'est cette ontologie que nous pourrons aligner avec le modèle **ontolex**. Cette ontologie est détaillée dans la section 10.4.1, et accessible à l'URL <http://ns.inria.org/ug/v1/mtt#>.

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .

<http://ns.inria.org/ug/v1/mtt#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1#> ;
    dc:title "The Meaning-Text Theory ontology (MTT)."@en .

mtt:LexicalUnitType a owl:Class ;
    rdfs:comment "The class of all lexical unit types."@en ;
    owl:equivalentClass ontolex:LexicalEntry ;
    ...

...
```

Ensuite, certaines connaissances dépendent de la langue, mais seraient communes à tout DEC développé pour cette langue. Nous proposons une petite ontologie pour le français et pour l'anglais à des fins d'illustration des exemples de ce mémoire :

- pour le français, cette ontologie est accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/fr#> ;
- pour l'anglais, cette ontologie est accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/en#>.

Ces ontologies sont détaillées dans la section 10.4.2. Par exemple pour le français,

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .

<http://ns.inria.org/ug/v1/mtt/fr#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt#> ;
    dc:title "A partial french-specific Meaning-Text Theory ontology (MTT-Fr)."@en .

mtt-fr:Present a mtt:GrammaticalUnitType , ug:UnitType , owl:Class ;
    owl:equivalentClass isocat:DC-4965 ; # present
    ...

mtt-fr:subjectale a ug:ActantialRelationSymbol , owl:ObjectProperty;
    owl:equivalentProperty isocat:DC-2261 ; # subject
    ...

...
```

Un DEC du français peut alors être (partiellement) modélisé dans une ontologie qui importe l'ontologie <http://ns.inria.org/ug/v1/mtt/fr#>. Nous proposons un petit DEC français et un petit DEC anglais à des fins d'illustration des exemples de ce mémoire :

- un petit DEC français, accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/fr/ecd#> ;
- un petit DEC anglais, accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/en/ecd#>.

Ces ontologies sont détaillées dans la section 10.4.3. Par exemple pour le français,

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .
@prefix ecd-fr: <http://ns.inria.org/ug/v1/mtt/fr/ecd#> .
@prefix isocat: <http://www.isocat.org/datcat/> .

<http://ns.inria.org/ug/v1/mtt/fr/ecd#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/fr#> ;
    dc:title "A small french Explanatory and Combinatorial Dictionary (ECD-Fr)."@en .

ecd-fr:lut-Avoir a mtt:LexicalUnitType , ug:UnitType , owl:Class .
    rdfs:subClassOf isocat:DC-1424 ; # verb
    ...
...
```

Précisons que l'étude des dictionnaires multilingues fait partie des perspectives de cette thèse.

Finalement, une représentation linguistique est modélisée par un GU dans une ontologie qui importe une ontologie de DEC. Nous proposons deux petites ontologies pour illustrer les exemples de GU de ce mémoire. Ces ontologies importent les ontologies des dictionnaires associés :

- pour les exemples en français, l'ontologie est accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/fr/example#> ;
- pour les exemples en anglais, l'ontologie est accessible à l'URL suivante : <http://ns.inria.org/ug/v1/mtt/en/example#>.

Ces ontologies sont détaillées dans la section 10.4.3. Par exemple pour le français,

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .
@prefix ecd-fr: <http://ns.inria.org/ug/v1/mtt/fr/ecd#> .
@prefix ex-fr: <http://ns.inria.org/ug/v1/mtt/fr/example#> .

<http://ns.inria.org/ug/v1/mtt/fr/example#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/fr/ecd#> ;
    dc:title "Examples of Unit Graphs."@en .

_:avoir-i12 a ecd-fr:lut-Avoir , mtt-fr:Present , mtt:LexicalUnit , ug:Unit ;
    mtt-fr:subjectale ex-fr:Orwell ;
    ...
...
```

Chacune des ontologies décrites dans ce chapitre est retranscrite dans l'annexe D.1.1, et accessible en ligne aux URL sus-mentionnées.

En résumé, dans ce chapitre, les exemples de cette section font implicitement usage des préfixes usuels du Web sémantique, et éventuellement usage des préfixes suivants :

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .
@prefix ecd-fr: <http://ns.inria.org/ug/v1/mtt/fr/ecd#> .
@prefix ex-fr: <http://ns.inria.org/ug/v1/mtt/fr/example#> .
@prefix mtt-en: <http://ns.inria.org/ug/v1/mtt/en#> .
@prefix ecd-en: <http://ns.inria.org/ug/v1/mtt/en/ecd#> .
@prefix ex-en: <http://ns.inria.org/ug/v1/mtt/en/example#> .
```

10.2 Représentation du formalisme des Graphes d'Unités

Cette section détaille le modèle générique du formalisme des GU. Nous décrivons successivement :

- La modélisation d'une hiérarchie de types d'unité sans structures actancielles, et la représentation des unités dans les GU (§10.2.1) ;
- la modélisation des SRelA et de la structure actancielle des types d'unité (§10.2.2) ;
- la modélisation des SRelC, des GU, et la simulation de la déduction au sein du formalisme des GU (§10.2.3).

Finalement, nous avons choisi de développer deux modèles hétérogènes interopérables par un ensemble de règles de transfert, nous avons choisi une organisation similaire pour chacune des sous-sections ci-dessous :

1. définition de la modélisation **gu-langue** ;
2. définition de la modélisation **gu-usage** ;
3. introduction des règles de transfert entre les deux modèles.

10.2.1 Unités et types d'unité, sans structure actancielle

Soit $\mathscr{T} = \langle T_D, \boldsymbol{S}_{\mathscr{T}}, \boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Gamma_1}, \boldsymbol{\gamma_1}, \boldsymbol{\Gamma_0}, \boldsymbol{\gamma_0}, C_A, \bot_A^{\sqcap}, \{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}} \rangle$ une hiérarchie de types d'unité. Nous supposons dans cette section qu'il n'existe aucun symbole de relation actancielle, ni définition de type d'unité. Cette section introduit successivement la modélisation des types d'unité et des unités (§10.2.1.1), la hiérarchisation des types d'unité (§10.2.1.2), et la conjonction de types d'unité (§10.2.1.3). Nous concluons sur la modélisation des types d'unité conjonctifs déclarés absurdes.

10.2.1.1 Types d'Unité et Unités

Modèle gu-langue Nous introduisons une classe OWL `ug:UnitType` pour les types d'unité (primitifs et conjonctifs).

```
ug:UnitType a owl:Class ;
    rdfs:label "UnitType" ;
    rdfs:comment "The class of unit types."@en .
```

Un TUP déclaré $t \in T_D$, suivant la définition 26, est associé à une URI uri(t). Il sera donc représenté par une ressource nommée uri(t). Par exemple, le type d'unité $/$to eat$\backslash$ associé à l'URI `<http//ns.inria.org/ug/v1/mtt/en/ecd#dsem-ToEat>` sera représenté de la manière suivante :

```
ecd-en:dsem-ToEat a ug:UnitType .
```

Nous introduisons deux instances particulières de `ug:UnitType` :

- `ug:Unit` représente le TUP universel premier $\top$;
- `ug:AbsurdUnit` représente le TUP absurde premier $\bot$.

```
ug:Unit a ug:UnitType ;
    rdfs:label "Unit" ;
    rdfs:comment "The prime universal unit type."@en .

ug:AbsurdUnit a ug:UnitType ;
    rdfs:label "AbsurdUnit" ;
    rdfs:comment "The prime absurd unit type."@en .
```

Modèle gu-usage Dans ce modèle, un type d'unité est représenté par une classe OWL. Nous déclarons donc `ug:Unit` et `ug:AbsurdUnit` comme étant des classes OWL. `ug:Unit` est la classe de toute unité. `ug:AbsurdUnit` n'est la classe d'aucune unité, donc est une sous-classe de `owl:Nothing`.

```
ug:Unit a owl:Class ;
    rdfs:label "Unit" ;
    rdfs:comment "The class of all units."@en .

ug:AbsurdUnit a owl:Class ;
    rdfs:subClassOf owl:Nothing ;
    rdfs:label "AbsurdUnit" ;
    rdfs:comment "The class of no units."@en .
```

Les types d'unité sont donc des sous-classes de `ug:Unit`. Par exemple, le type d'unité $/$to eat$\backslash$ associé à l'URI `ecd-en:dsem-ToEat` sera représenté de la manière suivante :

```
ecd-en:dsem-ToEat a owl:Class ;
    rdfs:subClassOf ug:Unit .
```

Ce modèle permet la représentation des unités : des instances du type d'unité `ug:Unit`. Par exemple, le code suivant représente une unité via un identifiant d'URI `<http//ns.inria.org/ug/v1/mtt/en/example#dsem-ToEat-i015>`, et spécifie qu'elle est conjointement des types $\top$ et $/$to eat$\backslash$.

```
ex-en:dsem-ToEat-i015 a ug:Unit, ecd-en:dsem-ToEat .
```

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage**.

```
# language-usage transfer rule: unit types
CONSTRUCT {
  ?t a owl:Class ;
    rdfs:subClassOf ug:Unit .
}
WHERE {
  ?t a ug:UnitType .
}
```

```
# usage-language transfer rule: unit types
CONSTRUCT {
  ?t a ug:UnitType .
}
WHERE {
  ?t a owl:Class ;
    rdfs:subClassOf ug:Unit .
}
```

10.2.1.2 Hiérarchisation des types d'unité

Modèle gu-langue Nous introduisons une relation réflexive et transitive `ug:specializationOf` sur l'ensemble des types d'unité afin de représenter le préordre $\overset{\cap}{\lesssim}$.

```
ug:specializationOf a owl:ObjectProperty ;
    a owl:ReflexiveProperty ;
    a owl:TransitiveProperty ;
    rdfs:label "specializationOf" ;
    rdfs:comment "The subject, a unit type, is a specialization of the object, a unit
        type."@en ;
    rdfs:domain ug:UnitType ;
    rdfs:range ug:UnitType .
```

Par exemple, le triplet suivant représente le fait que le type d'unité $/$to graze$\backslash$ est plus spécifique que le type d'unité $/$to eat$\backslash$.

```
ecd-en:dsem-ToGraze ug:specializationOf ecd-en:dsem-ToEat .
```

Tout type d'unité est plus spécifique que `ug:Unit`, et plus générique que `ug:AbsurdUnit`. Nous pouvons représenter cette contrainte à l'aide de l'axiome **ObjectHasValue**.

```
ug:UnitType rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ug:specializationOf ;
       owl:hasValue ug:Unit ] ;
    rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty [ owl:inverseOf ug:specializationOf ] ;
       owl:hasValue ug:AbsurdUnit ] .
```

Enfin, deux types d'unité sont liés par la relation `owl:sameAs` si et seulement s'ils sont la spécialisation l'un de l'autre. Les règles suivantes étendent la sémantique de OWL 2 RL pour capturer cette connaissance.

```
# two unit types that are the same specialize each other.
CONSTRUCT {
  ?t1 owl:specializationOf ?t2 .
  ?t2 owl:specializationOf ?t1 .
}
WHERE {
  ?t1 owl:sameAs ?t2 .
  ?t1 a ug:UnitType .
  ?t2 a ug:UnitType .
}
```

```
# two unit types that specialize each other are the same.
CONSTRUCT {
  ?t1 owl:sameAs ?t2 .
}
WHERE {
  ?t1 owl:specializationOf ?t2 .
  ?t2 owl:specializationOf ?t1 .
}
```

Modèle gu-usage Dans ce modèle, la hiérarchisation des types d'unité correspond à une hiérarchisation des classes. Nous utilisons donc l'axiome **SubClassOf**. Par exemple, le triplet suivant représente le fait que le type d'unité /to graze\ est plus spécifique que le type d'unité /to eat\.

```
ecd-en:dsem-ToGraze rdfs:subClassOf ecd-en:dsem-ToEat .
```

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage**.

```
# language-usage transfer rule: specialization relation
CONSTRUCT {
  ?t1 rdfs:subClassOf ?t2 .
}
WHERE {
  ?t1 ug:specializationOf ?t2 .
}
```

```
# usage-language transfer rule: specialization relation
CONSTRUCT {
  ?t1 ug:specializationOf ?t2 .
}
WHERE {
  ?t2 rdfs:subClassOf ug:Unit .
  ?t1 rdfs:subClassOf ?t2 .
}
```

10.2.1.3 Types d'Unité conjonctifs

Modèle gu-langue Nous introduisons une relation `ug:conjunctionOf` sur l'ensemble des types d'unité.

```
ug:conjunctionOf a owl:ObjectProperty ;
   rdfs:label "conjunctionOf" ;
   rdfs:comment "The subject, a unit type, is the conjunction of the object, a list of unit
       types."@en ;
   rdfs:domain ug:UnitType ;
   rdfs:range ug:UnitType .
```

Par exemple, le code suivant représente un type d'unité anonyme qui est la conjonction de ⌈USA⌉ et *singular*.

```
[] ug:conjunctionOf ( ecd-en:USA mtt-en:singular ) .
```

Ainsi, nous pouvons déclarer que ce TUC est absurde de la manière suivante :

```
[] ug:conjunctionOf ( ecd-en:USA mtt-en:singular ) ;
   ug:specializationOf ug:AbsurdUnit .
```

La sémantique de OWL 2 RL ne permet pas d'inférer que ce type d'unité est le plus générique des types d'unité plus spécifique à la fois de ⌈USA⌉ et de *singular*. Nous enrichissons donc l'ensemble des règles de OWL 2 RL/RDF avec les deux règles SPARQL suivantes pour capturer cette connaissance.

```
# a conjunctive unit type is more specific than any unit type it is composed of
CONSTRUCT { ?tc ug:specializationOf ?t . }
WHERE { ?tc (ug:conjunctionOf/(rdf:rest)*)/rdf:first ?t . }
```

```
# if a unit type is more specific than all the unit types a conjunctive unit type is composed
    of, then it is more specific than the conjunctive unit type
CONSTRUCT {
  ?t ug:specializationOf ?tc .
}
WHERE {
  ?tc ug:conjunctionOf ?list .
  ?list rdf:first ?t1 .
  ?t ug:specializationOf ?t1 .
  FILTER NOT EXISTS {
    ?list rdf:rest+/rdf:first ?tn .
    FILTER NOT EXISTS {
     ?t ug:specializationOf ?tn .
    }
  }
}
```

Modèle gu-usage Nous faisons usage du constructeur de classe **ObjectIntersection**. Par exemple, le code suivant représente la conjonction de ⌈USA⌉ et *singular*.

```
[] owl:intersectionOf ( ecd-en:USA mtt-en:singular ) .
```

Nous pouvons représenter le fait que ce TUC est absurde de la manière suivante :

```
[] owl:intersectionOf ( ecd-en:USA mtt-en:singular ) ;
   rdfs:subClassOf ug:AbsurdUnit .
```

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage**.

```
# language-usage transfer rule: conjunctive unit types
CONSTRUCT {
  ?tc owl:intersectionOf ?list .
}
WHERE {
  ?tc ug:conjunctionOf ?list .
}
```

```
# usage-language transfer rule: conjunctive unit types
CONSTRUCT {
  ?tc ug:conjunctionOf ?list .
}
WHERE {
  ?tc rdfs:subClassOf ug:Unit .
  ?tc owl:intersectionOf ?list .
}
```

10.2.2 Prise en compte de la structure actancielle des types d'unité

Cette section précise la modélisation des SRelA et des structures actancielles de types d'unité. Nous introduisons successivement la modélisation des SRelA (§10.2.2.1), des racines de PosA, PosAObl, et PosAInt (§10.2.2.2), des positions actancielles des types d'unité (§10.2.2.3), et finalement des signatures de ces positions actancielles (§10.2.2.4).

10.2.2.1 Symboles de relation actancielle

Modèle gu-langue Nous introduisons une classe OWL `ug:ActantialRelationSymbol` pour les SRelA.

```
ug:ActantialRelationSymbol a owl:Class ;
    rdfs:label "ActantialRelationSymbol" ;
    rdfs:comment "The class of actantial relation symbols."@en .
```

Un SRelA $s \in \boldsymbol{S}_{\mathscr{T}}$, suivant la définition 27, est associé à une URI uri(s). Il sera donc représenté par une ressource nommée uri(t). Par exemple, le SRelA *eater* associé à l'URI `<http://ns.inria.org/ug/v1/mtt/en/ecd#eater>` sera représenté de la manière suivante.

```
ecd-en:eater a ug:ActantialRelationSymbol .
```

Modèle gu-usage Dans ce modèle, un SRelA est représenté par une propriété OWL fonctionnelle. Nous introduisons une propriété fonctionnelle `ug:actantialRelation` qui sera la super-propriété de chaque SRelA. Toute relation actancielle lie deux unités, donc le domaine et le co-domaine de `ug:actantialRelation` sont `ug:Unit`.

```
ug:actantialRelation a owl:ObjectProperty ;
    a owl:FunctionalProperty ;
    rdfs:label "actantialRelation" ;
    rdfs:comment "The generic actantial relation."@en ;
    rdfs:domain ug:Unit ;
    rdfs:range ug:Unit .
```

Par exemple, le SRelA associé à l'URI `<http://ns.inria.org/ug/v1/mtt/en/ecd#eater>` sera représenté de la manière suivante :

```
ecd-en:eater a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation .
```

Ce modèle permet la représentation des triplets actanciels dans les GU. Par exemple, le code suivant représente qu'une unité identifiée par l'URI <http://ns.inria.org/ug/v1/mtt/en/example#ToEat-i015> gouverne une unité identifiée par l'URI <http://ns.inria.org/ug/v1/mtt/en/example#Animal-i354> via une relation actancielle <http://ns.inria.org/ug/v1/mtt/en/ecd#eater>.

```
ex-en:ToEat-i015 ecd-en:eater ex-en:Animal-i354 .
```

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage**.

```
# language-usage transfer rule: actantial relation symbols
CONSTRUCT {
  ?s a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation .
}
WHERE {
  ?s a ug:ActantialRelationSymbol .
}
```

```
# usage-language transfer rule: actantial relation symbols
CONSTRUCT {
  ?s a ug:ActantialRelationSymbol .
}
WHERE {
  ?s a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation .
}
```

Enfin, si deux SRelA sont liés par la relation `owl:sameAs` dans le modèle **gu-usage**, nous pouvons en déduire qu'ils sont des propriétés OWL équivalentes dans le modèle **gu-usage**. La règle suivante étend la sémantique de OWL 2 RL pour représenter cette connaissance.

```
# two actantial relation symbols that are the same, are equivalent object properties.
CONSTRUCT {
  ?s1 owl:EquivalentProperty ?s2 .
}
WHERE {
  ?s1 owl:sameAs ?s2 .
  ?s1 a ug:ActantialRelationSymbol .
  ?s2 a ug:ActantialRelationSymbol .
}
```

10.2.2.2 Racines de PosA, de PosAObl, et de PosAInt

Modèle gu-langue Chaque SRelA possède une unique racine de PosA, une unique racine de PosAObl, et une unique racine de PosAInt. Nous introduisons donc trois paires de propriétés OWL :

- `ug:actantSlotRoot` et `ug:actantSlotRootOf` ;
- `ug:obligatoryActantSlotRoot` et `ug:obligatoryActantSlotRootOf` ;
- `ug:prohibitedActantSlotRoot` et `ug:prohibitedActantSlotRootOf` ;

`ug:actantSlotRoot` lie un SRelA au type d'unité qui représente la racine de PosA. Cette propriété est donc fonctionnelle. `ug:actantSlotRootOf` est la propriété inverse de `ug:actantSlotRoot`.

```
ug:actantSlotRoot a owl:ObjectProperty ;
    a owl:FunctionalProperty ;
    rdfs:label "actantSlotRoot" ;
    rdfs:comment "The subject, an actantial symbol relation, has for actant slot root the
        object, a unit type."@en ;
    rdfs:domain ug:ActantialRelationSymbol ;
    rdfs:range ug:UnitType ;
```

```
    owl:inverseOf ug:actantSlotRootOf .

ug:actantSlotRootOf a rdf:Property ;
    a owl:InverseFunctionalProperty ;
    rdfs:label "actantSlotRootOf" ;
    rdfs:comment "The subject, a unit type, is the actant slot root of the object, an
        actantial symbol relation."@en ;
    rdfs:domain ug:UnitType ;
    rdfs:range ug:ActantialRelationSymbol ;
    owl:inverseOf ug:actantSlotRoot .
```

Par exemple, le code suivant représente le lien entre *eater* et un type d'unité anonyme qui représente sa racine de PosA.

```
ecd-en:eater ug:actantSlotRoot _:rapos .
```

Nous devons imposer que la racine de PosA d'un SRelA soit plus générique que ses racines de PosAObl et de PosAInt. Cette connaissance peut être représentée en OWL 2 RL à l'aide des axiomes suivants :

```
ug:specializationOf
    owl:propertyChainAxiom ( ug:obligatoryActantSlotRootOf ug:actantSlotRoot ) ;
    owl:propertyChainAxiom ( ug:prohibitedActantSlotRootOf ug:actantSlotRoot ) .
```

Enfin, nous devons imposer que le TUC formé des racines de PosAObl et de PosAInt d'un même SRelA soit absurde. Nous enrichissons donc l'ensemble des règles de OWL 2 RL/RDF avec la règle SPARQL suivante, qui capture la généralisation de cette connaissance : la proposition 7.1.2.

```
# the conjunction of the obligatory actant slot root and the prohibited actant slot root of a
    given actantial relation symbol is an absurd unit type
CONSTRUCT {
  ?t ug:specializationOf ug:AbsurdUnit .
}
WHERE {
  ?t ug:specializationOf/ug:obligatoryActantSlotRootOf ?s .
  ?t ug:specializationOf/ug:prohibitedActantSlotRootOf ?s .
}
```

Modèle gu-usage Nous ne pouvons que partiellement représenter la racine de PosA d'un SRelA avec l'axiome **ObjectPropertyDomain**. Par exemple le triplet ci-dessous signifie que la racine de PosA *eater* est plus spécifique que (ou équivalent à) /to eat\.

```
ecd-en:eater rdfs:domain ecd-en:dsem-ToEat .
```

Pour ce qui est des racines de PosAObl et de PosAInt, nous avons déjà entrevu dans la section 6.1.1.1 comment les représenter en logiques de description. En OWL, on aimerait utiliser les constructeurs de classes auxiliaires **ObjectMinCardinality** et **ObjectMaxCardinality**. Par exemple, les classes anonymes ci-dessous représenteraient respectivement la racine de PosAObl et la racine de PosAInt de la SRelA <http://ns.inria.org/ug/v1/mtt/en/ecd#eater>.

```
[] a owl:Restriction ;
    owl:onProperty ecd-en:eater ;
    owl:minCardinality "1"^^xsd:nonNegativeInteger .

[] a owl:Restriction ;
    owl:onProperty ecd-en:eater ;
    owl:maxCardinality "0"^^xsd:nonNegativeInteger .
```

Cependant, la modélisation des racines de PosAObl n'est pas conforme au profil OWL 2 RL. Nous ne modélisons donc que les racines de PosAInt dans le modèle **gu-usage**. Nous utilisons donc le modèle **gu-langue** pour inférer que le TUC formé des racines de PosAObl et de PosAInt d'un même SRelA est absurde.

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage** pour les racines de PosA.

```
# language-usage transfer rule: ASlotRoots
CONSTRUCT {
  ?s rdfs:domain ?t .
}
WHERE {
  ?s ug:actantSlotRoot ?t .
}
```

```
# usage-language transfer rule: ASlotRoots
CONSTRUCT {
  ?rapos ug:specializationOf ?t .
}
WHERE {
  ?s rdfs:domain ?t .
  ?s ug:actantSlotRoot ?rapos .
}
```

Pour les racines de PosAInt, les règles de transfert sont de la forme suivante :

```
# language-usage transfer rule: ProASlotRoots
CONSTRUCT {
  ?t owl:equivalentClass [
    a owl:Restriction ;
    owl:onProperty ?s ;
    owl:maxCardinality
        "0"^^xsd:nonNegativeInteger ] .
}
WHERE {
  ?s ug:prohibitedActantSlotRoot ?t .
}
```

```
# usage-language transfer rule: ProASlotRoots
CONSTRUCT {
  ?s ug:prohibitedActantSlotRoot ?t .
}
WHERE {
  ?s rdfs:subPropertyOf ug:actantialRelation .
  ?t a owl:Restriction ;
    owl:onProperty ?s ;
    owl:maxCardinality
        "0"^^xsd:nonNegativeInteger .
}
```

10.2.2.3 Positions actancielles

Modèle gu-langue Nous utilisons trois propriétés OWL pour lier un type d'unité à ses PosA : `ug:actantSlot`, `ug:obligatoryActantSlotRootoryActantSlot`, et `ug:prohibitedActantSlot`. Nous ne représenterons pas les PosA optionnelles. `ug:obligatoryActantSlotRootoryActantSlot` et `ug:prohibitedActantSlot` sont toutes deux des sous-propriétés de `ug:actantSlot`, car nous savons qu'une PosA obligatoire ou optionnelle est une PosA. Prenons ici simplement les exemples de `ug:actantSlot` et `ug:obligatoryActantSlotRootoryActantSlot`.

```
ug:actantSlot a owl:ObjectProperty ;
    rdfs:label "actantSlot" ;
    rdfs:comment "The subject, a unit type, has for actant slot the object, an actantial
        symbol relation." ;
    rdfs:domain ug:UnitType ;
    rdfs:range ug:ActantialRelationSymbol .

ug:obligatoryActantSlotRootoryActantSlot a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantSlot ;
    rdfs:label "obligatoryActantSlot" ;
    rdfs:comment "The subject, a unit type, has for obligatory actant slot rootory actant slot
        the object, an actantial symbol relation." ;
```

```
    rdfs:domain ug:UnitType ;
    rdfs:range ug:ActantialRelationSymbol .
```

Nous savons qu'un type d'unité (primitif ou conjonctif) possède une PosA (resp. PosAObl, PosAInt) *s* s'il est plus spécifique que la racine de PosA (resp. PosAObl, PosAInt) *s*. Ces connaissances peuvent être modélisées à l'aide des axiomes suivants :

```
ug:actantSlot owl:propertyChainAxiom ( ug:specializationOf ug:actantSlotRootOf ) .
ug:obligatoryActantSlotRootoryActantSlot
    owl:propertyChainAxiom ( ug:specializationOf ug:obligatoryActantSlotRootOf ) .
ug:prohibitedActantSlot
    owl:propertyChainAxiom ( ug:specializationOf ug:prohibitedActantSlotRootOf ) .
```

A l'inverse, un type d'unité qui possède une PosA (resp. PosAObl, PosAInt) *s* est plus spécifique que la racine de PosA (resp. PosAObl, PosAInt) *s*. Ces connaissances pourraient être modélisées à l'aide des axiomes suivants :

```
ug:specializationOf owl:propertyChainAxiom ( ug:actantSlot ug:actantSlotRoot ) ;
    owl:propertyChainAxiom ( ug:obligatoryActantSlotRootoryActantSlot
        ug:obligatoryActantSlotRoot ) ;
    owl:propertyChainAxiom ( ug:prohibitedActantSlot ug:prohibitedActantSlotRoot ) .
```

Cependant chacun de ces derniers axiomes rend la hiérarchie des propriétés non régulière pour les LD. Par exemple, `specializationOf` est *défini* par `ug:actantSlot`, lui même *défini* par `specializationOf`. Nous préférons donc conserver la conformité syntaxique avec OWL 2 RL, mais enrichir l'ensemble des règles de OWL 2 RL/RDF avec les trois règles SPARQL suivantes, qui permettent les mêmes inférences.

```
CONSTRUCT {
  ?t ug:specializationOf ?rapos .
}
WHERE {
  ?t ug:actantSlot/ug:actantSlotRootOf ?rapos .
}
```

```
CONSTRUCT {
  ?t ug:specializationOf ?oblaslotroot .
}
WHERE {
  ?t ug:obligatoryActantSlotRootoryActantSlot/ug:obligatoryActantSlotRootOf ?oblaslotroot .
}
```

```
CONSTRUCT {
  ?t ug:specializationOf ?proaslotroot .
}
WHERE {
  ?t ug:prohibitedActantSlot/ug:prohibitedActantSlotRootOf ?proaslotroot .
}
```

Modèle gu-usage Dans ce modèle, et de par les contraintes de conformité imposées par le profil OWL 2 RL, nous ne pouvons représenter que les positions actancielles interdites. Par exemple le code suivant représente le fait que /to graze\ possède une PosAInt *container*.

```
ecd-en:dsem-ToGraze rdfs:subClassOf [ a owl:Restriction ;
    owl:onProperty ecd-en:container ;
    owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
```

Par contre, la sémantique de OWL 2 RL suffit à inférer l'héritage des PosAObl.

Règles de transfert Les règles de transfert entre les modèles **gu-langue** et **gu-usage** pour les structures actancielles sont de la forme suivante.

```
# language-usage transfer rule: prohibited actant slot
CONSTRUCT {
  ?t rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty ?s ;
    owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
}
WHERE {
  ?t ug:prohibitedActantSlot ?s .
}
```

```
# usage-language transfer rule: prohibited actant slot
CONSTRUCT {
  ?t ug:prohibitedActantSlot ?s .
}
WHERE {
  ?s rdfs:subPropertyOf ug:actantialRelation .
  ?t rdfs:subClassOf ug:Unit ;
    rdfs:subClassOf [ a owl:Restriction ;
      owl:onProperty ?s ;
      owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
}
```

10.2.2.4 Représentation des signatures

Modèle gu-langue Nous nous inspirons de l'expression de classe **ObjectAllValuesFrom**. La signature d'un type d'unité *t* pour une PosA *s* est représentée par un type d'unité auxiliaire, plus générique que *t*. Nous définissons donc une classe OWL `ug:Signature`, sous-classe des types d'unité, pour représenter les signatures.

```
ug:Signature a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
    rdfs:label "Signature" ;
    rdfs:comment "The class of signatures."@en .
```

Nous définissons deux propriétés OWL pour renseigner une signature :

- la propriété `ug:onActantSlot` définit la PosA concernée.
- la propriété `ug:allUnitsFrom` définit le type d'unité concerné.

```
ug:onActantSlot a owl:ObjectProperty ;
    rdfs:label "onActantSlot" ;
    rdfs:comment "The subject, a signature, affect the object, an actant slot." ;
    rdfs:domain ug:Signature ;
    rdfs:range ug:ActantialRelationSymbol .

ug:allUnitsFrom a owl:ObjectProperty ;
    rdfs:label "allUnitsFrom" ;
    rdfs:comment "The subject, a signature, is restricted to the object, a unit type." ;
    rdfs:domain ug:Signature ;
    rdfs:range ug:UnitType .
```

Ainsi le code suivant définit la signature de /to graze\ à {/vegetal\}.

```
ecd-en:dsem-ToGraze ug:specializationOf [ a ug:Signature ;
    ug:onActantSlot ex:eaten ;
    ug:allUnitsFrom ex:Vegetal ] .
```

Toute signature qui concerne une PosA *s* devrait être plus spécifique que la racine de PosA *s*. Nous enrichissons donc l'ensemble des règles de OWL 2 RL/RDF avec la règle SPARQL suivante, qui capture cette connaissance.

```
CONSTRUCT {
  ?sig ug:specializationOf ?rapos .
}
WHERE {
  ?sig ug:onActantSlot/ug:actantSlotRoot ?rapos .
}
```

Ensuite, si deux signatures concernant une même PosA *s* portent sur des types d'unité qui sont plus spécifiques l'un que l'autre, alors ces signatures sont plus spécifiques l'une que l'autre. Nous enrichissons donc l'ensemble des règles de OWL 2 RL/RDF avec la règle SPARQL suivante, qui capture cette connaissance.

```
CONSTRUCT {
  ?sig1 ug:specializationOf ?sig2 .
}
WHERE {
  ?sig1 ug:onActantSlot ?s .
  ?sig1 ug:allUnitsFrom ?t1 .
  ?sig2 ug:onActantSlot ?s .
  ?sig2 ug:allUnitsFrom ?t2 .
  ?t1 ug:specializationOf ?t2 .
}
```

La restriction de la définition 38 est automatiquement validée grâce à la transitivité de la relation `ug:specializationOf`. En effet, supposons qu'un type d'unité t_1 soit plus spécifique qu'un type d'unité t_2, qui possède une signature $t^{\cap}$ pour une PosA *s*. La transitivité de la relation `ug:specializationOf` impose alors que t_1 possède également la signature $t^{\cap}$ pour la PosA *s*.

Enfin, si la signature d'un type d'unité pour une PosAObl est absurde, alors ce type d'unité est absurde. Nous enrichissons donc l'ensemble des règles de OWL 2 RL/RDF avec la règle SPARQL suivante, qui capture cette connaissance.

```
CONSTRUCT {
  ?t ug:specializationOf ug:AbsurdUnit .
}
WHERE {
  ?t ug:obligatoryActantSlot ?s ; ug:specializationOf ?sig .
  ?sig ug:onActantSlot ?s ; ug:allUnitsFrom/ug:specializationOf ug:AbsurdUnit .
}
```

Modèle gu-usage Dans ce modèle, nous utilisons directement l'expression de classe **ObjectAllValuesFrom**. Par exemple, le code suivant définit la signature de /to graze\ à {/vegetal\}.

```
ecd-en:dsem-ToGraze rdfs:subClassOf [ a owl:Restriction ;
    owl:onProperty ex:eaten ;
    owl:allValuesFrom ex:Vegetal ] .
```

La sémantique de OWL 2 RL permet alors directement les mêmes inférences que pour le modèle **gu-langue**.

Règles de transfert Les règles SPARQL suivantes assurent le transfert entre les modèles **gu-langue** et **gu-usage** pour les signatures.

```
# language-usage transfer rule: signatures
CONSTRUCT {
  ?sig a owl:Restriction ;
    owl:onProperty ?s ;
    owl:allValuesFrom ?t .
}
WHERE {
  ?sig a ug:Signature ;
    ug:onActantSlot ?s ;
    ug:allUnitsFrom ?t .
}
```

```
# usage-language transfer rule: signatures
CONSTRUCT {
  ?sig a ug:Signature ;
    ug:onActantSlot ?s ;
    ug:allUnitsFrom ?t .
}
WHERE {
  ?s rdfs:subClassOf ug:actantialRelation .
  ?t rdfs:subClassOf ug:Unit .
  ?sig rdfs:subClassOf ug:Unit ;
    a owl:Restriction ;
    owl:onProperty ?s ;
    owl:allValuesFrom ?t .
}
```

Détermination de l'acyclicité du graphe $G_{\mathscr{T}}$. Maintenant que nous avons entièrement modélisé les structures actancielles en OWL, nous souhaitons déterminer si le graphe $G_{\mathscr{T}}$ associé à un dictionnaire est cyclique ou non. La requête SPARQL suivante renvoie VRAI si le graphe $G_{\mathscr{T}}$ est cyclique.

```
ASK {
  ?t (ug:specializationOf/ug:allUnitsFrom)+ [ ug:specializationOf ?t ] .
  FILTER NOT EXISTS {
    ?t ug:specializationOf ug:AbsurdUnit .
  }
}
```

10.2.3 Graphes d'unités et raisonnement

Maintenant que nous avons précisé la modélisation d'une hiérarchie des types d'unité sans définition de TUP, cette section introduit la modélisation des GU et la capture de leur sémantique par des règles SPARQL. Nous introduisons d'abord le modèle des SRelC (§10.2.3.1). Nous précisons ensuite la classe des GU réguliers qui peuvent faire l'objet d'un aller-retour en RDF (§10.2.3.2). Enfin, nous passons en revue la base des règles axiomatiques d'inférence introduite dans le chapitre 9, et voyons comment la base des règles OWL 2 RL/RDF doit être augmentée pour simuler la déduction dans le modèle **gu-usage** (§10.2.3.3).

10.2.3.1 Hiérarchie des symboles de relation circonstancielle

Soit $\mathscr{C} \stackrel{\text{def}}{=} \langle \boldsymbol{S}_{\mathscr{C}}, \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}, \mathscr{T}, \{\boldsymbol{\sigma}_s\}_{s \in \boldsymbol{S}_{\mathscr{C}}} \rangle$ une hiérarchie de symboles de relation circonstancielle. Un SRelC est représenté par une propriété OWL. Nous introduisons une propriété OWL `ug:circumstantialRelation` qui sera la super-propriété de chaque SRelC. Toute relation circonstancielle lie deux unités, donc le domaine et le co-domaine de `ug:circumstantialRelation` sont `ug:Unit`.

```
ug:circumstantialRelation a owl:ObjectProperty ;
    rdfs:label "circumstantialRelation" ;
    rdfs:comment "The generic circumstantial relation."@en ;
    rdfs:domain ug:Unit ;
    rdfs:range ug:Unit .
```

Un SRelC $s \in \boldsymbol{S}_{\mathscr{C}}$, suivant la définition 56, est associé à une URI uri(s). Il sera donc représenté par une ressource nommée uri(s). Par exemple, le SRelC dsense associée à l'URI `<http://ns.inria.org/ug/v1/mtt#dSense>`, sera représenté de la manière suivante :

```
mtt:dSense a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:circumstantialRelation .
```

Nous représentons l'ensemble de comparaisons déclarées de SRelC à l'aide de l'axiome **subPropertyOf**, et les signatures à l'aide des axiomes **ObjectPropertyDomain** et **ObjectPropertyRange**. Par exemple, nous verrons que la signature de *dsense* est représentée de la manière suivante :

```
mtt:dSense a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:circumstantialRelation ;
    rdfs:domain mtt:SurfaceSemanticUnit ;
    rdfs:range mtt:DeepSemanticUnit .
```

Ce modèle permet la représentation des triplets circonstanciels dans les GU. Par exemple, le code suivant représente qu'une unité identifiée par l'URI `ex-en:ssem-ToEat-i98` gouverne une unité identifiée par l'URI `ex-en:dsem-ToEat-i414>` via une relation circonstancielle `mtt:dSense`.

```
ex-en:ssem-ToEat-i98 mtt:dSense ex-en:dsem-ToEat-i414 .
```

10.2.3.2 Représentation des graphes d'unités

Les graphes d'unités ne peuvent être modélisés que dans le modèle **gu-usage**. Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle$ un graphe d'unité. Dans l'extension de la conceptualisation de la TST que nous avons proposée, ce sont les identifiants d'unité qui possèdent une URI, et plusieurs identifiants d'unité peuvent identifier la même unité.

Un nœud de graphe RDF possède zéro (ressource anonyme) ou une URI. Nous définissons donc un sous-ensemble des graphes d'unités, les graphes d'unités *réguliers*, qui peuvent faire l'objet d'un aller-retour en RDF.

Définition 111 (Graphe d'unité régulier)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un GU. G est dit *régulier* si et seulement si les deux conditions ci-dessous sont réunies :

– Tout nœud d'unité u possède au plus un marqueur, i.e.,

$$\forall u \in U, |marker(u)| \leq 1$$

– Les marqueurs des nœuds différents sont disjoints, i.e.,

$$\forall u_1, u_2 \in U, u_1 \neq u_2 \implies marker(u_1) \cap marker(u_2) = \varnothing$$

Précisons que si deux nœuds différents ont un élément de marqueur commun, alors on pourrait inférer par la règle axiomatique d'inférence **mrk-eq** qu'ils représentent la même unité. Nous conjecturons que pour tout GU G il existe un GU régulier G^{reg} tel que :

– G peut être déduit de G^{reg} ;

– G^{reg} peut être déduit de G.

Nous pouvons alors proposer un aller-retour entre le formalisme et RDF, basé sur une bijection entre le graphe sous-jacent le graphe d'unité, et un graphe RDF. Nous utilisons l'axiome **SameIndividual** pour représenter la relation d'équivalences déclarées.

Définition 112 (Equivalence entre un graphe d'unité régulier et un graphe RDF)**.** Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un GU régulier. La fonction rdf transforme G de la manière suivante :

- la transformation en RDF de chaque nœud $u \in U$, notée `rdf(u)`, est telle que :
  - u est représenté par un nœud anonyme, `rdf(u)`=`_:u`, si et seulement si $marker(u) = \varnothing$.
  - u est représenté par une ressource d'URI uri(m), `rdf(u)`=`uri(m)`, si et seulement si $marker(u) = \{m\}$.
  - $type(u) = \varnothing$, si et seulement si le graphe RDF contient un triplet `rdf(u) a ug:Unit`.
  - si $type(u) \neq \varnothing$, alors le graphe RDF contient un triplet `rdf(u) a uri(t)` si et seulement si $t \in type(u)$.
- le graphe RDF contient un triplet `rdf(u) uri(s) rdf(v)` si et seulement si G contient un triplet actanciel ou circonstanciel $(u,s,v) \in A \cup C$.
- le graphe RDF contient un triplet `rdf(u) owl:sameAs rdf(v)` si et seulement si G contient une relation d'équivalences déclarées $(u,v) \in Eq$.

Par exemple, le graphe d'unité de la figure 4.3 peut être représenté de la manière suivante.

```
###
# noeuds d'unites
###
_:oper1 a ecd-fr:Oper1 , mtt-fr:indicatif , mtt-fr:present .
ex-fr:o1 a ecd-fr:Orwell .
_:doute a ecd-fr:Doute , mtt-fr:indetermine , mtt-fr:singulier .
_:nepas a ecd-fr:NePas .
_:effet a ecd-fr:Effet , mtt-fr:defini , mtt-fr:singulier .
_:pos2 a ecd-fr:Pos2 .
_:engagement a ecd-fr:Engagement , mtt-fr:defini , mtt-fr singulier .
ex-fr:02 a ecd-fr:Orwell .
_:politique a ecd-fr:Politique , mtt-fr:defini , mtt-fr:singulier .
_:qualite a ecd-fr:Qualite , mtt-fr:defini , mtt-fr:pluriel .
_:oeuvre a ecd-fr:Oeuvre , mtt-fr:defini , mtt-fr:pluriel .
ex-fr:o3 a ecd-fr:Orwell .

###
# triplets actanciels
###
_:oper1 mtt:i ex-fr:o1 .
_:oper1 mtt:ii _:doute .
_:doute mtt:ii _:effet .
_:effet mtt:i _:engagement .
_:effet mtt:ii _:qualite .
_:engagement mtt:i ex-fr:o2 .
_:engagement mtt:ii _:politique .
_:qualite mtt:i _:oeuvre .
_:oeuvre mtt:i ex-fr:o3 .

###
# triplets circonstanciels
###
_:oper1 mtt:attr _:nepas .
_:effet mtt:attr _:pos2 .

###
# relations d'equivalences declarees
###
ex-fr:o1 owl:sameAs ex-fr:o2 .
ex-fr:o2 owl:sameAs ex-fr:o1 .
ex-fr:o2 owl:sameAs ex-fr:o3 .
ex-fr:o3 owl:sameAs ex-fr:o2 .
```

10.2.3.3 Simulation de la déduction

Nous passons maintenant en revue les différentes règles de la base de règles axiomatiques d'inférence de la section 9.2. Nous vérifions ce qui peut être capturé par les règles OWL 2 RL/RDF, et complétons par des règles SPARQL si besoin.

Le tableau 10.2 récapitule la capture de la base de règles axiomatiques d'inférence. Bien que la plupart des règles sont directement capturées par les règles OWL 2 RL/RDF, certaines font exception. Détaillons ces cas particuliers.

Règles de la base axiomatique d'inférence	Règles SPARQL
eq-ref	OWL 2 RL/RDF **eq-ref**
eq-sym	OWL 2 RL/RDF **eq-sym**
eq-trans	OWL 2 RL/RDF **eq-trans**
typ-comp	OWL 2 RL/RDF **cax-sco**
typ-absurd	OWL 2 RL/RDF **cls-int1** et **cax-sco**
a-eq-g	OWL 2 RL/RDF **eq-rep-s**
a-eq-a	OWL 2 RL/RDF **eq-rep-o**
a-fp	OWL 2 RL/RDF **prp-spo1** et **prp-fp**
a-proaslot-root	OWL 2 RL/RDF **cls-maxc1**
a-sig	OWL 2 RL/RDF **cls-avf**
c-eq-g	OWL 2 RL/RDF **eq-rep-s**
c-eq-c	OWL 2 RL/RDF **eq-rep-o**
c-comp	OWL 2 RL/RDF **prp-spo1**
c-sig	OWL 2 RL/RDF **a-sig**
mrk-eq	–
eq-mrk	–
typ-top	–
eq-typ	UG/RDF **eq-typ**
a-aslot-root	UG/RDF **a-aslot-root**
a-oblaslot-root	UG/RDF **a-oblaslot-root**
def+	UG/RDF **def+**
def-	UG/RDF **def-**

TABLE 10.2 – Simulation des règles axiomatiques d'inférence par des règles SPARQL.

Règles mrk-eq et eq-mrk Ces règles ne peuvent pas être capturées en SPARQL, puisqu'elles s'appliquent aux (ou produisent des) GU non réguliers.

Règle eq-typ Cette règle n'a pas d'équivalent direct dans la base OWL 2 RL/RDF, mais peut être capturée par la règle SPARQL suivante.

```
# Rule UG/RDF eq-typ
CONSTRUCT {
  ?v a ?t .
}
WHERE {
  ?u a ?t .
  ?t rdfs:subClassOf ug:Unit .
  ?u owl:sameAs ?v .
}
```

Règle typ-top Cette règle est inapplicable par définition de l'équivalence entre un GU régulier et un graphe RDF. En effet, pour tout nœud d'unité u,

- si le type de u est vide, alors le graphe RDF contient par définition le triplet `rdf(u)a ug:Unit` ;
- sinon, le graphe RDF contient au moins un triplet de la forme `rdf(u)a uri(t)`, où $uri(t)$ représente un type d'unité. Chaque type d'unité devant être défini comme une sous-classe de `ug:Unit`, les règles de OWL 2 RL permettent alors d'inférer que `rdf(u)a ug:Unit`.

Règle a-aslot-root Cette règle n'a pas d'équivalent direct dans la base OWL 2 RL/RDF, mais peut être capturée par la règle SPARQL suivante.

```
# Rule UG/RDF a-aslot-root
CONSTRUCT {
  ?u a ?rapos .
}
WHERE {
  ?s rdfs:subPropertyOf ug:actantialRelation .
  ?s ug:actantSlotRoot ?rapos .
  ?u ?s ?v .
}
```

Règle a-oblaslot-root Cette règle n'a pas d'équivalent direct dans la base OWL 2 RL/RDF, car elle génère un nouveau nœud dans le graphe. Elle peut cependant être capturée par la règle SPARQL suivante. Notons que nous incluons une garde pour capturer la condition d'applicabilité des règles axiomatiques d'inférence (cf., définition 103).

```
# Rule UG/RDF a-oblaslot-root
CONSTRUCT {
  ?u ?s [] .
}
WHERE {
  ?u a ?t .
  ?s ug:obligatoryActantSlotRoot ?t .
  FILTER NOT EXISTS {
    ?u ?s ?v .
  }
}
```

Règles d'expansion et de contraction des définitions de types d'unité Pour chaque définition lexicographique formelle, deux règles SPARQL doivent être générées. Par exemple, les deux règles SPARQL suivantes capturent les règles **def+** et **def-** pour la définition lexicographique formelle de $\ulcorner \text{PEIGNE}_{\text{B.2.D}} \urcorner$. Notons que nous incluons une garde pour capturer la condition d'applicabilité des règles axiomatiques d'inférence (cf., définition 103).

Règle d'expansion **def+**

```
# expansion rule
CONSTRUCT {
  ?u a ex:Outil .
  ?personne a ex:Personne .
  ?activite a ex:Demeler .
  ?fibres a ex:Fibres .
  ?partieDe a ex:PartieDe .
  ?peigne ex:Objet .
  ?u ex:utilisateur ?personne .
  ?u ex:activite ?activite .
  ?activite ex:demeleur ?personne .
  ?activite ex:fibres ?fibres .
  ?partieDe ex:partie ?fibres .
  ?partieDe ex:tout ?peigne .
}
WHERE {
  ?u a ex:peigneB2d .
  ?u ex:utilisateur ?personne .
  ?u ex:activite ?activite .
  ?u ex:peigne ?peigne .
  FILTER NOT EXISTS {
    ?u a ex:Outil .
    ?personne a ex:Personne .
    ?activite a ex:Demeler .
    ?fibres a ex:Fibres .
    ?partieDe a ex:PartieDe .
    ?peigne ex:Objet .
    ?u ex:utilisateur ?personne .
    ?u ex:activite ?activite .
    ?activite ex:demeleur ?personne .
    ?activite ex:fibres ?fibres .
    ?partieDe ex:partie ?fibres .
    ?partieDe ex:tout ?peigne .
  }
}
```

Règle de contraction **def-**

```
# contraction rule
CONSTRUCT {
  ?u a ex:peigneB2d .
  ?u ex:utilisateur ?personne .
  ?u ex:activite ?activite .
  ?u ex:peigne ?peigne .
}
WHERE {
  ?u a ex:Outil .
  ?personne a ex:Personne .
  ?activite a ex:Demeler .
  ?fibres a ex:Fibres .
  ?partieDe a ex:PartieDe .
  ?peigne ex:Objet .
  ?u ex:utilisateur ?personne .
  ?u ex:activite ?activite .
  ?activite ex:demeleur ?personne .
  ?activite ex:fibres ?fibres .
  ?partieDe ex:partie ?fibres .
  ?partieDe ex:tout ?peigne .
  FILTER NOT EXISTS {
    ?u a ex:peigneB2d .
    ?u ex:utilisateur ?personne .
    ?u ex:activite ?activite .
    ?u ex:peigne ?peigne .
  }
}
```

La section suivante introduit la modélisation des définitions lexicographiques, et nous proposerons une méthode -à base de règles- dans la section 10.3.3 pour générer automatiquement la base des règles d'expansion et de contraction.

10.3 Modèle des définitions formelles

Nous souhaitons proposer une modélisation des définitions formelles des types d'unité, selon la conceptualisation proposée dans le chapitre 4. La section 10.3.1 présente un modèle qui prend appui sur le modèle **gu-langue**, et facilite la satisfaction des deux objectifs suivants :

- Les conditions de validité d'une définition formelle (cf., définition 22), doivent être vérifiables facilement. De même, le modèle doit permettre de capturer la validation et l'inférence introduite dans la section 4.2.3. Nous étudierons la satisfaction de ce critère dans la section 10.3.2.
- Le modèle doit faciliter la génération automatique des règles d'expansion et de contraction. Nous proposerons une méthode générique de génération d'une base de règles dans la section 10.3.3.

Nous utilisons en exemple la définition lexicographique formelle de $\lceil \text{PEIGNE}_{\text{B.2.D}} \rceil$ représentée sur la figure 4.12.

10.3.1 Choix de modélisation pour les définitions formelles

Cette section introduit un modèle pour les définitions formelles de types d'unité, selon la conceptualisation proposée dans le chapitre 4.

Modélisation des nœuds de participant Les nœuds des définitions lexicographiques formelles sont des nœuds de participant - au niveau du dictionnaire -, et sont donc d'une autre nature que les nœuds des représentations linguistiques - au niveau des textes -. Cependant, nous avons fait le choix de nous conformer à la vision du monde à deux niveaux des formalismes du Web sémantique : classes et instances.

Nous décidons donc de représenter les définitions lexicographiques formelles au niveau des textes, et de leur adjoindre des méta-données. Nous utilisons donc des unités sémantiques profondes au lieu des nœuds de participant.

Nous introduisons une classe OWL `ug:Participant` pour annoter les unités qui représentent des nœuds de participant.

```
ug:Participant a owl:Class ;
    rdfs:subClassOf ug:Unit ;
    rdfs:label "Participant" ;
    rdfs:comment "The class of units that represent participants of formal lexicographic
        definitions."@en .
```

La représentation de la définition lexicographique de $\lceil \text{PEIGNE}_{\text{B.2.D}} \rceil$ a donc pour base l'ensemble des triplets suivants :

```
_:outil a ug:Participant ;
    a ex:Outil .
_:personne a ug:Participant ;
    a ex:Personne .
_:demeler a ug:Participant ;
    a ex:Demeler .
_:fibres a ug:Participant ;
    a ex:Fibres .
_:partieDe a ug:Participant ;
    a ex:PartieDe .
_:objet a ug:Participant ;
    a ex:Objet .
```

```
13
14 _:outil ex:utilisateur _:personne .
15 _:outil ex:activite _:demeler .
16 _:demeler ex:demeleur _:personne .
17 _:demeler ex:fibres _:fibres .
18 _:partieDe ex:partie _:fibres .
19 _:partieDe ex:tout _:objet .
```

Un nœud de participant peut éventuellement être marqué obligatoire ou interdit, mais pas les deux. Nous introduisons donc deux sous-types disjoints de `ug:Participant` : `ug:ObligatoryParticipant`, et `ug:ProhibitedParticipant`. Le fait que ces deux sous-classes sont disjointes peut être capturé en OWL à l'aide de l'axiome **DisjointClasses**.

```
1 ug:ObligatoryParticipant a owl:Class ;
2     rdfs:subClassOf ug:Participant ;
3     rdfs:label "ObligatoryParticipant" ;
4     rdfs:comment "The class of obligatory participants."@en .
5
6 ug:ProhibitedParticipant a owl:Class ;
7     rdfs:subClassOf ug:Participant ;
8     rdfs:label "ProhibitedParticipant" ;
9     rdfs:comment "The class of prohibited participants."@en .
10
11 ug:ObligatoryParticipant owl:disjointWith ug:ProhibitedParticipant .
```

Ensuite, chaque nœud de participant est associé à un ou plusieurs marqueurs, qui sont des SRelA. Nous introduisons donc une relation `ug:marker` pour lier un participant à chacun de ses marqueurs. Cette relation lie un noeud de participant à un SRelA vu comme un objet.

```
1 ug:marker a owl:ObjectProperty ;
2     rdfs:label "marker" ;
3     rdfs:comment "The subject participant has for marker the object actantial relation
          symbol." ;
4     rdfs:domain ug:Participant ;
5     rdfs:range ug:ActantialRelationSymbol .
```

Enfin, afin de faciliter la génération des règles d'expansion et de contraction associées à une règle, nous associons un nom unique à chaque nœud de participant à l'aide de la propriété d'annotation `rdfs:label`. Nous attribuons aux applications le contrôle de l'unicité de ces noms.

Par exemple, le nœud de participant de type /démêler\ dans la définition de $\lceil \text{PEIGNE}_{\text{B.2.D}} \rceil$ est représenté de la manière suivante.

```
1 _:demeler rdfs:label "demeler" ;
2     a ug:Participant ;
3     a ex:Demeler ;
4     a ug:ObligatoryParticipant ;
5     ug:marker ex:activite .
```

Lien entre un type d'unité et sa définition lexicographique Nous introduisons une relation OWL `ug:definition` pour associer un type d'unité au nœud de participant central de sa définition lexicographique formelle. Finalement, nous représentons l'ensemble des triplets qui représente une définition lexicographique dans un graphe nommé dont l'URI est celle du participant central. Nous verrons dans la prochaine section que cette représentation permet d'écrire plus facilement des règles de validation en SPARQL.

La définition 22 impose que le nœud de participant central soit obligatoire, nous pouvons représenter cette contrainte en spécifiant le co-domaine de `ug:definition` à `ug:ObligatoryParticipant`.

```
ug:definition a owl:ObjectProperty ;
   rdfs:label "definition" ;
   rdfs:comment "The subject unit type has a for central participant of its formal
       lexicographic definition the object participant." ;
   rdfs:domain ug:UnitType ;
   rdfs:range ug:ObligatoryParticipant .
```

Exemple complet Le code suivant représente complètement la définition de $\lceil \text{PEIGNE}_{\text{B.2.D}} \rceil$.

```
ex:peigneB2d ug:definition _:outil .

_:outil {

    _:outil rdfs:label "outil" ;
        a ug:Participant ;
        a ex:Outil .

    _:personne rdfs:label "personne" ;
       a ug:Participant ;
       a ex:Personne ;
       a ug:ObligatoryParticipant ;
       ug:marker ex:utilisateur .

    _:demeler rdfs:label "demeler" ;
       a ug:Participant ;
       a ex:Demeler ;
       a ug:ObligatoryParticipant ;
       ug:marker ex:activite .

    _:fibres rdfs:label "fibres" ;
       a ug:Participant ;
       a ex:Fibres .

    _:partieDe rdfs:label "partieDe" ;
       a ug:Participant ;
       a ex:PartieDe .

    _:objet rdfs:label "objet" ;
       a ug:Participant ;
       a ex:Objet ;
       a ug:ObligatoryParticipant ;
       ug:marker ex:peigne .

    _:outil ex:utilisateur _:personne .
    _:outil ex:activite _:demeler .
    _:demeler ex:demeleur _:personne .
    _:demeler ex:fibres _:fibres .
    _:partieDe ex:partie _:fibres .
    _:partieDe ex:tout _:objet .
}
```

10.3.2 Simulation de la validation et de l'inférence

Parmi les différentes conditions de validité des définitions lexicographiques formelles imposées par la définition 22, nous avons représenté la première, et nous n'avons pas besoin de représenter la seconde puisque nous ne représentons pas la nature des arcs entre nœuds de participant. Quand à la troisième, nous pouvons l'imposer à l'aide de la requête SPARQL suivante :

```
CONSTRUCT {
  GRAPH ?p0 { ?p ug:marker ?s . }
}
WHERE {
  ?s rdfs:subPropertyOf ug:actantialRelation .
  ?t ug:definition ?p0 .
  GRAPH ?p0 { ?p0 ?s ?p . }
}
```

Cette règle ajoute un triplet à un graphe nommé. Il s'agit d'une extension de SPARQL implémentée par le logiciel Corese développé dans l'équipe Wimmics.

Ensuite, le type des nœuds de participant qui possèdent un marqueur doit être obligatoire (resp. interdit) si la PosA associée est obligatoire (resp. interdite). Nous pouvons représenter cette contrainte à l'aide des deux règles SPARQL suivantes, grâce à l'utilisation des graphes nommés.

```
CONSTRUCT {
  GRAPH ?p0 {
        ?p a ug:ObligatoryParticipant .
  }
}
WHERE {
  ?t rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:minCardinality "1"^^xsd:nonNegativeInteger ] ;
    ug:definition ?p0 .
  GRAPH ?p0 { ?p ug:marker ?s . }
}
```

```
CONSTRUCT {
  GRAPH ?p0 {
    ?p a ug:ProhibitedParticipant .
  }
}
WHERE {
  ?t rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:maxCardinality "0"^^xsd:nonNegativeInteger ] ;
   ug:definition ?p0 .
  GRAPH ?p0 { ?p ug:marker ?s . }
}
```

Enfin, les quatre configurations qui impliquent de l'inférence illustrées sur la figure 4.15 peuvent être représentées à l'aide des règles SPARQL suivantes :

```
CONSTRUCT {
  GRAPH ?p0 {
    ?p2 a ug:ObligatoryParticipant .
  }
}
WHERE {
  ?t0 ug:definition ?p0 .
  ?t1 rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:minCardinality
          "1"^^xsd:nonNegativeInteger ] .
  ?s rdfs:subPropertyOf ug:actantialRelation .
  GRAPH ?p0 {
    ?p1 a ?t1 .
    ?p2 a ?t2 .
    ?p1 ?s ?p2 .
    ?p1 a ug:ObligatoryParticipant .
  }
}
```

```
CONSTRUCT {
  GRAPH ?p0 {
    ?p2 a ug:ProhibitedParticipant .
  }
}
WHERE {
  ?t0 ug:definition ?p0 .
  ?t1 rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:maxCardinality
          "0"^^xsd:nonNegativeInteger ] .
  ?s rdfs:subPropertyOf ug:actantialRelation .
  GRAPH ?p0 {
    ?p1 a ?t1 .
    ?p2 a ?t2 .
    ?p1 ?s ?p2 .
    ?p1 a ug:ObligatoryParticipant .
  }
}
```

```
CONSTRUCT {
  GRAPH ?p0 {
    ?p1 a ug:ProhibitedParticipant .
  }
}
WHERE {
  ?t0 ug:definition ?p0 .
  ?t1 rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:minCardinality
          "1"^^xsd:nonNegativeInteger ] .
  ?s rdfs:subPropertyOf ug:actantialRelation .
  GRAPH ?p0 {
    ?p1 a ?t1 .
    ?p2 a ?t2 .
    ?p1 ?s ?p2 .
    ?p2 a ug:ProhibitedParticipant .
  }
}
```

```
CONSTRUCT {
  GRAPH ?p0 {
    ?p1 a ug:ProhibitedParticipant .
  }
}
WHERE {
  ?t0 ug:definition ?p0 .
  ?t1 rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?s ;
     owl:maxCardinality
          "0"^^xsd:nonNegativeInteger ] .
  ?s rdfs:subPropertyOf ug:actantialRelation .
  GRAPH ?p0 {
    ?p1 a ?t1 .
    ?p2 a ?t2 .
    ?p1 ?s ?p2 .
    ?p2 a ug:ObligatoryParticipant .
  }
}
```

10.3.3 Génération des règles d'expansion et de contraction

Nous souhaitons générer des règles SPARQL à partir de la description d'un DEC. Nous envisageons deux points de départ pour cela :

- utiliser une requête SPARQL pour générer une représentation RDF des règles d'expansion et de contraction à l'aide de la syntaxe SPIN [5] de SPARQL ;
- utiliser les requêtes SPARQL TEMPLATE de Corby *et al.* (2014), qui génèrent du texte à partir d'un graphe RDF de manière récursive.

Nous présentons dans cette section une approche basée sur les requêtes SPARQL TEMPLATE. La base de règles SPARQL TEMPLATE que nous avons développée est retranscrite dans l'annexe D.1.5, et disponible à l'URL <http://ns.inria.org/ug/v1/ug-definition-rules-generation-rules.rul>.

Point de départ de la génération de la base de règles La règle SPARQL TEMPLATE suivante est le point d'entrée de la génération de la base de règles SPARQL.

```
# generates the contraction and expansion rules set
# starting point, called once.
TEMPLATE st:start {
  "<?xml version='1.0' encoding='UTF-8'?> \n"
  " <rules xmlns='http://ns.inria.org/edelweiss/2011/rule#'> \n"
  " "st:call-template(ug:printContractionRule)
  " "st:call-template(ug:printExpansionRule)
  " </rules> \n"
}
WHERE { }
```

Nous appelons cette règle à l'aide de la requête SPARQL SELECT suivante.

```
SELECT
    (st:apply-template-with(<http://ns.inria.org/ug/v1/ug-definition-rules-generation-rules.rul>)
    as ?rules)
WHERE { }
```

La règle SPARQL TEMPLATE suivante génère la règle de contraction **def-** associée à chaque définition formelle. Appliquée au graphe de la section 10.3.1, cette règle génère la règle **def-** de la section 10.2.3.3.

```
# generates the contraction rule
# called for every formal definition unit type.
TEMPLATE ug:printContractionRule {
  "<rule><value><![CDATA[ \n"
  "CONSTRUCT { \n"
    st:call-template(ug:printPrint,?t,?def)
  "} WHERE { \n"
    st:call-template(ug:printExpansion,?t,?def)
  " FILTER NOT EXISTS { \n"
      st:call-template(ug:printPrint,?t,?def)
  " } \n"
  "}]" "]></value></rule> \n"
}
WHERE { ?t ug:definition ?def . }
```

5. SPIN - SPARQL Inference Notation - http://www.w3.org/Submission/spin-overview/

La règle SPARQL TEMPLATE suivante génère une règle d'expansion **def+** associée à chaque définition formelle. Appliquée au graphe de la section 10.3.1, cette règle génère la règle **def+** de la section 10.2.3.3.

```
# generates the expansion rule
# called for every formal definition unit type.
TEMPLATE ug:printExpansionRule(?t,?def) {
  "<rule><value><![CDATA[ \n"
  "CONSTRUCT { \n"
    st:call-template(ug:printExpansion,?t,?def)
  "} WHERE { \n"
    st:call-template(ug:printPrint,?t,?def)
  " FILTER NOT EXISTS { \n"
      st:call-template(ug:printExpansion,?t,?def)
  " } \n"
  "}]" "]></value></rule> \n"
}
WHERE { ?t ug:definition ?def . }
```

Génération du template de l'empreinte du type défini La règle SPARQL TEMPLATE suivante génère une représentation de l'empreinte du type d'unité défini, avec une variable pour chaque nœud de participant. Appliquée au graphe de la section 10.3.1, cette règle génère le pattern WHERE de la règle **def+**, et le pattern CONSTRUCT de la règle **def-** de la section 10.2.3.3.

```
# generates the print pattern
TEMPLATE ug:printPrint(?t,?def) {
  "?"xsd:string(?label) " a " st:turtle(?def) " . \n"
  st:call-template(ug:printActantSlot,?def,?centralname)
}
WHERE {
  GRAPH ?def {
    ?def rdfs:label ?centralname .
  }
}
```

Notons que la génération des variables est facilitée grâce au nommage des nœuds de participant. Par exemple le nœud de participant `_:personne` associé au nom `"personne"` générera une variable `?personne`.

La règle SPARQL TEMPLATE suivante génère la représentation des positions actancielles du type défini.

```
# generates the description of actantial positions of the defined unit type
TEMPLATE ug:printPrintActantSlot(?def,?centralname) {
  "?"xsd:string(?centralname) " " st:turtle(?marker) " " "?"xsd:string(?name) " . \n"
}
WHERE {
  GRAPH ?def {
    ?u ug:marker ?marker .
    ?u rdfs:label ?name .
  }
}
```

Génération du template de l'expansion du type défini La règle SPARQL TEMPLATE suivante génère une représentation de l'expansion du type d'unité défini, avec une variable pour chaque nœud de participant. Appliquée au graphe de la section 10.3.1, cette règle génère le pattern CONSTRUCT de la règle **def+**, et le pattern WHERE de la règle **def-** de la section 10.2.3.3.

```
# generates the expansion pattern
TEMPLATE ug:printExpansion(?t,?def) {
  st:call-template(ug:printExpansionNode,?def)
  st:call-template(ug:printExpansionArc,?def)
}
WHERE { }
```

La règle SPARQL TEMPLATE suivante génère une représentation de chaque nœud de participant.

```
# generates the description of each participant node
TEMPLATE ug:printExpansionNode(?def) {
  "?"xsd:string(?centralname) " a " st:turtle(?t) " . \n"
}
WHERE {
  GRAPH ?def {
    ?u a ?t .
    ?u rdfs:label ?name .
  }
  FILTER NOT EXISTS ( ?t rdfs:subClassOf ug:Participant )
}
```

La règle SPARQL TEMPLATE suivante génère une représentation de chacun des arcs entre les nœuds de participant.

```
# generates the description of each actantial triple
TEMPLATE ug:printExpansionArc(?def) {
"?"xsd:string(?uname) st:turtle(?p) "?"xsd:string(?vname) " . \n"
}
WHERE {
  ?p rdfs:subPropertyOf ug:actantialRelation .
  GRAPH ?def {
    ?u ?p ?v .
    ?u rdfs:label ?uname .
    ?v rdfs:label ?vname .
  }
}
```

10.4 Instanciation pour la Théorie Sens-Texte

Nous avons détaillé le modèle générique des GU. Cette section détaille maintenant une instanciation pour la modélisation de la théorie Sens-Texte. Nous détaillons le contenu des différentes ontologies du modèle proposé, que nous avons décrit dans la section 10.1.3 :

- La section 10.4.1 détaille l'ontologie générique pour le modèle Sens-Texte, indépendante de la langue et des dictionnaires. C'est cette ontologie que nous pouvons aligner avec le modèle **ontolex** du groupe Ontology-Lexica.
- La section 10.4.2 détaille les ontologies dépendantes de la langue, mais indépendantes des dictionnaires. Nous pouvons aligner ces ontologies avec les registres de Data Category.
- La section 10.4.3 détaille les ontologies qui modélisent les DEC, et celles qui modélisent des faits.

10.4.1 L'ontologie du modèle Sens-Texte

Nous proposons une ontologie pour le modèle Sens-Texte, qui importe l'ontologie du modèle des GU. C'est cette ontologie que nous pourrons aligner avec le modèle **ontolex**. Nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt#>`, avec le préfixe `mtt:`.

```
<http://ns.inria.org/ug/v1/mtt#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1#> ;
    dc:title "The Meaning-Text Theory ontology (MTT)."@en .
```

Cette ontologie est retranscrite dans l'annexe D.2.1, et accessible en ligne à l'URL susmentionnée. Nous y décrivons les différentes catégories de types d'unité pour les différents niveaux de représentation linguistique (§10.4.1.1), les relations actancielles et circonstancielles universelles de la TST (§10.4.1.2), et deux ensembles de relations additionnelles (§10.4.1.3) :

- des relations qui lient des types d'unité de niveaux adjacents dans le dictionnaire, par exemple $^{(}$to eat$^{)}$ et $^{/}$to eat$^{\backslash}$;
- des relations circonstancielles qui lient des unités de niveaux adjacents dans le discours, par exemple $^{(}$to eat$^{)}$: et $^{/}$to eat$^{\backslash}$:.

10.4.1.1 Types d'Unité linguistiques

L'ontologie du modèle Sens-Texte propose tout d'abord une catégorisation des types d'unité suivant les différents niveaux de représentation linguistique :

- les types d'unité sémantique profonde ;
- les types d'unité sémantique de surface ;
- les types d'unité lexicale ;
- les types d'unité grammaticale.

Prenons l'exemple des types d'unité sémantique profonde. Nous définissons :

- une classe OWL `mtt:DeepSemanticUnitType`, sous-classe de `ug:UnitType` ;
- une classe OWL `mtt:DeepSemanticUnit`, instance de la première, et sous-classe de la classe OWL `ug:Unit`.

```
mtt:DeepSemanticUnitType a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
    rdfs:label "DeepSemanticUnitType" ;
    rdfs:comment "The super class of all deep semantic unit types."@en .
```

```
5
6  mtt:DeepSemanticUnit a owl:Class ;
7      a mtt:DeepSemanticUnitType ;
8      rdfs:subClassOf ug:Unit ;
9      rdfs:label "DeepSemanticUnit" ;
10     rdfs:comment "The class of all deep semantic units."@en .
```

Tout type d'unité sémantique profond est plus spécifique que `mtt:DeepSemanticUnit`. Nous pouvons représenter cette contrainte à l'aide de l'axiome **ObjectHasValue**.

```
1 mtt:DeepSemanticUnitType rdfs:subClassOf [ a owl:Restriction ;
2        owl:onProperty ug:specializationOf ;
3        owl:hasValue mtt:DeepSemanticUnit ] .
```

Nous définissons de la même manière le modèle des autres catégories de types d'unité :

- `mtt:SurfaceSemanticUnitType` et `mtt:SurfaceSemanticUnit` ;
- `mtt:LexicalUnitType` et `mtt:LexicalUnit` ;
- `mtt:GrammaticalUnitType` et `mtt:GrammaticalUnit` ;

Dans la langue, un type d'unité ne peut pas appartenir à deux catégories à la fois. Cette connaissance est capturée à l'aide de l'axiome **DisjointClasses**.

```
1 [] rdf:type owl:AllDisjointClasses ;
2   owl:members (
3     mtt:LexicalUnitType
4     mtt:GrammaticalUnitType
5     mtt:SurfaceSemanticUnitType
6     mtt:DeepSemanticUnitType ) .
```

Enfin, dans le discours, une unité appartient au maximum à l'un des cas suivants :

- instance d'un type d'unité lexicale et d'un type d'unité grammaticale ;
- instance d'un type sémantique de surface ;
- instance d'un type sémantique profond.

Cette connaissance est capturée à l'aide de l'axiome **DisjointClasses** de la manière suivante.

```
1 mtt:LexicalUnit owl:disjointWith mtt:SurfaceSemanticUnit .
2 mtt:LexicalUnit owl:disjointWith mtt:DeepSemanticUnit .
3 mtt:GrammaticalUnit owl:disjointWith mtt:SurfaceSemanticUnit .
4 mtt:GrammaticalUnit owl:disjointWith mtt:DeepSemanticUnit .
5 mtt:SurfaceSemanticUnit owl:disjointWith mtt:DeepSemanticUnit .
```

Alignement avec le modèle ontolex. Dans le modèle **ontolex**, la classe `ontolex:LexicalEntry` correspond à un type d'unité lexicale. Nous proposons donc l'alignement suivant avec le modèle **ontolex**.

```
1 ug:LexicalUnitType owl:equivalentClass ontolex:LexicalEntry .
```

Ensuite, nous avons montré que le niveau sémantique profond est celui qui porte les sens dans l'extension de la conceptualisation de la théorie Sens-Texte que nous proposons. Ainsi, la classe `ontolex:LexicalSense` correspond-elle aux types d'unité sémantique profonde. Nous proposons donc l'alignement suivant avec le modèle **ontolex**.

```
1 ug:DeepSemanticUnitType owl:equivalentClass ontolex:LexicalSense .
```

10.4.1.2 Relations universelles et structure actancielle

L'ontologie du modèle Sens-Texte contient ensuite les relations actancielles et circonstancielles de la TST qui sont universelles.

Relations syntaxiques profondes universelles. L'ensemble des relations syntaxiques profondes est considéré indépendant de la langue et des dictionnaires. Nous représentons donc chacune de ces relations dans l'ontologie `mtt:`. Par exemple, les codes suivants définissent le SRelA **I** et le SRelC **ATTR**.

```
mtt:i a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "i" ;
    rdfs:comment "The deep syntactic actantial relation symbol i."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .
```

```
mtt:attr a owl:ObjectProperty
    rdfs:subPropertyOf ug:circumstantialRelation ;
    rdfs:label "attr" ;
    rdfs:comment "The deep syntactic circumstantial relation symbol attr."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .
```

Relations sémantiques de surface universelles. L'ensemble des relations sémantiques de surface est considéré indépendant de la langue et des dictionnaires. Nous représentons donc chacune de ces relations dans l'ontologie `mtt:`. Par exemple, le code suivant définit le SRelA 1.

```
mtt:1 a ug:ActantialRelationSymbol ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "1" ;
    rdfs:comment "The surface semantics actantial relation symbol 1."@en ;
    rdfs:domain mtt:SurfaceSemanticUnit ;
    rdfs:range mtt:SurfaceSemanticUnit .
```

Alignement des positions actancielles avec le modèle ontolex. Le modèle **ontolex** possède un module pour représenter les frames sémantiques et syntaxiques des unités lexicales [6]. Quand bien même la conceptualisation de la linguistique que ce modèle représente n'est pas celle de la TST, remarquons que l'extension de la conceptualisation de la TST que nous avons proposée s'en rapproche. En effet, dans le modèle **ontolex**, les *arguments sémantiques* sont associés via la relation `ontolex-synsem:semArg` aux instances de `ontolex:LexicalSense`. Nous avons conceptualisé la structure actancielle sémantique profonde de manière similaire, en associant une position actancielle aux types d'unité sémantique profonde directement. Nous proposons donc l'alignement suivant avec le modèle **ontolex**.

```
ontolex-synsem:semArg rdfs:subPropertyOf ug:actantSlot .
```

Si un type d'unité est sémantique profond, et possède une relation de type `ug:actantSlot`, alors il possède une relation de type `ontolex-synsem:semArg`. Nous utilisons la règle SPARQL suivante pour capturer cette connaissance.

6. Module **ontolex-synsem**, en cours de spécification, http://www.w3.org/community/ontolex/wiki/Syntax_and_Semantics_Module

```
CONSTRUCT {
  ?t ontolex-synsem:semArg ?s .
}
WHERE {
  ?t a mtt:DeepSemanticUnitType .
  ?t ug:actantSlot ?s .
}
```

Gardons à l'esprit que le modèle des GU possède une sémantique plus riche pour les PosA que le modèle **ontolex**. Le fait que deux types d'unité sémantique profonde soient liés à un même individu OWL via la relation `ug:actantSlot` ne porte pas la même sémantique dans le modèle **ontolex** ou dans le modèle des GU. Nous devrons donc être particulièrement prudents lorsque l'on plongera un lexique qui utilise le modèle **ontolex** dans le modèle des GU.

Alignement des signatures avec le modèle ontolex. Le modèle **ontolex-synsem** propose de signer les *frames sémantiques* à l'aide de la relation `ontolex-synsem:isA`, qui lie un type d'unité sémantique profond à une *frame sémantique*. L'exemple suivant déclare que la première PosA de `<#sense1>` est signée `<#sense1>` (la propriété `ontolex-synsem:subjOfProp` est une sous-propriété de la propriété `ontolex-synsem:semArg`) :

```
<#sense1> a ontolex:LexicalSense ;
  ontolex-synsem:subjOfProp <#arg0> .

<#sense2> a ontolex:LexicalSense ;
  ontolex-synsem:isA <#arg0> .
```

Nous utilisons la règle SPARQL suivante pour assurer l'alignement entre le modèle **ontolex** et le modèle **gu-langue**.

```
CONSTRUCT {
  ?t1 ug:specializationOf [
    a ug:Signature ;
    ug:onActantSlot ?s ;
    ug:allUnitsFrom ?t2 ] .
}
WHERE {
  ?t1 a ontolex:LexicalSense ;
    ontolex-synsem:semArg ?s .
  ?t2 a ontolex:LexicalSense ;
    ontolex-synsem:isA ?s .
}
```

```
CONSTRUCT {
  ?t1 a ontolex:LexicalSense ;
    ontolex-synsem:semArg ?s .
  ?t2 a ontolex:LexicalSense ;
    ontolex-synsem:isA ?s .
}
WHERE {
  ?t1 ug:specializationOf [
    a ug:Signature ;
    ug:onActantSlot ?s ;
    ug:allUnitsFrom ?t2 ] .
}
```

10.4.1.3 Relations entre les unités et types d'unité de niveaux adjacents

L'ontologie du modèle Sens-Texte introduit deux ensembles de relations additionnelles :

- des relations qui lient des types d'unité de niveaux adjacents dans le dictionnaire, par exemple ⟨to eat⟩ et /to eat\ ;
- des relations circonstancielles qui lient des unités de niveaux adjacents dans le discours, par exemple ⟨to eat⟩: et /to eat\:.

Relations entre types d'unité linguistique. Nous introduisons deux relations :

- `mtt:surfaceSense` lie un type d'unité lexicale à un type d'unité sémantique de surface associé.

– `mtt:deepSense` lie un type d'unité sémantique de surface à un type d'unité sémantique profonde associé.

```
mtt:surfaceSense a owl:ObjectProperty ;
    rdfs:label "surfaceSense" ;
    rdfs:comment "The subject, a lexical unit type, is meaningful and is associated to the
        object, a suface semantic unit type."@en ;
    rdfs:domain mtt:LexicalUnitType ;
    rdfs:range mtt:SurfaceSemanticUnitType .

mtt:deepSense a owl:ObjectProperty ;
    rdfs:label "deepSense" ;
    rdfs:comment "The subject, a surface semantic unit type, is associated to the object, a
        deep semantic unit type."@en ;
    rdfs:domain mtt:SurfaceSemanticUnitType ;
    rdfs:range mtt:DeepSemanticUnitType .
```

Le niveau sémantique de surface représente une indirection dans le modèle Sens-Texte qui n'existe pas dans le modèle **ontolex**. Nous proposons donc l'alignement suivant :

```
ontolex:sense owl:propertyChainAxiom ( mtt:surfaceSenseType mtt:deepSenseType ) .
```

Par exemple, si l'on considère le lexique suivant :

```
ex:lut-toEat a mtt:LexicalUnitType .
ex:ssem-toEat a mtt:SurfaceSemanticUnitType .
ex:dsem-toEat a mtt:DeepSemanticUnitType .
ex:lut-toEat mtt:surfaceSenseType ex:ssem-toEat .
ex:ssem-toEat mtt:deepSenseType ex:dsem-toEat .
```

Alors on peut inférer la relation `ontolex:sense` suivante :

```
ex:lut-toEat ontolex:sense ex:dsem-toEat .
```

Relations entre unités linguistiques. Nous introduisons deux SRelC, qui permettent de lier deux unités de deux niveaux de représentation adjacents dans un GU :

– `mtt:sSense` lie une unité lexicale à une unité sémantique de surface associée.

– `mtt:dSense` lie une unité sémantique de surface à une unité sémantique profonde associée.

```
mtt:sSense a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:circumstantialRelation ;
    rdfs:label "sSense" ;
    rdfs:comment "The subject, a lexical unit, is meaningful and is associated to the object,
        a suface semantic unit."@en ;
    rdfs:domain mtt:LexicalSemanticUnit ;
    rdfs:range mtt:SurfaceSemanticUnit .

mtt:dSense a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:circumstantialRelation ;
    rdfs:label "dSense" ;
    rdfs:comment "The subject, a surface semantic unit, is associated to the object, a deep
        semantic unit."@en ;
    rdfs:domain mtt:SurfaceSemanticUnit ;
    rdfs:range mtt:DeepSemanticUnit .
```

10.4.2 Modèle Sens-Texte d'une langue

Les types d'unité grammaticaux et les relations syntaxiques de surface sont dépendantes d'une langue, mais indépendantes des dictionnaires écrits pour cette langue. Pour une langue donnée, ces connaissances sont consignées dans une ontologie qui importe (par transitivité) :

- l'ontologie du modèle Sens-Texte ;
- l'ontologie des graphes d'unités.

Ces ontologies peuvent être alignées avec les catégories de données. Pour illustrer les exemples de cette thèse, nous avons défini deux petites ontologies :

- pour le français, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/fr#>`, avec le préfixe `mtt-fr` ;
- pour l'anglais, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/en#>`, avec le préfixe `mtt-en`.

Par exemple pour l'anglais :

```
<http://ns.inria.org/ug/v1/mtt/en#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt#> ;
    dc:title "The English Meaning-Text Theory ontology (MTT-en)."@en .
```

Ces ontologies sont retranscrites dans les annexes D.2.3.1 et D.2.3.2, et accessibles en ligne aux URL sus-mentionnées.

Relations syntaxiques. Les relations syntaxiques de surface sont dépendantes d'une langue, mais indépendantes des dictionnaires écrits pour cette langue. Par exemple, le code suivant déclare deux relations syntaxiques de surface utilisées dans la représentation linguistique de la figure 1.4.

```
mtt-fr:subjectale a owl:ObjectProperty ;
  a ug:ActantialRelationSymbol ;
  rdfs:subPropertyOf ug:actantialRelation .

mtt-fr:modificative a owl:ObjectProperty ;
  rdfs:subPropertyOf ug:circumstantialRelation .
```

Déterminer si une relation syntaxique de surface est actancielle ou circonstancielle ne rentre pas dans le cadre d'étude de cette thèse.

Utilisation des catégories de données. L'implémentation ISOcat du standard ISO:12620 attribue à chaque catégorie de données une URI dans l'espace de nom `<http://www.isocat.org/datcat/>`. Nous utilisons le préfixe `isocat:`.

```
@prefix isocat: <http://www.isocat.org/datcat/>
```

Par exemple, l'URI `isocat:DC-2276` correspond à la catégorie *nom dénombrable*. Dans l'ontologie `mtt-en`, nous utilisons la surcharge de OWL pour définir `isocat:DC-2276` comme un type d'unité grammaticale.

```
isocat:DC-2276 a owl:Class ;
  rdfs:label "countable noun"@en ;
  rdfs:label "nom dénombrable"@fr ;
  a mtt:GrammaticalUnitType ;
  rdfs:subClassOf mtt:GrammaticalUnit .
```

Ainsi, le code suivant déclare par exemple que le type de lexie ⌈CHEVAL⌉ est un nom dénombrable.

```
ecd-fr:lut-Cheval ug:specializationOf isocat:DC-2276 .
```

Déterminer l'adéquation des catégories de données pour le modèle Sens-Texte d'une langue ne rentre pas dans le cadre d'étude de cette thèse. Notons simplement que nous pouvons représenter la hiérarchie des types d'unité grammaticale de la section 3.2.1 comme suit.

```
mtt-fr:Nombre rdfs:subClassOf mtt:GrammaticalUnit .
mtt-fr:Pluralisable rdfs:subClassOf mtt-fr:Nombre .
mtt-fr:Singulier rdfs:subClassOf mtt-fr:Pluralisable .
mtt-fr:Pluriel rdfs:subClassOf mtt-fr:Pluralisable .
mtt-fr:Singulier owl:disjointWith mtt-fr:Pluriel .
```

Alignement avec le modèle ontolex. Dans le modèle **ontolex**, le lien en un type linguistique et une catégorie de données est explicité par la relation `ontolex-synsem:linguisticProperty`. Par exemple, le code suivant déclare que le type de lexie ⌈CHEVAL⌉ est un nom dénombrable.

```
ecd-fr:lut-Cheval a ontolex:LexicalEntry ;
  ontolex-synsem:partOfSpeech isocat:DC-2276 .
```

Ainsi, l'alignement avec le modèle **ontolex** peut s'exprimer de la manière suivante :

```
ontolex-synsem:partOfSpeech rdfs:subPropertyOf ug:specializationOf .
```

Nous pouvons alors inférer de l'exemple ci-dessus que :
- `ecd-en:Cat` spécialise `isocat:DC-1333` dans le modèle **gu-langue** ;
- grâce aux règles de transformation, que `ecd-en:Cat` est une sous-classe de `isocat:DC-1333` dans le modèle **gu-usage**.

```
ecd-fr:lut-Cheval ug:specializationOf a isocat:DC-1333 .
ecd-fr:lut-Cheval rdfs:subClassOf a isocat:DC-1333 .
```

Une direction future de travail est d'étudier plus en détail les catégories de données de ISOcat, et leur utilisation dans le modèle Sens-Texte. Une limite possible de la norme ISO-12620 est qu'une même URL peut être utilisée pour décrire des catégories de données de plusieurs langues, avec des sémantiques différentes.

10.4.3 L'ontologie d'un DEC particulier, et l'ontologie des faits

Ontologie d'un DEC. Un dictionnaire d'une langue donnée est représenté par une ontologie qui importe (par transitivité) :
- l'ontologie du modèle Sens-Texte spécifique à la langue ;
- l'ontologie du modèle Sens-Texte ;
- l'ontologie des graphes d'unités.

Pour illustrer les exemples de cette thèse, nous avons défini deux petites ontologies :
- pour le français, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/fr/ecd#>`, avec le préfixe `ecd-fr` ;
- pour l'anglais, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/en/ecd#>`, avec le préfixe `ecd-en`.

Par exemple pour l'anglais :

```
<http://ns.inria.org/ug/v1/mtt/en/ecd#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/en#> ;
    dc:title "A small English Explanatory and Combinatorial Dictionary (ECD-en)."@en .
```

Ces ontologies sont retranscrites dans les annexes D.2.4.1 et D.2.4.2, et accessibles en ligne aux URL sus-mentionnées.

Par exemple, le code suivant représente la catégorisation grammaticale des types de lexie ⌈FOOTBALL⌉, ⌈LUNETTES⌉, et ⌈CHEVAL⌉, telle que décrite dans la section 3.2.1.

```
ecd-fr:lut-Football a owl:Class ;
  a mtt:LexicalUnitType ;
  rdfs:subClassOf mtt:LexicalUnit ;
  rdfs:subClassOf mtt-fr:Singulier .

ecd-fr:lut-Lunettes a owl:Class ;
  a mtt:LexicalUnitType ;
  rdfs:subClassOf mtt:LexicalUnit ;
  rdfs:subClassOf mtt-fr:Pluriel .

ecd-fr:lut-Cheval a owl:Class ;
  a mtt:LexicalUnitType ;
  rdfs:subClassOf mtt:LexicalUnit ;
  rdfs:subClassOf mtt-fr:Pluralisable .
```

Ontologie des faits. Enfin, une représentation linguistique d'une langue donnée peut être modélisé par une ontologie qui importe (par transitivité) :

- l'ontologie d'un DEC ;
- l'ontologie du modèle Sens-Texte spécifique à la langue ;
- l'ontologie du modèle Sens-Texte ;
- l'ontologie des graphes d'unités.

Pour illustrer les exemples de cette thèse, nous avons défini deux petites ontologies :

- pour le français, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/fr/exemple#>`, avec le préfixe `ex-fr` ;
- pour l'anglais, nous utilisons l'espace de nommage `<http://ns.inria.org/ug/v1/mtt/en/exemple#>`, avec le préfixe `ex-en`.

Par exemple pour l'anglais :

```
<http://ns.inria.org/ug/v1/mtt/en/exemple#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/en/ecd#> ;
    dc:title "A small English linguistic representation."@en .
```

L'ontologie de l'anglais est retranscrite dans l'annexe D.2.5. Ces deux ontologies sont accessibles en ligne aux URL sus-mentionnées.

Par exemple, le graphe d'unité de la figure 4.3, dont la représentation a été proposée dans la section 10.2.3.2, fait partie de l'ontologie `ex-fr`.

Conclusion

Nous avons étudié l'opérationnalisation de nos travaux sur le Web des données lexicales.

Au vu des exigences du formalisme des Graphes d'Unités, nous avons justifié le choix d'utiliser le métamodèle OWL 2 RL, et d'étendre la base de règles OWL 2 RL/RDF par un ensemble de règles SPARQL pour capturer la sémantique de notre formalisme. Nous avons alors justifié l'utilisation de deux modèles hétérogènes :

- le modèle **gu-langue** adopte le point de vue de la compétence, et décrit les types d'unité et les SRelA au niveau assertionnel. Il s'agit du point de vue adopté par les ontologies lexicales du Web des données.
- Le modèle **gu-usage**, au contraire, adopte le point de vue de la performance. Les types d'unité et les SRelA sont décrits au niveau terminologique, laissant le niveau assertionnel aux graphes d'unités.

Ces modèles sont rendus interopérables par un ensemble de règles de transfert, qui définissent un morphisme d'ontologies au sens de Flouris *et al.* (2008).

Nous avons donc détaillé le modèle double du formalisme des GU. Pour chaque aspect du formalisme, nous avons proposé une modélisation selon les deux points de vue : langue et usage, et défini des règles de transfert pour assurer la cohérence et l'interopérabilité entre les deux modèles. Nous avons alors montré que le critère d'acyclicité du graphe $G_{\mathscr{T}}$, suffisant pour assurer la décidabilité du problème de déduction logique en l'absence de définitions lexicographiques, peut s'exprimer à l'aide d'une règle SPARQL ASK simple.

Le modèle **gu-usage** est adapté à la représentation des GU. Nous avons passé en revue les différentes règles de la base de règles axiomatiques d'inférence de la section 9.2, et avons montré que la base de règles OWL 2 RL/RDF permet de capturer la plupart de la sémantique du formalisme des GU. Afin de compléter cette capture, nous avons ajouté trois règles indépendantes des dictionnaires, ainsi qu'une règle de contraction et une règle d'expansion pour chaque définition formelle d'un dictionnaire particulier.

Puisque les définitions lexicographiques formelles telles que nous les avons conceptualisées dans la partie II ne sont pas complètement représentées par le formalisme des GU, nous en avons proposé une représentation directement en RDF. La modélisation proposée est intéressante à deux points de vue :

- nous pouvons capturer la validation et l'inférence dans les définitions lexicographiques, en accord avec les résultats du chapitre 4 ;
- nous avons introduit un petit ensemble de règles SPARQL TEMPLATE, qui permet de générer automatiquement une base de règles d'expansion et de contraction à partir d'un dictionnaire.

Finalement, nous avons spécialisé le modèle générique du formalisme des GU pour la théorie Sens-Texte, et pour illustrer les exemples de ce mémoire. Nous avons proposé une organisation pyramidale des modèles :

- une spécialisation du modèle des GU pour le modèle Sens-Texte. Nous avons proposé un alignement de cette ontologie avec le modèle **ontolex** développé par le groupe communautaire Ontology-Lexica ;
- une spécialisation supplémentaire pour chaque langue ;

– une spécialisation supplémentaire pour chaque dictionnaire ;
– une spécialisation pour chaque graphe d'unité (chaque énoncé).

Nous avons donc instancié cette pyramide d'ontologies pour illustrer les exemples de ce mémoire, et publié en ligne nos résultats à l'adresse <http://ns.inria.fr/ug/v1#>.

Ce chapitre répond donc aux attentes présentées dans la section 2.4.2. Nous avons présenté une modélisation du formalisme des GU avec les formalismes du Web sémantique. Nous pouvons ainsi profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances. Notons que le logiciel Corese (Corby *et al.*, 2004) développé dans l'équipe Wimmics est parfaitement adapté pour implémenter le formalisme des Graphes d'Unités, et ce sans qu'aucune modification ne soit nécessaire.

Enfin, nous avons proposé une spécialisation du modèle des GU pour la TST, que nous avons alignée avec le modèle **ontolex** proposé par le groupe communautaire Ontology-Lexica. Nous espérons retirer de cette approche :

– une adoption facilitée de nos travaux par la communauté de la représentation des connaissances lexicales ;
– la possibilité de réutiliser des données existantes pour peupler notre formalisme.

CHAPITRE 11

Perspectives

__U__n problème livré à lui-même se dessèche ou pourrit. Mais fertilisez un problème à l'aide d'une solution et vous allez en faire éclore des dizaines.
Norman Frederik Simpson, Un tintement tonitruant

Sommaire

Introduction

Dans ce mémoire, nous avons présenté notre étude d'ingénierie des connaissances appliquée à la TST. Nous avons régulièrement précisé les limites de notre cadre d'étude, ce qui nous a permis d'aboutir à une itération complète pour un sous-ensemble des connaissances de la TST, de l'extension de la conceptualisation à l'implémentation sur le Web des données.

Ce chapitre ouvre progressivement notre cadre d'étude, et passe donc en revue différentes perspectives de recherche qu'offrent nos travaux. Nous adoptons une organisation du spécifique au général, et du court terme au long terme.

Nous présentons dans un premier temps les approfondissements et extensions envisagées de notre étude d'ingénierie des connaissances :

- spécifiquement pour la représentation des définitions lexicographiques (§11.1) ;
- pour la représentation d'autres aspects du modèle Sens-Texte (§11.2). Nous nous intéresserons plus particulièrement aux perspectives qui concernent la représentation des modules de correspondance sens-texte.

Nous détaillons ensuite différentes perspectives applicatives pour nos travaux :

- des applications prenant directement appui sur le formalisme des GU et son opérationnalisation dans leur état actuel (§11.3) ;
- des applications à plus long terme, nécessitant différents approfondissements de nos travaux (§11.4).

Finalement, nous introduisons une perspective de travail théorique concernant les méthodologies respectives de la lexicologie/lexicographie et de l'ingénierie des connaissances (§11.5).

11.1 Enrichissement de la représentation des définitions lexicographiques

Nous avons régulièrement restreint le cadre d'étude de nos travaux jusqu'à l'étude de la représentation des définitions lexicographiques ayant exclusivement des nœuds de participant obligatoire. Nous revenons maintenant sur ces restrictions, et envisageons comment elles pourraient être levées à la lumière de nos travaux. Nous choisissons une organisation inverse à l'organisation générale de ce mémoire :

- approfondissement de l'étude de l'opérationnalisation du formalisme des GU en présence de définitions de TUP (§11.1.1) ;
- amélioration de la représentation des définitions lexicographiques dans le formalisme des GU, selon leur conceptualisation actuelle (§11.1.2) ;
- conceptualisation complète des définitions lexicographiques de la TST (§11.1.3).

11.1.1 Approfondissement de l'étude du raisonnement logique

L'étude de la déduction logique en présence de définitions de TUP mérite d'être complétée.

Condition nécessaire et suffisante pour la décidabilité du raisonnement logique. Dans le cas où un support $\mathscr{S}$ ne contient aucune définition de TUP, nous avons déterminé une condition nécessaire et suffisante pour que tout GU G non absurde y présente une extension finie : le graphe $G_{\mathscr{T}}$ doit être acyclique (§9.3.2.2). Nous souhaitons étendre ce résultat au cas où $\mathscr{S}$ contient des définitions de TUP. Nous avons proposé une première condition suffisante basée sur la conjonction de l'acyclicité du graphe $G_{\mathscr{T}}$ du support, et de l'acyclicité de l'ensemble des définitions de TUP (§9.3.2.3). Cependant, nous avons également montré deux exemples simples qui préservent la finitude des extensions de GU non absurdes, malgré la violation de chacune de ces contraintes.

Bien que nous ayons justifié en quoi la condition suffisante proposée semble acceptable pour les lexicographes de la TST, une direction possible de recherche est d'approfondir notre étude pour déterminer des conditions suffisantes moins restrictives. Nous connaissons déjà cette condition dans le cas où le support ne possède pas de définition de TUP : l'acyclicité du graphe $G_{\mathscr{T}}$.

Il s'agira enfin, en présence d'un support de GU ne remplissant pas la condition suffisante de décidabilité définie, de déterminer la plus petite correction à effectuer pour respecter cette condition.

Étude de la complexité du raisonnement logique Ensuite, lorsque le problème de déduction est décidable, il serait intéressant d'en déterminer plus précisément la complexité spatiale et temporelle. A l'instar du formalisme des Graphes Conceptuels, et au contraire du formalisme des Logiques de Description, le formalisme des GU propose donc un ensemble d'axiomes et constructeurs élémentaires, et repose sur des conditions globales pour assurer une limite de la complexité des problèmes de décision.

Étude du caractère absurde des GU Nous nous sommes concentrés sur le problème de déduction pour le formalisme des GU, mais nous avons également mentionné le problème de déterminer si un GU est absurde ou non. La consistance et l'inconsistance d'un GU est un problème de décision important, et nous avons souligné deux directions de travail à court terme pour l'étude de ce problème :

- malgré le fait qu'un GU G absurde puisse avoir une extension infinie, nous conjecturons que déterminer si G est absurde par déduction (cf., déf. 106) est décidable. La complexité de cette décision devrait également être précisée en fonction du support.
- l'ensemble des GU absurdes est défini comme étant l'ensemble des GU qui ne sont satisfaits par aucun modèle. Nous conjecturons qu'un GU est absurde (cf., déf. 99) si et seulement s'il est absurde par déduction (cf., déf. 106).

Il s'agira enfin, en présence d'un GU absurde, de déterminer la plus petite correction à effectuer pour lever l'absurdité.

11.1.2 Représentation des composantes optionnelles et interdites

Comme nous l'avons précisé dans la section 8.3.3, seules les définitions lexicographiques formelles dont tous les nœuds de participant sont obligatoires sont actuellement représentées dans le formalisme des GU. Cette section présente des travaux futurs qui impliquent la révision de la notion de règle de GU pour représenter les définitions lexicographiques formelles ayant :

- des composantes optionnelles (§11.1.2.1) ;
- des composantes interdites (§11.1.2.2).

11.1.2.1 Composantes optionnelles

La prise en compte des composantes optionnelles dans les définitions lexicographiques formelles permet d'entrevoir des problématiques d'optimisation des bases de règles.

Considérons la définition lexicographique formelle d'un type d'unité *B* illustrée par la figure 11.1a. Les nœuds de participant correspondant aux PosA s_1 et s_3 sont optionnels. Si l'on étend la transformation proposée dans la section 8.3.3, alors les quatre définitions de TUP, illustrées sur la figure 11.1b, sont générées.

Considérons maintenant le GU *g* représenté sur la figure 11.1c. Chacune des quatre règles d'expansion y est applicable. Il est possible de clore *G* par application successive des règles d'expansion de **D1**, **D2** puis **D3**. Cependant, le résultat est le même si l'on applique la seule règle d'expansion de **D4**. A l'inverse, si la règle d'expansion de **D1** n'est pas applicable, alors il est inutile de vérifier l'applicabilité des règles d'expansion de **D2** et **D3**.

Un point de départ pour optimiser l'extension des GU est de remarquer que les règles peuvent s'organiser selon une relation du type :

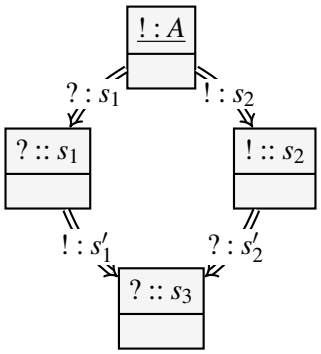
si la règle R_1 n'est pas applicable, alors la règle R_2 n'est pas applicable.

La figure 11.1b représente cette relation pour les règles d'expansion des définitions de TUP. Nous entrevoyons ainsi le lien avec les algorithmes d'optimisation des bases de règles, tels que l'algorithme de Rete (Forgy, 1982) par exemple.

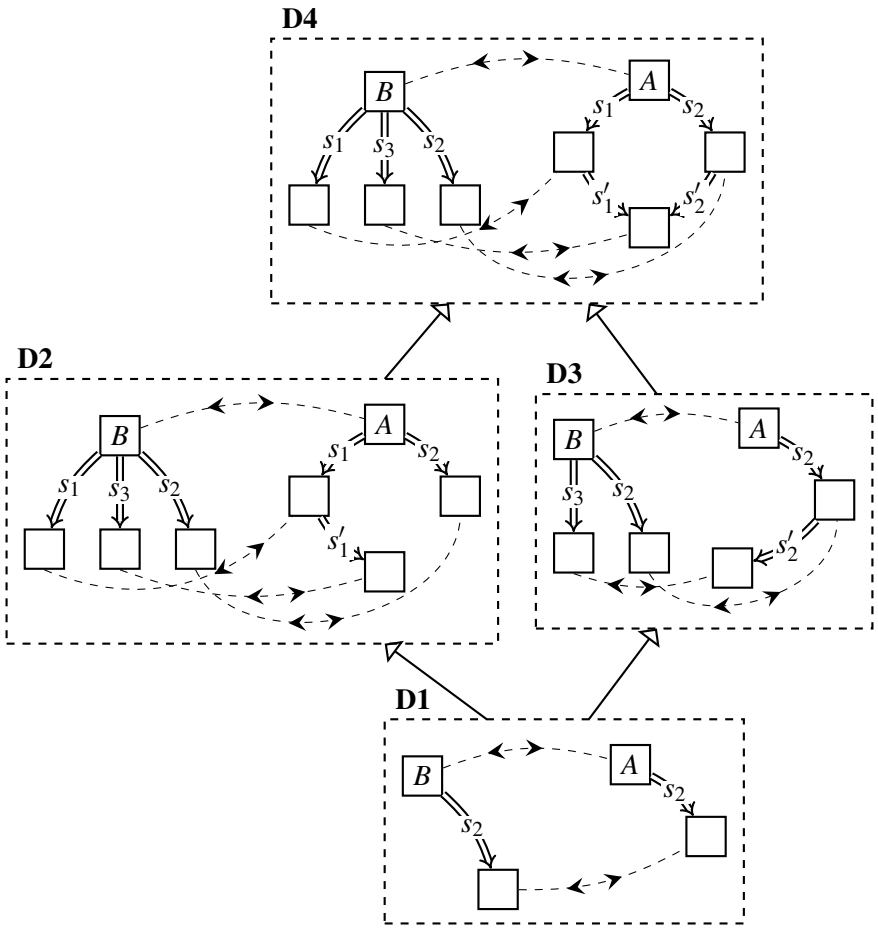
Révision de la définition des règles. Au niveau du formalisme des GU, nous envisageons donc de compléter la définition des règles pour y inclure un ensemble de composantes optionnelles.

Impact sur l'opérationnalisation. L'amélioration de la représentation des bases de règles et des définitions de TUP ouvre des perspectives de recherche pour l'étude du raisonnement dans le formalisme des GU. En effet, une précision de la représentation des règles peut aider à la détermination d'une contrainte nécessaire et suffisante pour que tout GU non absurde possède une extension finie.

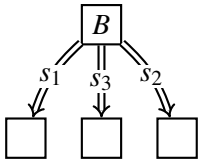
En ce qui concerne l'opérationnalisation à l'aide des formalismes du Web sémantique, nous étudierons la possibilité d'utiliser le mot-clé OPTIONAL du langage de requête SPARQL. Pour les moteurs de règles, factoriser ainsi les règles représente une optimisation importante.



(a) Définition lexicographique formelle avec plusieurs composantes optionnelles



(b) Treillis des définitions de TUP générées.



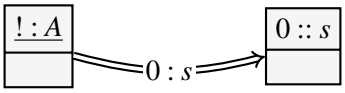
(c) GU sur lequel chaque règle d'expansion est applicable.

FIGURE 11.1 – Limites actuelles de la représentation des composantes optionnelles.

11.1.2.2 Composantes interdites

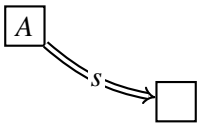
Prendre en compte des composantes interdites dans les définitions lexicographiques formelles nécessite de réviser la notion de règles de GU.

Considérons la définition lexicographique formelle illustrée par la figure 11.2a. Le nœud de participant correspondant à la PosA s est interdit. La figure 11.2b représente la règle de contraction du TUP générée selon la section 8.3.3. Considérons le GU G illustré sur la figure 11.2c représente un graphe d'unité. La règle D_t^- est applicable à G, alors que G possède justement un nœud d'unité qui est interdit.



(a) Définition lexicographique formelle avec une composante interdite

(b) Règle de contraction générée



(c) Graphe sur lequel la règle de contraction est applicable, à tord.

FIGURE 11.2 – Limites actuelles de la représentation des composantes interdites.

Révision de la définition des règles. Au niveau du formalisme des GU, nous envisageons donc de compléter la définition des règles pour y inclure un ensemble de composantes interdites.

Impact sur l'opérationnalisation. L'amélioration de la représentation des bases de règles et des définitions de TUP impacte de manière plus importante l'étude du raisonnement dans le formalisme du GU.

Suivant que le type de conclusion proposé se base sur de l'attachement procédural ou non, la question de la préservation de la monotonicité du formalisme pourrait alors être soulevée. Il s'agirait alors de déterminer les conditions nécessaires et suffisantes sur l'ensemble des définitions de TUP pour que le raisonnement dans la formalisme des GU reste monotone.

En ce qui concerne l'opérationnalisation à l'aide des formalismes du Web sémantique, nous étudierons la possibilité d'utiliser le mot-clé NOT EXISTS du langage de requête SPARQL. Pour les moteurs de règles, une telle garde peut également accélérer l'évaluation des requêtes.

11.1.3 Conceptualisation complète des définitions

Nous avons fait le choix d'étudier une conceptualisation partielle des définitions lexicographiques de la TST.

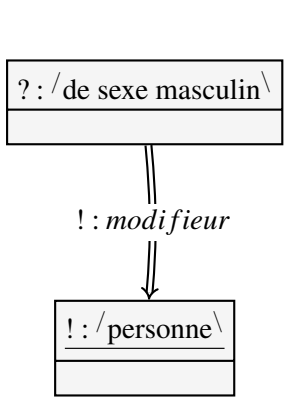
Dans cette section nous revenons sur la structure complète des définitions lexicographiques détaillée dans la section 1.3.2.1, et précisons les travaux futurs qui concernent les composantes faibles (§11.1.3.1), et la partie présuppositionnelle (§11.1.3.2). Enfin, nous introduirons un travail futur qui concerne l'étude de la polysémie lexicale dans les définitions lexicographiques (§11.1.3.3).

11.1.3.1 Composantes faibles

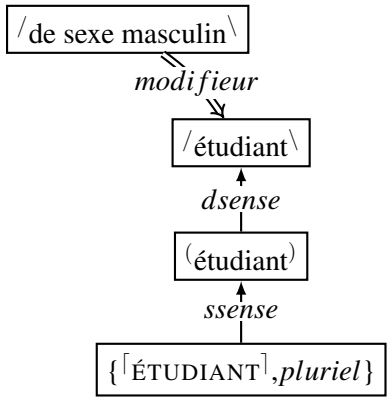
Nous avons étudié la conceptualisation des PosA optionnelles des prédicats linguistiques, et avons étendu cette conceptualisation à la conceptualisation des participants optionnels dans les définitions lexicographiques formelles de types de lexie.

Une composante faible pouvant être pensée ou non dans la situation linguistique SIT($\ulcorner$L$\urcorner$), il s'agit donc d'une composante optionnelle par nature. Cependant, contrairement aux composantes optionnelles que nous avons étudiées, le blocage des composantes faibles peut dépendre du contexte à un autre niveau de représentation linguistique.

Considérons l'exemple du type de lexie $\ulcorner$ÉTUDIANT$\urcorner$ que l'on a détaillé dans la section 1.3.2.1. La figure 11.3a est une définition lexicographique formelle simplifiée de $\ulcorner$ÉTUDIANT$\urcorner$. Nous cherchons à représenter le fait que /de sexe masculin\ est une composante faible, et qu'elle est bloquée si la lexie $\ulcorner$ÉTUDIANT$\urcorner$ est employée au pluriel. Une possibilité est de définir une contrainte négative : il s'agit d'un GU qui ne doit être projetable par homomorphisme sur aucune représentation linguistique. La figure 11.3b illustre une telle règle de contrainte négative.



(a) Définition lexicographique formelle simplifiée de $\ulcorner$ÉTUDIANT$\urcorner$, avec une composante optionnelle



(b) GU de contrainte négative : il ne doit pas exister d'homomorphisme entre ce GU et une représentation linguistique.

FIGURE 11.3 – Représentation possible des composantes faibles avec le formalisme des Graphes d'Unités

11.1.3.2 Partie présuppositionnelle

La partie présuppositionnelle des définitions lexicographiques spécifie ce qui reste vrai lorsque la lexie est employée avec une négation (cf., §1.3.2.1). Par exemple, si X n'aide pas Y à faire Z, alors il reste vrai que Y fait ou essaie de faire Z. Ainsi, la représentation des présuppositions

implique la représentation de la négation dans la TST, et des mécanismes de résolution de cette négation dans les représentations.

Il convient donc de mener une étude en tandem avec les linguistes de la TST :

1. étudier la négation linguistique sous ses différentes formes ;
2. proposer une conceptualisation adaptée au niveau sémantique profond ;
3. réviser le formalisme des graphes d'unités pour représenter la négation.

Conceptualisation de la négation au niveau sémantique profond. Dans les représentations sémantiques de surface, la partie présuppositionnelle est parfois représentée sous la forme d'une paraphrase à l'affirmative (cf., figure 1.2), ou à l'aide d'un nœud typé ⟨not⟩ qui modifie un autre nœud (cf., figure 8.1). Au niveau sémantique profond, nous aurons le choix de réutiliser cette dernière représentation, ou d'envisager une représentation alternative. Nous pourrons en particulier considérer de mettre un sous-graphe d'unité avec une négation, comme cela est fait dans le fragment des Graphes Conceptuels qui considère la négation.

Impact sur l'étude du raisonnement. Nous devrons ensuite adapter la sémantique formelle du formalisme des GU, proposer d'enrichir la base de règles axiomatiques d'inférence, et étudier l'impact de ces révisions sur l'étude du raisonnement dans le formalisme des GU.

Modélisation avec les formalismes du Web sémantique. Finalement, précisons que la recommandation du W3C RDF 1.1 comprend la notion de graphes nommés, qui permet d'associer à tout graphe RDF une URI, et d'utiliser cette ressource dans un graphe RDF. Nous pourrons donc utiliser ce dernier standard pour la représentation des négations, et des composantes présuppositionnelles dans les définitions lexicographiques.

11.1.3.3 Polysémie lexicale et représentation de la disjonction

Nous n'avons pas étudié la représentation de la disjonction dans les définitions lexicographiques ni dans les représentations linguistiques. Le formalisme des GU permet déjà de représenter certaines formes de disjonction :

Si un type de lexie est polysémique, il est possible (mais pas souhaitable) de lui associer deux définitions lexicographiques formelles, ou de le représenter avec deux TUP.

Les relations qui lient des nœuds d'unité de deux représentations linguistiques adjacentes sont considérées comme des relations circonstancielles. Ainsi, si un GU G contient deux triplets circonstanciels $(u, dsense, v_1)$ et $(u, dsense, v_2)$, alors le nœud d'unité u est ambigu, car il est lié à deux sens différents.

Enfin, dans les représentations sémantiques de surface de la TST, la disjonction est parfois simplement représentée à l'aide d'un nœud typé ⟨or⟩ qui lie deux nœuds.

Il convient donc de mener une étude en tandem avec les linguistes de la TST pour étudier la disjonction linguistique sous ses différentes formes (exclusive ou inclusive, polysémie, ambiguïté, vague), et en proposer une conceptualisation et une représentation dans le formalisme des GU.

Enfin, notons que disjonction, conjonction et négations sont intimement liées par les lois de l'algèbre de Boole. Il conviendra d'étudier leurs interactions.

11.2 Extension de la couverture de la TST

Cette section présente les perspectives qui concernent quatre aspects du modèle Sens-Texte que nous avons choisis de placer au dehors de notre champ d'étude :

- la représentation des niveaux linguistiques surfaciques (plus proches des textes, §11.2.1) ;
- la représentation des Fonctions Lexicales (§11.2.2) ;
- la représentation des structures communicationnelles (§11.2.3) ;
- la représentation des modules de correspondance entre niveaux linguistiques (§11.2.4).

11.2.1 Représentation des niveaux plus surfaciques

Nous avons orienté notre étude sur l'étude des niveaux linguistiques les plus proches des sens. Entre autre, nous n'avons étudié que la notion de structure actancielle sémantique dans la théorie des actants.

D'un autre côté, nous avons préparé le terrain pour l'étude des niveaux linguistiques plus proches des textes, par exemple :

- en considérant qu'à tout type d'unité peut être associé une structure actancielle.
- en introduisant une hiérarchie des SRelC, principalement utilisés dans les niveaux les plus surfaciques.

Une direction de recherche importante est donc de poursuivre notre étude pour les autres niveaux linguistiques : syntaxiques, morphologiques, et phonologiques. Nous détaillons ci-dessous seulement deux directions de travail particulièrement importantes pour la TST.

Conceptualisation des structures actancielles syntaxiques. Il s'agit d'approfondir l'étude de la conceptualisation des types d'unité lexicale, en prenant en compte le second volet de la théorie des actants : celui qui concerne la structure actancielle syntaxique des types d'unité lexicale (Mel'čuk, 2004b). Deux types d'actants syntaxiques sont pensés : les actants syntaxiques profonds, et les actants syntaxiques de surface. Une question importante à laquelle cette étude devra donc répondre est la suivante :

Si l'on associe aux types d'unité lexicale une structure actancielle composée des actants syntaxiques profonds et des actants syntaxiques de surface, alors est-il possible de les organiser en une hiérarchie au sein de laquelle la structure actancielle est héritée et potentiellement spécialisée ?

L'héritage et la spécialisation des structures actancielles est au cœur de la construction du formalisme des GU. Dans l'éventualité d'une réponse négative à cette question de recherche, nous devrions introduire deux nouvelles catégories de types d'unité linguistique pour porter les deux structures actancielles syntaxiques :

- des types d'unité syntaxiques profonds, dont la structure actancielle correspond à la structure actancielle syntaxique profonde du type de lexie associé, et qui sont instanciés dans les représentations syntaxiques profondes ;
- des types d'unité syntaxiques de surface, dont la structure actancielle correspond à la structure actancielle syntaxique de surface du type de lexie associé, et qui sont instanciés dans les représentations syntaxiques de surface.

Représentation des autres zones du DEC. Les zones phonologiques, morphologiques, et syntaxiques du DEC nécessitent plus de travail concernant la représentation des niveaux linguistiques homonymes. Dans la zone syntaxique, le tableau de régime décrit les correspondances possibles, impossibles, ou contraintes, entre les actants syntaxiques profonds et les actants syntaxiques de surface. Notons donc simplement qu'à l'instar de la modélisation des définitions lexicographiques à l'aide des formalisme du Web Sémantique, nous pourrions :

- proposer une modélisation simple des tableaux de régime ;
- écrire une base de règles SPARQL TEMPLATE pour générer automatiquement une base de règles de transformation élémentaires pour passer d'une représentation syntaxique profonde à une représentation syntaxique de surface.

11.2.2 Vers la représentation des Fonctions Lexicales

Nous avons présenté les Fonctions Lexicales de la TST dans la section 1.1.1.3, comme l'une des conceptualisations les plus complètes des différentes relations lexicales.

Pour simplifier, une fonction lexicale F définit deux modifications génériques des types de lexie auxquelles elle s'applique :

- une modification du sens : la *dérivation sémantique* ;
- une modification de la compositionnalité dans les phrases.

Une des conceptualisations les plus poussées des Fonctions Lexicales est proposée par Kahane et Polguère (2001), et utilisée dans le projet RELIEF (Lux-Pogodalla et Polguère, 2011). Par ailleurs, Coyne et Rambow (2009) discutent des similarités entre la TST et le réseau FrameNet, et proposent une extension de la conceptualisation des Fonctions Lexicales pour décrire les relations entre "frames lexicales".

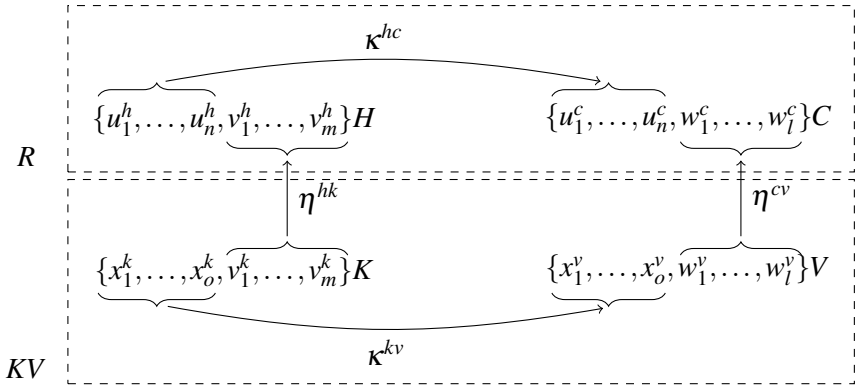
Dans la conceptualisation étendue que nous avons proposée, le sens d'un type de lexie est défini par sa définition lexicographique formelle, au niveau sémantique profond. La dérivation sémantique d'une fonction lexicale F correspondrait donc à une modification générique appliquée aux définitions lexicographiques formelles des types de lexie auxquelles F s'applique.

Des règles génériques pour le formalisme des GU. Dans un travail en cours, nous révisons la définition des règles de GU pour les rendre génériques. Dans une règle générique $R = \langle H, C, \kappa^{hc} \rangle$, κ^{hc} n'est plus une bijection entre les nœuds distingués de H et ceux de C, mais seulement une fonction partielle injective. La règle générique R peut alors être instanciée pour un objet *clé-valeur*, noté $KV = (K, V, \kappa^{kv}, \eta^{kh}, \eta^{vc})$, pour former une règle $R(KV) = \langle R(K), C(V), \kappa \rangle$. Le processus d'instanciation est le suivant :

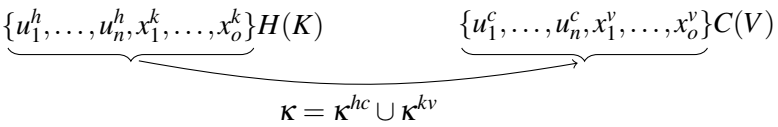
- l'hypothèse $H(K)$ de $R(KV)$ est obtenue en fusionnant l'hypothèse générique H avec la clé K selon η^{kh} ;
- la conclusion $C(V)$ de $R(KV)$ est obtenue en fusionnant la conclusion générique C avec la valeur V selon η^{vc}.
- $\kappa = \kappa^{hc} \cup \kappa^{kv}$ est une bijection des nœuds distingués de l'hypothèse $H(K)$ vers la conclusion $C(V)$.

La figure 11.4a illustre une règle générique R, et un objet *clé-valeur* associé KV. La figure 11.4b représente alors la règle $R(KV)$ obtenue par instanciation de R pour KV.

Il convient donc de définir formellement ce nouveau type de règles dans le formalisme des GU, et d'étudier son opérationnalisation.



(a) Une règle générique (H, C, κ_{hc}) et un objet clé-valeur $(K, V, \kappa_{hk}, \kappa_{cv}, \kappa_{kv})$.



(b) Règle obtenue par instanciation de la règle générique avec l'objet clé-valeur.

FIGURE 11.4 – Illustration des règles génériques et objets clé-valeur.

Représentation des dérivations sémantiques et des règles de correspondance analogues. Nous pourrons alors utiliser cette généralisation des règles de différentes manières :

- Une définition de TUP correspond à deux règles. Nous pouvons donc définir la notion de définition générique de TUP à partir des notions de définition de TUP et de règle générique. Ce qui devrait permettre de représenter la dérivation sémantique.
- Les règles de correspondance entre niveaux de représentation linguistique peuvent être rendues génériques également.

Ce sont deux points de départ pour une étude du raisonnement par analogie.

Représentation de la combinatoire lexicale restreinte. Pour conclure, il semble pertinent d'étudier dans quelle mesure la conjonction d'une définition générique de TUP et de règles de correspondances génériques peut permettre de représenter les Fonctions Lexicales. Nous en induisons les questions de recherche ultérieures suivantes :

- comment la cinquantaine de Fonctions Lexicales standard de la TST peut-elle être représentée avec le formalisme des GU ? Cette étude peut s'accompagner d'un besoin d'étendre la conceptualisation des Fonctions Lexicales et/ou d'affiner leur classification.
- pouvons-nous définir de nouvelles Fonctions Lexicales non-standard de manière satisfaisante ?
- les combinaisons et compositions de fonctions lexicales (Mel'čuk, 1996; Kahane et Polguère, 2001), peuvent-elles être formalisées de manière satisfaisante ?

11.2.3 Représentation de la structure communicationnelle

La structure communicationnelle est une composante extrêmement importante des représentations linguistiques dans la TST. Elle permet en particulier de réduire le potentiel paraphrastique des représentations linguistiques, et facilite le choix d'une linéarisation des représentations sémantiques.

Dans le cadre de notre étude concernant la TST, les travaux de Mel'čuk (2001) semblent un point de départ pertinent pour conceptualiser, représenter, et opérationnaliser les structures communicationnelles.

Des actes du langage et de la pragmatique. La structure communicationnelle des représentations sémantiques aide à la détermination de l'acte du langage engagé, ainsi que du contexte de l'énoncé.

Notons que les types de lexie choisis dans l'énoncé peuvent également contribuer à la détermination du contexte, justement via la partie présuppositionnelle de leurs définitions lexicographiques.

Implications possibles pour le formalisme des Graphes d'Unités. Notons que pour représenter certains aspects de la structure sémantique communicative tels que l'opposition Thème/Rhème, il est possible que l'on nécessite un moyen d'annoter des GU, ou la relation entre des GU. Nous pourrons alors :

- nous inspirer du fragment des Graphes Conceptuels *emboîtés* pour l'étude du raisonnement logique ;
- utiliser les graphes nommés de la recommandation RDF 1.1 pour modéliser ces connaissances sur le Web des données.

Utilité pour l'interaction en langue naturelle. Dans le cadre du développement d'une application d'interaction homme-machine en langue naturelle, il serait fondamental de mobiliser les trois domaines :

- la théorie des actes du langage
- la pragmatique linguistique ;
- la théorie de la révision des croyances (Belief revision).

Il s'agirait alors de distinguer les différentes opérations souhaitées par l'utilisateur. Par exemple [1] :

- assertif : dans ce cas, s'agit-il de réviser la base de connaissances (le monde n'a pas changé) ou de la mettre à jour (le monde a changé) ?
- directif, ordre : il s'agit-t-il d'agir sur la base de connaissances, ou d'effectuer une action sur le monde ?
- directif, demande : il s'agit d'effectuer une requête sur la base de connaissances et de renvoyer le résultat de la requête.

1. Nous utilisons la classification originelle des classes d'acte illocutoires d'Austin

11.2.4 Vers la représentation des modules de correspondance du modèle sens-texte

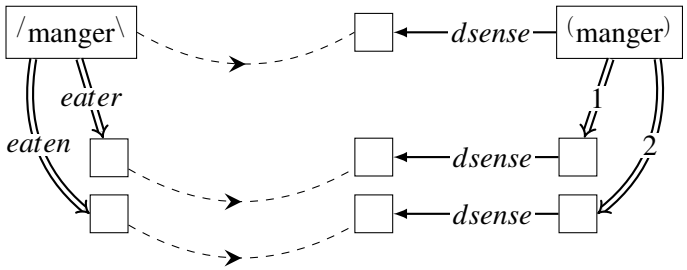
Nous avons placé en dehors de notre champ d'étude la représentation des règles de grammaire et des règles lexicales, qui composent les modules de correspondance entre niveaux de représentation linguistique. Nous avons cependant présenté une application directe des règles pour la représentation de la correspondance entre les structures actancielles du TUSemP /to eat\, et du TUSemS (to eat) (cf., figure 8.6).

Un ensemble de telles règles permet donc de composer un module de correspondance, et permet de décrire les réécritures possibles d'un graphe d'unités d'un niveau de représentation (ex. niveau sémantique profond) en un graphe d'unités d'un niveau adjacent (ex. niveau sémantique de surface).

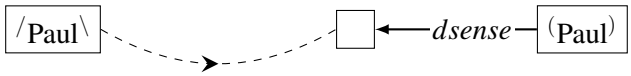
Cette section démontre tout d'abord que les règles de correspondance doivent être unifiées pour définir une transformation complexe (§11.2.4.1). Elle précise ensuite les problématiques soulevées par la représentation d'un module de transformation (§11.2.4.2), puis par les transformations de graphes d'unités en cascalde (§11.2.4.3)

11.2.4.1 Unification des règles de correspondance

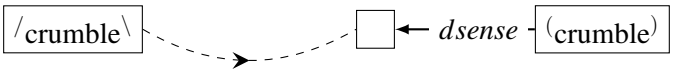
Les règles de correspondance doivent être unifiées pour définir une transformation d'un GU complexe. Considérons donc les trois règles de correspondance illustrées sur la figure 11.5.



(a) Règle de réécriture de /manger\ à (manger).



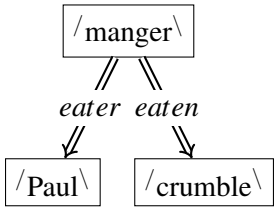
(b) Règle de réécriture de /Paul\ à (Paul).



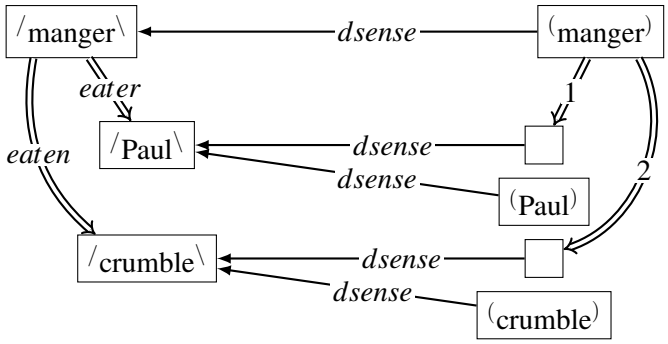
(c) Règle de réécriture de /crumble\ à (crumble).

FIGURE 11.5 – Exemple de trois règles de réécriture de GU.

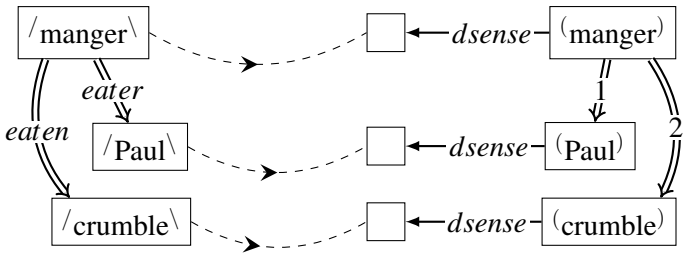
Considérons la représentation sémantique profonde illustrée par un GU *G* sur la figure 11.6a. Si l'on applique chacune des trois règles de correspondance indépendamment à *G*, alors on obtient le GU illustré sur la figure 11.6b, ce qui n'est pas le résultat attendu. Pour réécrire complètement le GU *G* au niveau sémantique de surface, il faut composer (ou unifier) ces règles de correspondance en une seule règle complexe (cf., fig. 11.6c). Le résultat de l'application de cette règle au GU *G* est illustré sur la figure 11.6d.



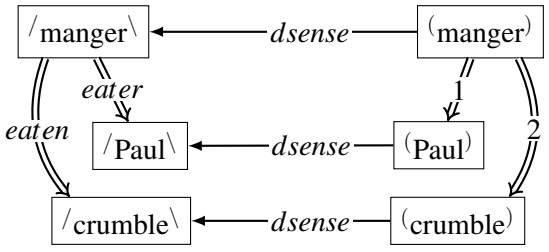
(a) Exemple de graphe d'unité.



(b) Résultat de l'application successive des règles de réécriture.



(c) Unification des trois règles de réécriture.



(d) Résultat de l'application de la règle unifiée.

FIGURE 11.6 – Application successive des règles de réécriture, ou application d'une règle unifiée.

11.2.4.2 Etude des transducteurs de graphes d'unités

Transducteurs de Graphes d'Unités. Les règles de correspondances représentent les briques de base des transducteurs de GU. Une perspective de recherche importante concerne la définition, l'étude, et les applications de ces transducteurs. Un point de départ à cette étude peut être l'étude des grammaires, transducteurs, et cascades de transducteurs de graphes existants pour le traitement automatique de la langue. Nous recensons deux approches inspirées de la TST :

- Les Grammaires d'Unification Polarisées (cf., Kahane, 2004), des grammaires génératives dans lesquelles chaque entrée du dictionnaire correspond à une construction grammaticale possible ;
- Le système MATE (Bohnet et Wanner, 2010), propose un prototype de transducteurs de graphes en cascade pour l'annotation riche de textes.

Il semble que ces travaux n'exploitent pas toute la finesse descriptive que la Théorie Sens-Texte pourrait apporter, et que l'étude des systèmes à base de règles et de leur complexité n'a pas été abordée.

Considérons un ensemble de règles, et un graphe G à transformer. Le problème de base est de sélectionner un ensemble de règles et une combinaison de ces règles telle qu'il existe un isomorphisme entre la combinaison des prémisses et G. La combinaison des conclusions des règles est alors la réécriture du graphe. La question suivante se pose donc : Comment sélectionner un ensemble de règles à appliquer pour réécrire un graphe ? Cette question se décompose en plusieurs sous-questions :

1. que faire si plusieurs ensembles de règles peuvent être appliqués de manière concurrente, quel ensemble de règles privilégier ?
2. que faire si aucun ensemble de règles ne convient ?

Aspect stochastique des transducteurs. Au delà de la question des algorithmes pour identifier les règles à appliquer, les combiner et appliquer la combinaison, il est fortement possible que plusieurs ensembles de règles soient applicables. Considérons par exemple quatre règles $A1$, $A2$, $B1$ et $B2$. $A2$ étant plus spécifique que $A1$, et $B2$ étant plus spécifique que $B1$. Supposons que des combinaisons $\{A1, B2\}$ et $\{A2, B1\}$ soient applicables pour transformer un graphe G. Il est impossible de décider quel ensemble de règles appliquer si l'on se contente d'une relation d'ordre entre les règles. Nous devons donc pondérer les règles, et calculer le poids d'une combinaison de règles en fonction du poids des règles. Nous devrons donc nous orienter vers l'étude des transducteurs stochastiques de Graphes.

Une piste pour pondérer les règles est de définir une mesure de distance d'édition entre leurs prémisses. Nous pouvons ici nous inspirer des transformations élémentaires des Graphes Conceptuels, et les adapter aux Graphes d'Unités. La distance entre deux règles correspondrait grossièrement au nombre de transformations élémentaires nécessaires pour passer de la prémisse de l'une à la prémisse de l'autre.

Robustesse des transformations : utilisation de la synonymie. Une seconde question se présente dans le cas où aucune combinaison d'aucun ensemble de règles ne convient. Cela peut être le cas lors de la traduction automatique, si dans la langue source un mot, une combinaison syntaxique ou une combinaison sémantique, n'a pas d'équivalent dans la langue cible. Il nous faut alors définir une méthode permettant de transformer le graphe dans une version équivalente sur laquelle une combinaison d'un ensemble de règles peut alors convenir. Là encore, nous pourrons nous inspirer des transformations élémentaires des Graphes Conceptuels.

Robustesse des transformations : extension à la quasi-synonymie. Finalement, il est fondamental pour un certain nombre d'applications que tout graphe admette une transformation, ce qui justifierait de définir un ensemble de conditions sur le lexique nécessaire pour qu'il existe toujours une combinaison d'un ensemble de règles qui convienne. Cependant, une alternative est de permettre de transformer le graphe en entrée en une version non pas strictement équivalente, mais légèrement détériorée, de sorte qu'il existe une combinaison d'un ensemble de règles qui convienne.

11.2.4.3 Étude de cascades de transducteurs de GU

A plus long terme, nous pourrons étudier la cascade de tels transducteurs, et envisager les applications suivantes.

Application pour le traitement automatique de la langue. Parmi les applications possibles, notons l'analyse et la génération de textes en utilisant la richesse descriptive de la Théorie Sens-Texte, la traduction automatique, l'annotation riche de texte, l'extraction de connaissances et l'interrogation en langue naturelle de corpus tel que Wikipedia. Pointons ici le travail de Friburger et Maurel (2004) sur l'utilisation de transducteurs de graphes pour la détection d'entités nommées par exemple.

Approximation d'une cascade de transducteurs. Nous savons que la génération et l'analyse d'énoncés à l'aide de règles est coûteux en temps et en place. Une direction de recherche possible consiste à étudier l'approximation d'une cascade de transducteurs stochastiques de graphes d'unités par un unique transducteur stochastique de chaînes de caractères.

Apprentissage automatique et cercle vertueux. De nombreux travaux concernent l'apprentissage automatique de transducteurs. Les transducteurs devant être générés automatiquement à partir du DEC, nous soulevons ici une question de recherche importante :

Comment peut-on peupler automatiquement un DEC via l'apprentissage automatisé d'une cascade de transducteurs ?

Étant donné que d'un meilleur DEC on générerait une meilleure cascade de transducteurs, une réponse à cette question induirait automatiquement la possibilité de définir un cercle vertueux pour l'apprentissage du DEC.

11.3 Applications directes

Nous présentons dans cette section quatre applications directes de nos travaux :
– l'implémentation du formalisme des GU (§11.3.1) ;
– la population manuelle du formalisme des GU (§11.3.2) ;
– l'intégration avec d'autres ressources lexicales existantes (§11.3.3) ;
– une étude en lexicologie multilingue (§11.3.4).

11.3.1 Implémentations

Grâce à la modélisation que nous avons proposée avec les formalismes du Web Sémantique, nous pouvons dès à présent proposer une implémentation du formalisme des GU sur le Web des données. Nous avons noté en conclusion du chapitre 10 que le logiciel Corese (Corby *et al.*, 2004) développé dans l'équipe Wimmics est parfaitement adapté pour implémenter le formalisme des Graphes d'Unités. Nous travaillons actuellement sur la validation de notre schéma et de nos bases de règles à l'aide d'un ensemble de tests. Nous envisageons à court terme de proposer une implémentation Java simplifiée pour travailler avec le formalisme des GU directement au dessus de Corese.

Ensuite, bien que nous nous soyons basés sur le métamodèle OWL 2 RL, le formalisme des Graphes d'Unités est fondamentalement différent des logiques de description. Nous envisageons donc de développer une API plus adaptée pour le formalisme des GU (nous nous inspirerons néanmoins de OWLAPI[2]). Quand à l'implémentation, nous devrions passer progressivement d'une implémentation entièrement basée sur des règles et le moteur Corese, à une implémentation spécifiquement optimisée.

Par ailleurs, notons que nos bases de règles représentent un cas d'étude intéressant pour d'autres travaux dans l'équipe Wimmics, comme l'interrogation des bases de règles ou l'exécution de règles dans un contexte distribué (Seye *et al.*, 2014). Enfin, la simplicité de notre solution pour générer une base de règles à l'aide d'une base de règles devrait contribuer à motiver l'intégration des règles SPARQL TEMPLATE dans la prochaine recommandation SPARQL.

11.3.2 Population manuelle du formalisme des GU

Nous envisageons deux perspectives pour la population manuelle du formalisme des GU : pour la lexicographie explicative et combinatoire, ou pour un autre domaine de spécialité.

En lexicographie explicative et combinatoire. Le premier public envisagé pour l'application du formalisme des GU est celui des lexicographes de la TST. En complément des perspectives présentées en conclusion du chapitre 5, l'implémentation du formalisme des GU permet d'envisager une intégration complète de nos travaux dans un éditeur de DEC tel que l'éditeur DiCo.

Les scénarios de la section 2.1 peuvent alors être envisagés. Par exemple, la requête SPARQL suivante répond au scénario n°1 : *Quelles lexies sont des prédicats sémantiques qui impliquent un actant du type* ⟨*humain*⟩*, et un actant du type* ⟨*objet*⟩ *?*

```
1 SELECT ?t
2 WHERE {
```

2. OWLAPI - Java API for OWL - http://owlapi.sourceforge.net/

```
3   ?lut mtt:surfaceSense ?semut .
4   ?semut mtt:deepSense ?dsemut .
5   ?dsemut ug:actantSlot ?a1 .
6   ?dsemut ug:specializationOf [ a ug:Signature ;
7     ug:onActantSlot ?a1 ;
8     ug:allUnitsFrom ecd-fr:dsem-humain ] .
9   ?dsemut ug:actantSlot ?a2 .
10  ?dsemut ug:specializationOf [ a ug:Signature ;
11    ug:onActantSlot ?a2 ;
12    ug:allUnitsFrom ecd-fr:dsem-objet ] .
13  }
```

Pour d'autres domaines de spécialités. Finalement, le formalisme des GU vient enrichir la famille des formalismes de représentation des connaissances. Il est envisageable qu'une prochaine étude d'ingénierie des connaissances appliquée à un autre domaine de spécialité considère ce formalisme comme étant le plus adapté pour en représenter les connaissances. Un éditeur de connaissances personnalisé pour ce domaine de spécialité pourraient alors être développés.

Enfin, pour des raisons de généricité, un éditeur générique conçu pour les ingénieurs des connaissances pourrait être développé.

11.3.3 Les Graphes d'Unités et autres ressources lexicales

Cette section soulève la question de l'intégration de ressources lexicales existantes dans le formalisme des GU.

FrameNet. Bien que basée sur une théorie linguistique différente, la ressource lexicale FrameNet possède certaines similarités avec la TST, et des travaux ont étudié l'apport potentiel de chacune de ces théories pour l'autre (Coyne et Rambow, 2009; Bouveret et Fillmore, 2008). Dans un premier temps, il serait intéressant d'affiner ces liens à la lumière de nos travaux. Ensuite, FrameNet semble un excellent candidat pour la construction automatisée d'une ébauche de DEC avec le formalisme des GU.

Réseaux lexicaux. Les réseaux lexicaux construits dans le cadre des projets Papillon et JeuxDeMots (Mangeot *et al.*, 2003; Lafourcade, 2007) sont tissés de liens inspirés des fonctions lexicales. Ils forment donc également de bons candidats pour construire une ébauche de DEC avec le formalisme des GU.

Nous savons que les lexicographes de la TST (du projet DiCo tout du moins) semblent préférer construire le DEC de manière centralisée, supervisée, et avec une maîtrise importante de la qualité. Cependant, il serait intéressant d'étudier dans quelle mesure le mode de construction contributive par le biais de jeux sérieux des projets Papillon et JeuxDeMots pourrait accélérer l'élaboration d'un DEC. La capacité de raisonner et d'exprimer des contraintes dans le formalisme des GU pourrait alors jouer un rôle important dans la maîtrise de la cohérence et de la qualité du DEC ainsi construit.

Ressources lexicales du Web des Données. Enfin, l'alignement que nous avons proposé avec le modèle **ontolex** ouvre la voie vers l'intégration d'autres ressources lexicales du Nuage des Données Linguistiques Liées. Citons notamment Princeton WordNet RDF [3] récemment publié, ainsi que UBY ou encore VerbNet.

3. Princeton WordNet RDF, version RDF de WordNet 3.1, http://wordnet-rdf.princeton.edu/

11.3.4 Lexicologie multilingue

Nous savons que le niveau sémantique de surface dépend de la langue, puisque la détermination de la structure actancielle sémantique de surface d'un type de lexie ⌈L⌋ repose sur des critères linguistiques portant sur l'utilisation de ⌈L⌋ dans la langue.

Le niveau sémantique profond, conçu pour représenter les *sens*, pourrait quant à lui être considéré comme un niveau de représentation interlingue. Il serait donc intéressant de mener une étude en lexicologie multilingue, et de porter les résultats des ressources lexicales multilingues basées sur pivot au formalisme des GU.

Les types d'unité sémantique profonde pourraient être considérés comme des acceptions interlingues, à l'instar des UW++ dans le projet UNL.

D'un autre côté, les TUSemP possèdent une structure actancielle formelle riche, sont organisés en une hiérarchie en cohérence avec cette structure actancielle, et peuvent faire l'objet d'une définition formelle sous forme de graphe. Lorsque l'on plongera des ressources lexicales multilingues existantes dans le formalisme des GU, il conviendra donc d'agir avec précaution puisque l'on gagnera en précision.

Nous faisons ainsi écho à la section 3.1.3, où nous avons envisagé le lien entre différents lexiques, comme dans le projet Papillon (Mangeot *et al.*, 2003) par exemple. Nous avons sous-entendu que /to eat\ et /manger\ ne devraient pas être confondus, car la structure actancielle de /to eat\ est plus riche que celle de /manger\.

La précision descriptive atteinte dans le formalisme des GU permet ainsi d'envisager l'enrichissement des ressources lexicales multilingues existantes. Pointons également les travaux de Rouquet (2012) concernant l'alignement entre une ontologie et un lexique interlingue, qui pourraient être un point de départ pour un travail futur.

11.4 Applications en TALN

L'étude que nous avons menée nous rapproche sensiblement des applications potentielles du système ULiS qui a fait l'objet des travaux préliminaires de cette thèse (Lefrançois et Gandon, 2011c,b).

Le projet ULiS avait originellement pour sujet d'étude la conception d'un système expert à base de pivot, conçu en conformité avec la TST, avec les formalismes du Web Sémantique, et dans le but de permettre la gestion multilingue de bases de connaissances.

Nous présentons donc dans cette section une mise à jour des scénarios envisagés pour le projet ULiS :

- la traduction automatique (§11.4.1) ;
- l'interaction avec un niveau conceptuel (§11.4.2) ;
- l'amélioration du système via le système (introspection, §11.4.3).

11.4.1 Systèmes de traduction automatique

Les perspectives de traduction automatique mobilisent les travaux futurs de lexicologie multilingue (§11.3.4), et de l'étude des cascades de transducteurs (§11.2.4.3).

Supposons que le niveau sémantique profond puisse effectivement être vu comme un niveau interlingue. Supposons que l'on ait généré l'ensemble de transducteurs représentant les composantes du modèle Sens-Texte de deux langues différentes. En composant les transducteurs d'analyse d'une langue et les transducteurs de génération de l'autre, nous obtenons une cascade de transducteurs permettant la traduction automatique entre la première et la seconde langue. La figure 11.7 représente ce scénario.

L'approximation de cette cascade de transducteurs par un unique transducteur stochastique de chaînes de caractères permettrait de trouver un compromis entre simplicité et qualité de la traduction.

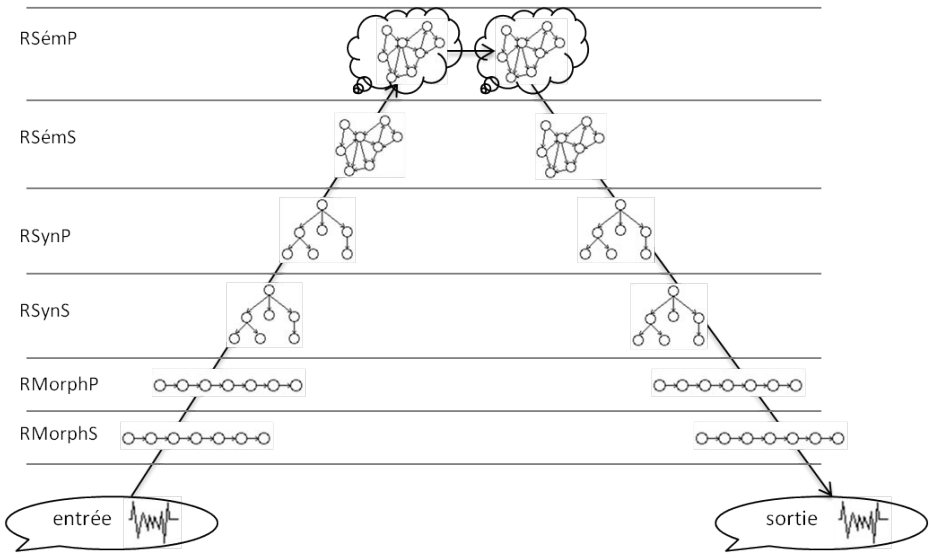


FIGURE 11.7 – Systèmes de traduction automatique.

11.4.2 Interaction avec un niveau conceptuel

Cette section offre un premier aperçu des perspectives à long terme de nos travaux concernant l'interaction homme-machine en langue naturelle.

Raisonnement dans et hors du formalisme des GU. La déduction logique dans le formalisme des GU n'est que le reflet de la logique interne du formalisme des GU. Par cela, nous entendons préciser que le raisonnement dans une représentation syntaxique ne concerne que la syntaxe. Nous avons défini le niveau sémantique profond de sorte qu'il corresponde aux situations linguistiques. Le raisonnement à ce niveau concerne donc les situations linguistiques. A priori, il n'est donc pas question d'y effectuer du raisonnement spatio-temporel, flou, multi-modal, d'ordre supérieur, ou autre raisonnement dépendant d'un domaine ou d'une logique particulière.

Articulation avec d'autres formalismes logiques. Suivant le type de raisonnement souhaité, il s'agit donc :

- de proposer une articulation entre le niveau sémantique profond et un autre formalisme logique ;
- de plonger une représentation sémantique profonde dans le formalisme logique pour y effectuer le type de raisonnement souhaité.

Cas de l'articulation avec des ontologies de domaines. Une des raisons d'être du modèle **ontolex** développé par le groupe communautaire Ontology-Lexica est justement de définir de telles articulations entre un lexique et une ontologie de domaine (quelle qu'en soit la logique associée). Une application possible de nos travaux concerne la génération automatique, à l'aide de requêtes SPARQL TEMPLATE, d'une base de règles de transformation pour assurer le passage d'une représentation sémantique profonde vers un fait dans l'ontologie cible, et vice-versa.

Cas particulier d'une articulation avec le vocabulaire SPIN. Le vocabulaire SPIN[4] propose une syntaxe RDF des règles SPARQL Query et SPARQL Update.

Forts de d'articulations entre un DEC et : (i) SPIN d'une part, (ii) une ontologie d'un domaine **D** d'autre part, il serait possible de plonger un GU dans le monde de SPIN et du domaine **D**. Le graphe RDF obtenu ne serait alors pas destiné au raisonnement logique, mais plutôt à être évalué sur une base de connaissances du domaine **D**.

Interaction avec une base de connaissances en langue naturelle. Nous mobilisons ici les travaux futurs de représentation de la structure communicationnelle (§11.2.3). Suivant l'opération demandée par l'utilisateur, nous pourrions donc interagir avec une base de connaissances en langue naturelle, et éventuellement renvoyer une réponse à l'utilisateur. Ce scénario est illustré sur la figure 11.8.

4. SPIN - SPARQL Inferencing Notation - http://www.w3.org/Submission/spin-overview/

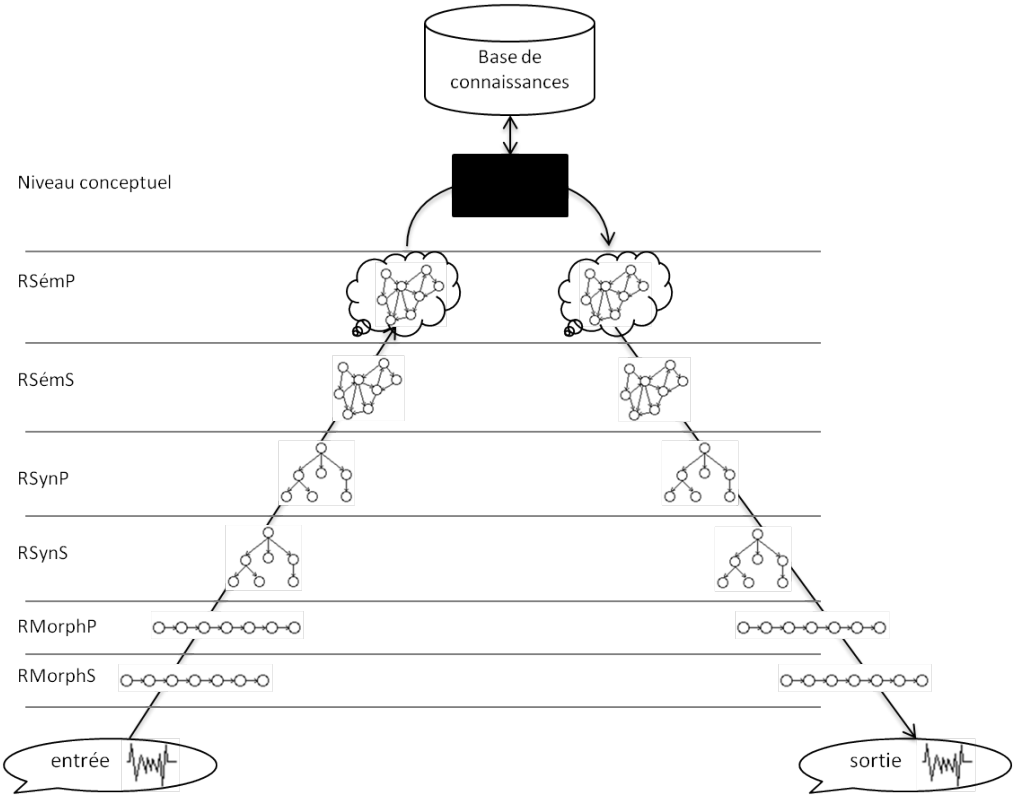


FIGURE 11.8 – Interaction avec une base de connaissances en langue naturelle.

11.4.3 Amélioration du système via le système

Enfin, le scénario d'amélioration du système via le système découle naturellement du scénario précédent, et du fait que le formalisme des GU possède une modélisation avec les formalismes du Web Sémantique. En supposant que l'on définisse l'articulation entre : (i) chacune des ontologies de la pyramide de la figure 10.2, et (ii) SPIN et un DEC d'introspection, alors il serait possible améliorer le DEC en langue naturelle. Ce scénario est illustré sur la figure 11.9.

Étant donné que d'un meilleur DEC on générerait une meilleure cascade de transducteurs, ce scénario offre à nouveau la perspective de définir un cercle vertueux pour l'apprentissage (semi-automatisé cette fois) du DEC.

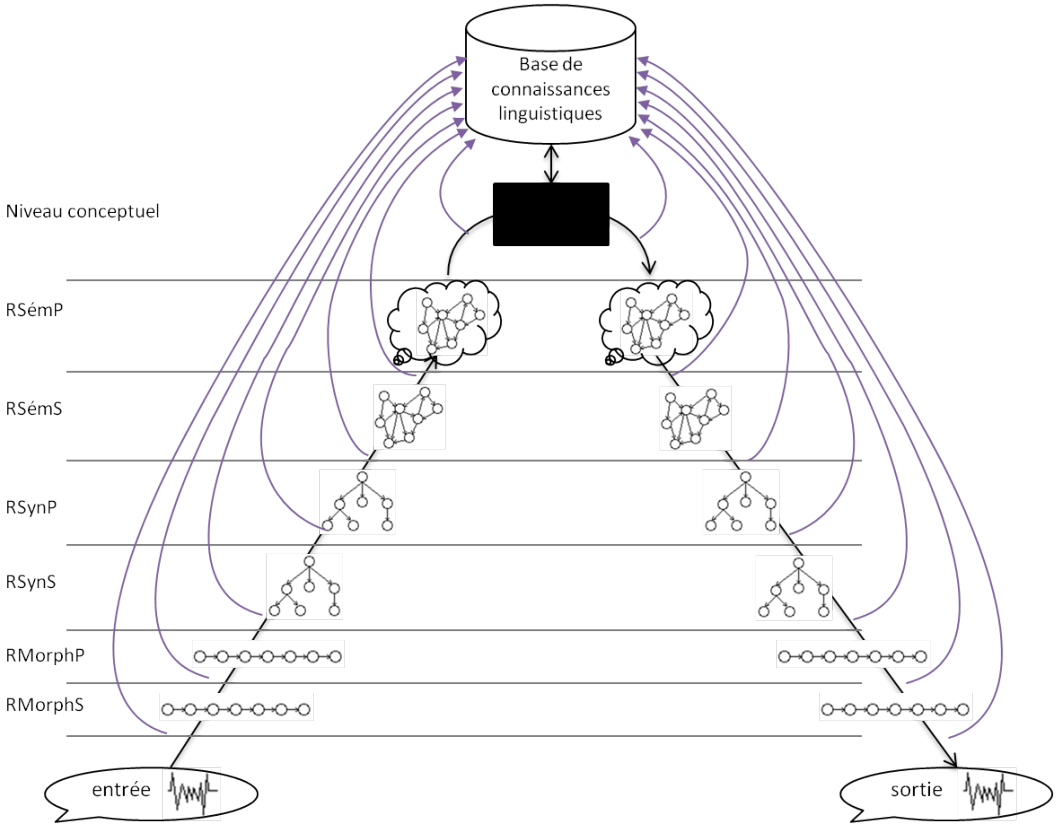


FIGURE 11.9 – Amélioration du système via le système.

11.5 Lexicologie et Ingénierie des connaissances

La lexicologie s'attache à définir un modèle pour les dictionnaires, ainsi qu'un ensemble de critères linguistiques qui mobilisent l'intuition des lexicographes, afin de leur permettre de motiver leurs décisions aux différentes étapes de l'écriture d'une entrée de dictionnaire.

La méthodologie de l'ingénierie des connaissances pourrait bénéficier de ces méthodologies à base de critères. En effet, même lorsque les choix lexicographiques concernent des concepts abstraits tels que la structure actancielle sémantique d'une lexie, des critères linguistiques mobilisant la connaissance de l'usage de la langue sont définis. Nous pensons en particulier aux travaux de définition des ontologies de haut niveau, qui représentent des concepts généraux.

Par ailleurs, la méthodologie de l'ingénierie des connaissances que nous avons utilisée offre une boucle de rétroaction sur les travaux des lexicologues et des lexicographes :

– l'extension de la conceptualisation que nous avons proposée pourrait permettre aux lexicologues de préciser leur théorie et de définir de nouveaux critères linguistiques. Nous pensons en particulier à l'introduction des participants interdits.
– la représentation et l'opérationnalisation de la TST que nous avons initiée devrait apporter aux lexicographes un contrôle plus fin de leur objet de travail. Nous pensons ici aux scénarios mis en avant dans la section 2.1.

Conclusion

Nous avons présenté les différentes perspectives de recherche ouvertes par nos travaux.

Nous avons d'abord présenté différentes directions de recherche concernant l'opérationnalisation du formalisme des GU dans son état actuel :

- Nous pouvons déterminer une condition nécessaire et suffisante pour la décidabilité du raisonnement logique en présence de définitions de TUP, ainsi que la complexité des différents problèmes de raisonnement.
- Nous travaillons sur une implémentation du formalisme des GU en deux étapes : d'abord de manière naïve avec le moteur Corese, puis en proposant des optimisations spécifiques.

Les règles de GU peuvent être révisées dans le but de représenter les composantes optionnelles et interdites des définitions lexicographiques formelles, dans leur conceptualisation actuelle. Nous avons présenté l'idée principale de cette révision, ainsi que son impact pressenti sur l'opérationnalisation du formalisme des GU.

Enfin, la conceptualisation des définitions lexicographiques doit être complétée pour prendre en compte les composantes faibles, les parties présuppositionnelles, et la polysémie lexicale. Nous avons présenté un plan de travail pour mener ces études en tandem avec les linguistes de la TST, et l'impact pressenti sur le formalisme des GU et son opérationnalisation.

Nous pouvons poursuivre notre étude d'ingénierie des connaissances pour d'autres aspects de la TST :

- Nous avons donné un avant goût des questions de recherche que soulèveront l'étude des structures actancielles syntaxiques ;
- Nous avons justifié en quoi le formalisme des GU, dans son état actuel, semble prometteur pour la représentation des Fonctions Lexicales.
- Nous avons démontré l'importance de la représentation de la structure communicationnelle des représentations linguistiques pour l'interaction en langue naturelle, et nous avons listé les domaines que cette étude mobiliserait.
- Les règles des GU peuvent être utilisées pour décrire les transformations élémentaires entre deux représentations linguistiques de niveaux adjacents. Il est nécessaire d'unifier ces règles pour transformer un GU en un autre. Nous entrevoyons donc que les modules de correspondance du modèle Sens-Texte peuvent être représentés par des transducteurs stochastiques de GU. Nous avons présenté différentes directions de recherche concernant l'élaboration de ces transducteurs, et l'étude de leur mise en cascade.

Nous avons présenté différentes applications directes de nos travaux. Dans la continuité de nos travaux présentés dans le chapitre 5, nous souhaitons intégrer nos travaux dans un éditeur de DEC.

Grâce à nos travaux concernant l'opérationnalisation du formalisme des GU sur le Web des données, nous pouvons envisager de construire automatiquement une ébauche de DEC à partir de bases de connaissances lexicales existantes.

Le niveau sémantique profond que nous avons défini peut être conçu comme un niveau interlingue. Nos travaux pourraient donc servir de point de départ pour une étude en lexicologie multilingue.

Au dessus de ces perspectives de recherche, nous avons présenté une mise à jour de trois scénarios de TALN envisagés pour le système ULiS :

- la traduction automatique ;

– l'interaction avec une base de connaissances en langue naturelle ;
– l'amélioration du système par une interaction avec le système.

Enfin, nous avons entrevu en quoi les méthodologies développées par les lexicologues et celles développées par les ingénieurs de la connaissance pourraient s'enrichir mutuellement.

Concluons sur deux remarques d'ordre général :

– Le formalisme des GU enrichit la famille des formalismes de représentation des connaissances. Il pourrait s'avérer adapté pour représenter les connaissances d'autres domaines de spécialité.
– En considérant la langue comme un système de signes parmi d'autres, une étude de représentation des connaissances menée de manière similaire à la nôtre pourrait mener aux mêmes perspectives applicatives que celles que nous avons décrite dans ce chapitre. Ce pourrait être le cas pour d'autres langages comme les langues des signes, qui possèdent leurs dictionnaires et leurs grammaires, ou bien pour d'autres systèmes de signes.

Conclusion générale

Le rêve est une construction de l'intelligence à laquelle le constructeur assiste sans savoir comment cela va finir.
Cesare Pavese, Le métier de vivre

En tant que théorie linguistique hautement précise du point de vue descriptif, la théorie linguistique Sens-Texte (TST) tient donc une place centrale dans le champ de la sémantique lexicale. La richesse de description fait sa force, mais elle a aussi des limites. En particulier, la représentation des définitions lexicographiques, au sens de l'Ingénierie des Connaissances (IC), n'a pas été étudiée et se fait désirer pour en faire bénéficier les applications en lexicographie et en TALN. Nous avons donc mené une étude en IC appliquée aux connaissances linguistiques de la TST, et pouvons maintenant répondre à la question de recherche principale de cette thèse, à savoir :

Quel formalisme de représentation des connaissances serait adapté à la représentation des prédicats linguistiques, des représentations linguistiques, et des définitions lexicographiques du Dictionnaire Explicatif et Combinatoire (DEC) ?

Nous proposons dans cette conclusion générale un résumé des contributions de nos travaux, puis une liste commentée des publications de nos travaux.

Résumé des contributions

Pour répondre à cette question, nous avons adopté une méthodologie en trois étapes inspirée de (Bachimont, 2000), qui définit également la structure générale de ce mémoire :

1. extension de la conceptualisation de la TST ;
2. choix d'un formalisme de représentation des connaissances adapté ;
3. opérationnalisation de ce formalisme.

Extension de la conceptualisation de la TST

La première étape de notre travail concernait l'extension de la conceptualisation de la TST, pour en faciliter la formalisation ultérieure.

Conceptualisation du sens au niveau sémantique profond. Au delà de certaines précisions terminologiques et notationnelles, notamment la distinction *type d'unité linguistique* (décrit dans le dictionnaire) et *unité linguistique* (utilisé dans un énoncé), nous avons motivé l'introduction de deux notions fondamentales pour formaliser le DEC :

- un niveau sémantique profond, confondu avec le niveau des situations linguistiques, pour représenter les sens ;
- un nouveau type de participant dans les situations linguistiques : les participants interdits.

Les types d'unité linguistique opérant au niveau sémantique profond portent le sens lexical, et peuvent être vus comme des étiquettes sémantiques auxquelles on ajouterait une structure actancielle. Nous avons défini ces types d'unité sémantique profonde (TUSemP) et leur structure actancielle de sorte que :

- la hiérarchie des TUSemP représente une hiérarchie des sens ;
- la structure actancielle est héritée et éventuellement spécialisée au sein de la hiérarchie des TUSemP, suivant les définitions 17 et 19.
- il existe une correspondance simple entre la structure actancielle d'un TUSemP et la structure actancielle sémantique du type d'unité lexicale associé.

Précision de la visualisation des représentations linguistiques. Les unités linguistiques dont la conceptualisation est ainsi précisée sont interconnectées dans des représentations linguistiques d'énoncés via des relations actancielles, circonstancielles, ou de coréférence anaphorique. Nous avons motivé des choix de visualisation pour les représentations linguistiques.

Reconceptualisation des définitions lexicographiques. Les définitions lexicographiques ont alors fait l'objet d'un travail de reconceptualisation important. Nous avons justifié leur repositionnement : (i) au niveau sémantique profond, et (ii) au niveau du dictionnaire. Le résultat de ce repositionnement est appelé *définition lexicographique formelle*, et peut être identifié au concept de situation linguistique dénoté. Nous avons finalement montré comment effectuer validation et inférence au sein de ces définitions lexicographiques formelles. L'étude des composantes faibles, de la partie présuppositionnelle, et de la polysémie lexicale fait l'objet de travaux futurs de cette thèse.

Étude de l'utilisation de la conceptualisation étendue dans un projet de lexicographie explicative et combinatoire. Nous nous sommes intéressés à l'utilisation de la conceptualisation ainsi étendue pour l'édition de la zone sémantique dénotationnelle du DEC, dans le cadre d'un projet de lexicographie explicative et combinatoire. Nous avons étudié le processus suivi dans le projet RELIEF, et nous avons défini et implémenté un nouveau workflow adapté à la conceptualisation étendue. Le prototype d'éditeur développé propose des opération innovantes de manipulation de graphes par "glisser-déposer". L'évaluation du prototype d'éditeur que nous avons effectué auprès des lexicographes du projet RELIEF nous permet d'envisager l'amélioration de notre workflow, et une intégration future de nos travaux dans des logiciels de lexicographie.

Formalisation des prédicats sémantiques et des définitions lexicographiques

Une fois mise au point la conceptualisation étendue de la TST, il nous a fallu choisir un formalisme de Représentation des Connaissances (RC) adapté pour la formaliser. Nous avons justifié l'utilisation des critères de choix de Gruber (1995).

Candidature des standards de RC existants. Ayant justifié la pertinence des candidatures des logiques de description et du formalisme des Graphes Conceptuels, nous avons dans un premier temps étudié leur adéquation pour la formalisation des connaissances de la TST.

Les logiques de description sont bien adaptées pour la formalisation d'une hiérarchie de types d'unité linguistique munis de structures actancielles, mais ne permettent pas de formaliser les définitions lexicographiques formelles de manière satisfaisante. Nous avons néanmoins montré que les définitions lexicographiques formelles pourraient être formalisées par projection sur un type de

concept. Ce problème de projection a été résolu pour deux exemples simples de définitions lexicographiques formelles, et nous avons montré que la représentation ainsi obtenue n'est pas claire et ne respecte pas notre engagement ontologique de décidabilité du raisonnement logique.

En ce qui concerne le formalisme des Graphes Conceptuels, nous avons identifié deux options de modélisation. L'option la plus proche de satisfaire les critères de Gruber consiste à modéliser chaque relation actancielle par une relation binaire. Nous avons néanmoins identifié trois limites importantes à l'utilisation de cette modélisation :

- elle implique une déformation d'encodage importante ;
- la sémantique naturelle des graphes conceptuels n'est pas une sémantique naturelle pour la TST ;
- des mécanismes non-standards de vérification de la cohérence de la base de règles d'inférence devraient être proposés.

Construction du formalisme des Graphes d'Unités. Nous avons ainsi justifié la construction d'un nouveau formalisme de RC, dit *des Graphes d'Unités*, spécialement adapté à la conceptualisation étendue des prédicats linguistiques et des définitions lexicographiques. Le cœur de ce mémoire concerne donc l'introduction et la caractérisation des objets mathématiques fondamentaux du formalisme des GU :

- une hiérarchie de types d'unité dotés d'une structure actancielle, qui permet de représenter les prédicats linguistiques ;
- une hiérarchie de symboles de relation circonstancielle : un ensemble de relations binaires non actancielles ;
- les Graphes d'Unités, qui permettent de représenter les représentations linguistiques ;
- les règles de GU, utilisées pour représenter :
  - les règles de correspondance entre les prédicats linguistiques de deux niveaux de représentation adjacents ;
  - un sous-ensemble des définitions lexicographiques formelles : celles qui ne contiennent que des composantes obligatoires.

Sur cette partie du travail, nous avons aussi notamment identifié les extensions possibles suivantes :

- la définition des Graphes d'Unités peut être révisée pour permettre de représenter la structure communicationnelle des énoncés ;
- la définition des règles de GU peut être révisée pour représenter les composantes optionnelles et interdites des définitions lexicographiques formelles ;
- les règles de GU peuvent être rendues génériques, formant un point de départ intéressant pour la représentation des Fonctions Lexicales de la TST ;
- les modules de correspondance du modèle Sens-Texte, qui décrivent la transformation d'une représentation linguistique d'un niveau (ex. sémantique) en une représentation linguistique d'un autre niveau (ex. syntaxique), peuvent être représentés par des transducteurs stochastiques de GU, définis à partir de règles de GU.

Opérationnalisation du formalisme des Graphes d'Unités

Le formalisme des GU que nous avons construit permet de représenter un sous-ensemble consistant des connaissances de la TST. Dans la troisième partie de cette thèse, nous nous sommes intéressés à l'opérationnalisation du formalisme des GU. Nous avons en particulier étudié :

– le raisonnement logique au sein du formalisme des GU ;
– l'opérationnalisation du formalisme sur le Web des données.

Résumons les contributions de nos travaux concernant ces deux études.

Raisonnement dans le formalisme des Graphes d'Unités. Nous avons tout d'abord défini une sémantique d'interprétation pour le formalisme des GU, basée sur l'algèbre relationnelle. Parallèlement à cela, nous avons introduit une base de règles axiomatiques d'inférence, permettant d'expliciter les connaissances dans les GU. Nous avons alors défini et caractérisé la déduction logique dans le formalisme des GU à l'aide de cette base de règles d'inférence et des homomorphismes de GU : un graphe GU H peut être déduit d'un GU G s'il existe un homomorphisme de H vers une dérivation de G.

Une condition suffisante pour que le problème de déduction soit décidable est que tout GU possède une expansion finie, c'est-à-dire un GU dérivé sur lequel aucune des règles axiomatiques d'inférence n'a d'effet. Nous avons déterminé une condition nécessaire et suffisante pour que tout GU possède une expansion finie en l'absence de définitions lexicographiques, ainsi qu'une condition suffisante en présence de définitions lexicographiques. Bien que les conditions de décidabilité pourraient être affinées dans des travaux futurs, celles que nous avons mis en lumière sont déjà adoptées en pratique par les lexicographes.

Enfin, nous avons montré que le problème de déduction logique par dérivation et homomorphisme est correct vis-à-vis de la sémantique formelle, et complet si les conditions suffisantes de décidabilité sont satisfaites.

Opérationnalisation sur le Web des données. Nous avons développé une combinaison de deux modèles hétérogènes mais interopérables, en conformité avec le métamodèle OWL 2 RL.

Le premier de ces modèles, **ug-language**, a été aligné au standard **ontolex** de représentation des connaissances lexicales sur le Web des données. Le second modèle, **gu-usage**, permet de représenter les GU.

L'interopérabilité de ces deux modèles, ainsi que la sémantique logique du formalisme des GU, a été capturée par un ensemble de règles SPARQL qui étend la base de règles OWL 2 RL/RDF. Nous avons notamment construit une base de règles SPARQL qui permet de générer une base de règles SPARQL de contraction et d'expansion pour les définitions lexicographiques.

Ce dernier apport de nos travaux de thèse offre une premier implémentation au formalisme des GU, et permet à la TST de commencer à profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances sur le Web des données.

Perspectives applicatives de nos travaux. Nous avons ainsi œuvré pour l'externalisation de la sémantique logique de la TST. Nos travaux ouvrent de nombreuses perspectives de recherche, tant pour la représentation de la TST que pour l'ingénierie des connaissances, ou pour le traitement des langues naturelles. Nous avons en particulier présenté des perspectives applicatives concernant :

– l'élaboration du lexique : éditeur de définitions lexicographiques, validation de la cohérence du lexique, découverte de nouvelles connaissances ;
– la participation à l'amélioration des applications de TALN.

Liste commentée de publications

Nos travaux ont fait l'objet de plusieurs publications en conférences et ateliers internationaux et nationaux.

Le projet ULiS est le résultat des travaux préliminaires de cette thèse.
(Lefrançois et Gandon, 2011a) propose à la communauté de la TST un premier modèle naïf pour représenter les définitions lexicographiques avec les logiques de description. Nous y définissons le problème de la projection d'une définition lexicographique sur un type de concept (cf., §6.1.2), et menons nos premières réflexions sur le positionnement des définitions lexicographiques formelles au niveau du dictionnaire, et au niveau des situations linguistiques (ce que nous appelions le niveau conceptuel dans cet article correspond au niveau des situations linguistiques, et a été renommé niveau sémantique profond par la suite).

L'article (Lefrançois et Gandon, 2011b) présente à la communauté de la terminologie et de l'intelligence artificielle une première version de l'architecture générale du modèle des GU (cf., §10.1.3), pour laquelle la lexicologie multilingue tient une place importante (cf., §11.3.4), et envisage les scénarios applicatifs mis à jour dans la section 11.4.

(Lefrançois et Gandon, 2011c) reprend l'ensemble des travaux sur ULiS et sur la représentation des définitions lexicographiques avec les logiques de description, et les présentent selon le point de vue de la représentation des connaissances lexicales sur le Web de données.

Une partie de l'extension de la conceptualisation des prédicats linguistiques et des définitions lexicographiques a été publiée dans (Lefrançois et Gandon, 2013c). Nous y démontrons notamment la nécessité d'introduire : (i) un niveau sémantique profond ; (ii) la notion de participant interdit dans les situations linguistiques.

Le prototype d'éditeur de définitions lexicographiques formelles présenté dans le chapitre 5 a fait l'objet des publications suivantes :

- (Lefrançois *et al.*, 2013) présente aux linguistes de la TST les définitions lexicographiques formelles, ainsi que le scénario d'édition proposé ;
- (Lefrançois *et al.*, 2014b) présente succinctement le prototype d'éditeur d'un point de vue ingénierie des connaissances.

(Lefrançois, 2013) fut le premier article publié à décrire le formalisme des GU. Nous y justifions brièvement l'inadéquation des logiques de description et du formalisme des Graphes Conceptuels, nous décrivons succinctement nos précisions terminologiques et notationnelles (cf., §3.1), et nous introduisons les objets fondamentaux du formalisme : la hiérarchie de types d'unité, la hiérarchie des symboles de relation circonstancielle, et les graphes d'unités.

L'homomorphisme entre GU, les règles de GU et les définitions de TUP ont été ensuite partiellement définies dans (Lefrançois et Gandon, 2013b).

La sémantique d'interprétation du formalisme des GU sans définitions de TUP, ainsi que la conséquence sémantique, a été décrite dans (Lefrançois et Gandon, 2013d).

L'article (Lefrançois et Gandon, 2013b) proposait également une seconde sémantique pour les GU, que nous n'avons pas retranscrite dans ce mémoire. De ces travaux préliminaires ont découlé la définition de la base de règles axiomatiques d'inférence (cf., §9.2), la déduction logique par homomorphisme et clôture de graphes (cf., §9.3.1), et l'étude des conditions suffisantes de décidabilité du raisonnement (cf., §9.3.2).

(Lefrançois et Gandon, 2013a) résume le problème de la représentation des connaissances de la TST, et offre le meilleur point d'entrée au formalisme des GU à la lumière des différents articles publiés en 2013 et avant.

Ces premiers apports ainsi que la plupart des démonstrations des théorèmes et propositions ont été compilés dans le rapport de recherche (Lefrançois et Gandon, 2013e).

Enfin, (Lefrançois *et al.*, 2014a) propose une réflexion sur la méthodologie d'ingénierie des connaissances en trois étapes que nous avons adopté pour nos travaux.

Bibliographie

ALONSO RAMOS, M. (2003). Hacia un Diccionario de colocaciones del español y su codificación. *Lexicografía computacional y semántica*, pages 11–34.

ALONSO RAMOS, M. (2004). Elaboración del Diccionario de colocaciones del español y sus aplicaciones. *In* BATTANER, P. et DECESARIS, J., éditeurs : *De Lexicographia. Actes del I symposium internacional de lexicografia*, pages 149–162. Edicions Petició, Barcelona.

ALTMAN, J. et POLGUÈRE, A. (2003). La BDéf : base de définitions dérivée du Dictionnaire explicatif et combinatoire. *In Proceedings of the First International Conference on Meaning-Text Theory (MTT'2003)*, pages 43–54, Paris, France.

APRESIAN, J., BOGUSLAVSKY, I., IOMDIN, L., LAZURSKY, A., SANNIKOV, V., SIZOV, V. et TSINMAN, L. (2003). ETAP-3 linguistic processor : a full-fledged NLP implementation of the MTT. *Proceedings of the First International Conference on Meaning–Text Theory (MTT'2003)*, pages 279–288.

AUSSENAC-GILLES, N., BIÉBOW, B. et SZULMAN, S. (2000). Revisiting Ontology Design : A Method Based on Corpus Analysis. *In* DIENG, R. et CORBY, O., éditeurs : *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, volume 1937 de *Lecture Notes in Computer Science*, pages 172–188. Springer Berlin Heidelberg, Berlin, Heidelberg.

AUSSENAC-GILLES, N., KAMEL, M., BUSCALDI, D. et COMPAROT, C. (2013). Construction d'ontologies à partir d'une collection de pages web structurées. *In IC 2013 - 24èmes Journées Francophones d'Ingénierie des Connaissances*, pages 1–17.

BAADER, F., CALVANESE, D., MCGUINESS, D., NARDI, D. et PATEL-SCHNEIDER, P., éditeurs (2003). *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press.

BACHIMONT, B. (2000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances. *In* CHARLET, J., ZACKLAD, M., KASSEL, G. et BOURIGAULT, D., éditeurs : *Ingénierie des connaissances, évolutions récentes et nouveaux défis*, chapitre 19. Eyrolles.

BAGET, J. (2001). Représenter des connaissances et raisonner avec des hypergraphes : de la projection à la dérivation sous contraintes. *Université de Montpellier II. Thèse*.

BAGET, J., LECLÈRE, M., MUGNIER, M. et SALVAT, E. (2009). Extending Decidable Cases for Rules with Existential Variables. *IJCAI*.

BAGET, J. et MUGNIER, M. (2002). Extensions of simple conceptual graphs : the complexity of rules and constraints. *Journal of Artificial Intelligence Research*, 16:425–465.

BAGET, J. et SALVAT, E. (2006). Rules dependencies in backward chaining of conceptual graphs rules. *Conceptual Structures : Inspiration and Application*.

BAGET, J.-F., CROITORU, M., LECLÈRE, M., MUGNIER, M.-L. et GUTIERREZ, A. (2010). Translations between RDF(S) and conceptual graphs. *In 18th International Conference on Conceptual Structures - From Information to Intelligence (ICCS'2010)*, pages 28–41.

BAGET, J.-F., GENEST, D. et MUGNIER, M.-L. (1999a). A pure graph-based solution to the SCG-1 initiative. *In* TEPFENHART, W. M. et WALLING, R. C., éditeurs : *Conceptual Structures : Standards and Practices, 7th International Conference on Conceptual Structures (ICCS '1999)*, pages 355–376, Blacksburg, Virginia, USA. Springer.

BAGET, J.-F., GENEST, D. et MUGNIER, M.-L. (1999b). Knowledge acquisition with a pure graph-based knowledge representation model. *In twelfth Workshop on Knowledge Acquisition, Modeling and Management*, pages 567–576. Springer-Verlag.

BAKER, C., FILLMORE, C. J. et LOWE, J. B. (1998). The berkeley framenet project. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1:16–60.

BARQUE, L. (2008). *Description et formalisation de la polysémie régulière du français*. Thèse de doctorat, Université Paris 7.

BARQUE, L., NASR, A. et POLGUÈRE, A. (2010). From the Definitions of the 'Trésor de la Langue Française' To a Semantic Database of the French Language. *In* FRYSKE AKADEMY, éditeur : *Proceedings of the XIV Euralex International Congress*, Fryske Akademy, pages 245–252, Leeuwarden, Pays-Bas. Anne Dykstra et Tanneke Schoonheim, dir.

BARQUE, L. et POLGUÈRE, A. (2013). Enrichissement formel des définitions du Trésor de la Langue Française informatisé (TLFi) dans une perspective lexicographique. *Lexique*, 21:221–244.

BERNERS-LEE, T. (2006). Linked Data - Design Issues.

BOGUSLAVSKY, I., IOMDIN, L. et SIZOV, V. (2004). Multilinguality in ETAP-3 : reuse of lexical resources. *In* SÉRASSET, G., éditeur : *Proc. COLING 2004 Multilingual Linguistic Ressources*, pages 1–8, Geneva, Switzerland. COLING.

BOHNET, B. et WANNER, L. (2010). Open source graph transducer interpreter and grammar development environment. *In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 19–21, Valletta, Malta. European Language Resources Association (ELRA).

BOUMA, G. (1992). Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2): 183–203.

BOUVERET, M. et FILLMORE, C. (2008). Matching Verbo-nominal Constructions in FrameNet with Lexical Functions in MTT. *In Proceedings of the Euralex'2008*, pages 297–308, Barcelona.

CAO, T. H., CREASY, P. N. et WUWONGSE, V. (1997). Fuzzy unification and resolution proof procedure for fuzzy conceptual graph programs. *In* LUKOSE, DICKSON AND DELUGACH, HARRY AND KEELER, MARY AND SEARLE, LEROY AND SOWA, J., éditeur : *Conceptual Structures : Fulfilling Peirce's Dream*, volume 1257 de *Lecture Notes in Computer Science*, pages 386–400. Springer Berlin Heidelberg.

CHEIN, M. et MUGNIER, M. (2004). Concept types and coreference in simple conceptual graphs. *In Conceptual Structures at Work*, volume 3127 de *Lecture Notes in Computer Science*, pages 303–318. Springer.

CHEIN, M. et MUGNIER, M.-L. (1992). Conceptual graphs : Fundamental notions. *Revue d'intelligence artificielle*, 6:365–406.

CHEIN, M. et MUGNIER, M.-L. (1995). Conceptual graphs are also graphs. Rapport technique, LIRMM (CNRS & Université de Montpellier II).

CHEIN, M. et MUGNIER, M.-L. (2009). *Graph-based Knowledge Representation : Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer.

CHIARCOS, C., HELLMANN, S. et NORDHOFF, S. (2011). Towards a Linguistic Linked Open Data cloud : The Open Linguistics Working Group. *TAL*, 52(3):245–275.

CIMIANO, P., BUITELAAR, P., MCCRAE, J. et SINTEK, M. (2011). LexInfo : A Declarative Model for the Lexicon-Ontology Interface. *Web Semantics : Science, Services and Agents on the World Wide Web*, 9(1):29–51.

CORBY, O., DIENG, R. et HÉBERT, C. (2000). A Conceptual Graph Model for W3C Resource Description Framework. *In* GANTER, B. et MINEAU, G. W., éditeurs : *Conceptual Structures : Logical, Linguistic, and Computational Issues*, numéro 1867 de Lecture Notes in Computer Science, pages 468–482. Springer Berlin Heidelberg.

CORBY, O., DIENG-KUNTZ, R. et FARON-ZUCKER, C. (2004). Querying the semantic web with corese search engine. *In ECAI*, volume 16, page 705.

CORBY, O., FARON-ZUCKER, C. et GANDON, F. (2014). SPARQL Template : A Transformation Language for RDF. Rapport de recherche, Inria.

COYNE, R. et RAMBOW, O. (2009). Meaning-text-theory and lexical frames. *In International Conference on Meaning-Text Theory 2009 (MTT'2009)*, pages 119–128, Montreal, Canada.

CUENCA GRAU, B., HORROCKS, I. et KRÖTZSCH, M. (2013). Acyclicity notions for existential rules and their application to query answering in ontologies. *Journal of Artificial Intelligence Research*, 47:741–808.

DALRYMPLE, M. (1999). Lexical-Functional Grammar. *In* WILSON, R. A. et KEIL, F. C., éditeurs : *the MIT Encyclopedia of the Cognitive Sciences*. The MIT Press.

DELAFORGE, N., GANDON, F. et MONNIN, A. (2012). L'avenir du web au prisme de la ressource. *In Séminaire IST Inria : le document numérique à l'heure du web de données*, pages 229–252.

DONG, Z. et DONG, Q. (2006). *HowNet and the Computation of Meaning*. World Scientific, London, UK.

DOWTY, D. R. (1979). *Word meaning and Montague grammar : The semantics of verbs and times in generative semantics and in Montague's PTQ*. Studies in Linguistics and Philosophy. Springer.

DOWTY, D. R., WALL, R. et PETERS, S. (1980). *Introduction to Montague semantics*. Studies in Linguistics and Philosophy. Springer.

ECKLE-KOHLER, J. et GUREVYCH, I. (2012). Standardizing Lexical-Semantic Resources–Fleshing out the abstract standard LMF. *Proceedings of the KONVENS'12 Workshop on Standards for Language Resources–Ongoing Developments and Practical Applications*, pages 194–198.

ECKLE-KOHLER, J., MCCRAE, J. et CHIARCOS, C. (2014). lemonUby-a large, interlinked, syntactically-rich resource for ontologies. *Semantic Web Journal*, (special issue on Multilingual Linked Open Data).

EVANS, R. et GAZDAR, G. (1989). The semantics of DATR. *In Proceedings of AISB-89*, pages 79–87.

EVANS, R. et GAZDAR, G. (1996). DATR : A language for lexical knowledge representation. *Computational Linguistics*, 22(2):167–216.

FELLBAUM, C. (1998). *WordNet : an electronic lexical database*. MIT Press, Cambridge, MA.

FILLMORE, C. (1977). Scenes-and-frames semantics. *In* ZAMPOLLI, A., éditeur : *Linguistic structures processing*, pages 55–81. Amsterdam : North-Holland.

FILLMORE, C. J., JOHNSON, C. R. et PETRUCK, M. R. (2003). Background to framenet. *International Journal of Lexicography*, 16(3):235–250.

FLOURIS, G., MANAKANATAS, D., KONDYLAKIS, H., PLEXOUSAKIS, D. et ANTONIOU, G. (2008). Ontology change : classification and survey. *The Knowledge Engineering Review*, 23(02):117–152.

FORGY, C. (1982). Rete : A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*.

FRANCOPOULO, G., BEL, N., GEORGE, M., CALZOLARI, N., MONACHINI, M., PET, M. et SORIA, C. (2007). Lexical markup framework : ISO standard for semantic information in NLP lexicons. *In Proceedings of the Workshop of the GLDV Working Group on Lexicography at the Biennial Spring Conference of the GLDV*, Tubingen.

FRANCOPOULO, G., GEORGE, M., CALZOLARI, N., MONACHINI, M., BEL, N., PET, M. et SORIA, C. (2006). Lexical markup framework (LMF). *In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 233–236, Gena, Italy.

FRIBURGER, N. et MAUREL, D. (2004). Finite-state transducer cascades to extract named entities in texts. *Theoretical Computer Science*, 313(1):93–104.

GEERAERTS, D. (2010). *Theories of lexical semantics*. Oxford Univ Press.

GRUBER, T. (1995). Toward principles for the design of ontologies used for knowledge sharing ? *International journal of human-computer studies*, 43(5-6):907–928.

GUARINO, N., éditeur (1998). *Formal Ontology in Information Systems : Proceedings of the First International Conference (FIOS'98), June 6-8, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st édition.

GUGERT, R. (2013). Scénarisation d'interactions avec les objets du formalisme des Graphes d'Unités et prototypage d'un éditeur de définitions lexicographiques formelles. Rapport technique, Inria.

HELBIG, H. (2006). *Knowledge Representation and the Semantics of Natural Language*. Cognitive Technologies. Springer.

IORDANSKAJA, L. et MEL'ČUK, I. A. (1990). Semantics of Two Emotion Verbs in Russian : BOJAT'SJA 'to be afraid' & NADEJAT'SJA 'to hope'. *Australian Journal of linguistics*, 10(2):307–357.

KAHANE, S. (2002). Grammaire d'Unification Sens-Texte : Vers un modele mathématique articulé de la langue.

KAHANE, S. (2003). The Meaning-Text Theory. *Dependency and Valency, An International Handbooks of Contemporary Research*, 25(1):546–569.

KAHANE, S. (2004). Grammaires d'unification polarisées. *In Proceedings of TALN (TALN'2004)*.

KAHANE, S. et LAREAU, F. (2005). Grammaire d'Unification Sens-Texte : modularité et polarisation. *Proceedings of TALN (TALN'2005)*.

KAHANE, S. et POLGUÈRE, A. (2001). Formal foundation of lexical functions. *In Actes du colloque COLLOCATION : Computational Extraction, Analysis and Exploitation*, pages 8–15, Toulouse.

KELLER, B. (1995). DATR theories and DATR models. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 55–62.

LAFOURCADE, M. (2007). Making people play for Lexical Acquisition with the JeuxDeMots prototype. *In 7th International Symposium on Natural Language Processing (SNLP'07)*, Pattaya, Thailand.

LAREAU, F. (2003). *La synthese automatique de paraphrases comme outil de vérification des dictionnaires et grammaires de type sens-texte*. Mémoire de maîtrise, Université de Montréal.

LECLÈRE, M. (1998). Raisonner avec des définitions de types dans le modèle des graphes conceptuels. *Revue d'intelligence artificielle*, 12(2):243–278.

LEFRANÇOIS, M. (2013). Représentation des connaissances du DEC : Concepts fondamentaux du formalisme des Graphes d'Unités. *In Actes de la 15ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL'2013)*, pages 164–177, Les Sables d'Olonne, France.

LEFRANÇOIS, M. et GANDON, F. (2011a). ILexicOn : Toward an ECD-Compliant Interlingual Lexical Ontology Described with Semantic Web Formalisms. *In* BOGUSLAVSKY, I. et WANNER, L., éditeurs : *Proceedings of the 5th International Conference on Meaning-Text Theory (MTT'2011)*, pages 155–164, Barcelona, Spain. INALCO.

LEFRANÇOIS, M. et GANDON, F. (2011b). ULiS : An Expert System on Linguistics to Support Multilingual Management of Interlingual Knowledge bases. *In* KAGEURA, K. et ZWEIGENBAUM, P., éditeurs : *Proc. of the 9th International Conference on Terminology and Artificial Intelligence (TIA 2011)*, pages 108–114, Paris, France. INALCO.

LEFRANÇOIS, M. et GANDON, F. (2011c). ULiS : An Expert System on Linguistics to Support Multilingual Management of Interlingual Semantic Web Knowledge bases. *In* ELENA MONTIEL-PONSODA JOHN MCCRAE, P. B. et CIMIANO, P., éditeurs : *Proc. 2nd Workshop on the Multilingual Semantic Web, collocated with ISWC (MSW'2011)*, volume 775, pages 50–61, Paris, France. CEUR-WS.

LEFRANÇOIS, M. et GANDON, F. (2013a). Rationale, Concepts, and Current Outcome of the Unit Graphs Framework. *In Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, pages 382–388.

LEFRANÇOIS, M. et GANDON, F. (2013b). Reasoning with Dependency Structures and Lexicographic Definitions using Unit Graphs. *In Proc. of the 2nd International Conference on Dependency Linguistics (Depling'2013)*, Prague, Czech Republic. ACL Anthology.

LEFRANÇOIS, M. et GANDON, F. (2013c). The Unit Graphs Framework : A graph-based Knowledge Representation Formalism designed for the Meaning-Text Theory. *In Proceedings of the 6th International Conference on Meaning-Text Theory (MTT'2013)*, Prague, Czech Republic.

LEFRANÇOIS, M. et GANDON, F. (2013d). The Unit Graphs Framework : Foundational Concepts and Semantic Consequence. *In Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, Hissar, Bulgaria. ACL Anthology.

LEFRANÇOIS, M. et GANDON, F. (2013e). The Unit Graphs Mathematical Framework. Research Report RR-8212, Inria.

LEFRANÇOIS, M., GANDON, F. et GIBOIN, A. (2014a). Méthodologie d'ingénierie des connaissances pour la représentation des définitions lexicographiques de la théorie Sens-Texte. *In Proc. 8th International Conference Terminology & Ontology : Theories and applications (TOTh'2014)*, Chambéry.

LEFRANÇOIS, M., GANDON, F., GIBOIN, A. et GUGERT, R. (2014b). Un éditeur de définitions lexicographiques formelles pour la théorie Sens-Texte. *In Actes de la 24e conférence d'Ingénierie des Connaissances (IC'2014)*, Clermont Ferrand. HAL.

LEFRANÇOIS, M., GUGERT, R., GANDON, F. et GIBOIN, A. (2013). Application of the Unit Graphs Framework to Lexicographic Definitions in the RELIEF project. *In Proceedings of the 6th International Conference on Meaning-Text Theory (MTT'2013)*, Prague, Czech Republic.

L'HOMME, M.-C. (2008). Le DiCoInfo : Méthodologie pour une nouvelle génération de dictionnaires spécialisés. *Traduire*, 217:78–103.

LUX-POGODALLA, V. et POLGUÈRE, A. (2011). Construction of a French Lexical Network : Methodological Issues. *In Proceedings of the International Workshop on Lexical Resources*, Ljubljana.

MANGEOT, M., SÉRASSET, G. et LAFOURCADE, M. (2003). Construction collaborative d'une base lexicale multilingue, le projet Papillon. *Traitement Automatique des Langues*.

MEL'ČUK, I. (1996). Lexical Functions : A Tool for the Description of Lexical Relations in a Lexicon. *In* WANNER, L., éditeur : *Lexical Functions in Lexicography and Natural Language Processing*, pages 37–102. Benjamins Academic Publishers, Amsterdam/Philadelphia.

MEL'ČUK, I. (2004a). Actants in semantics and syntax I : Actants in semantics. *Linguistics*, 42(1):1–66.

MEL'ČUK, I. (2004b). Actants in semantics and syntax II : Actants in syntax. *Linguistics*, 42(2):247–291.

MEL'ČUK, I. (2006). Explanatory combinatorial dictionary. *Open Problems in Linguistics and Lexicography*, pages 225–355.

MEL'ČUK, I., ARBATCHEWSKY-JUMARIE, N., ELNITSKY, L., IORDANSKAJA, L. et LESSARD, A. (1984). *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques I*. Les Presses de l'Université de Montréal, Montréal.

MEL'ČUK, I., ARBATCHEWSKY-JUMARIE, N., IORDANSKAJA, L., MANTHA, S. et POLGUÈRE, A. (1999). *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques IV*. Les Presses de l'Université de Montréal, Montréal, Canada.

MEL'ČUK, I., CLAS, A. et POLGUÈRE, A. (1995). *Introduction à la lexicologie explicative et combinatoire*. Duculot, Paris/Louvain-la-Neuve.

MEL'ČUK, I. A. (1997). *Vers une linguistique sens-texte*. Leçon inaugurale au collège de France.

MEL'ČUK, I. A. (2001). *Communicative organization in natural language : the semantic-communicative structure of sentences*. Studies in language companion. J. Benjamins, Amsterdam ; Philadelphia.

MEL'ČUK, I. A. (2003). Collocations dans le dictionnaire. *In* SZENDE, T., éditeur : *Les écarts culturels dans les Dictionnaires bilingues*, pages 19–64. Paris : Honoré Champion.

MEL'ČUK, I. A., ARBATCHEWSKY-JUMARIE, N., ELNITSKY, L. et LESSARD, A. (1988). *Dictionnaire explicatif et combinatoire du français contemporain*. Presses de l'Université de Montréal, Montréal, Canada.

MEL'ČUK, I. A., ARBATCHEWSKY-JUMARIE, N., ELNITSKY, L. et LESSARD, A. (1992). *Dictionnaire explicatif et combinatoire du francais contemporain*. Presses de l'Université de Montréal, Montréal, Canada.

MEL'ČUK, I. A. et WANNER, L. (2001). Towards a lexicographic approach to lexical transfer in machine translation (illustrated by the German–Russian language pair). *Machine Translation*.

MILLER, G. A. (1995). WordNet : a lexical database for English. *Communications of the ACM*, 38(11):39–41.

MINSKY, M. (1975). A Framework for Representing Knowledge. *In* WINSTON, P., éditeur : *The Psychology of Computer Vision*. McGraw-Hill.

MONK, A., DAVENPORT, L., HABER, J. et WRIGHT, P. (1993). *Improving your human-computer interface : A practical technique*. London : Prentice Hall.

MONNIN, A. (2013). *Vers une philosophie du Web : le Web comme devenir-artefact de la philosophie (entre URIs, tags, ontologie (s) et ressources)*. Thèse de doctorat, Université Paris 1 Panthéon-Sorbonne.

MOOT, R. et RETORÉ, C. (2012). *The Logic of Categorial Grammars - A deductive account of natural language syntax and semantics*. Lecture Notes in Computer Science. Springer-Verlag Berlin and Heidelberg GmbH & Co. K.

PIANTA, E., BENTIVOGLI, L. et GIRARDI, C. (2002). Developing an aligned multilingual database. *Proc. 1st Int'l Conference on Global WordNet*.

POLGUÈRE, A. (1990). *Structuration et mise en jeu procédurale d'un modèle linguistique déclaratif dans un cadre de génération de texte*. Thèse de doctorat, Département de linguistique, Université de Montréal.

POLGUÈRE, A. (2000a). Towards a theoretically-motivated general public dictionary of semantic derivations and collocations for French. *In ninth EURALEX International Congress (EURALEX'2000)*, pages 517–527, Stuttgart.

POLGUÈRE, A. (2000b). Une base de données lexicales du français et ses applications possibles en didactique. *Revue de Linguistique et de Didactique des Langues*, 21:75–97.

POLGUÈRE, A. (2003). Étiquetage sémantique des lexies dans la base de données DiCo. *Traitement automatique des langues*, 44(2):39–68.

POLGUÈRE, A. (2007). Lessons from the Lexique actif du français. *In Proceedings of the 3rd International Conference on Meaning-Text Theory (MTT'2007)*.

POLGUÈRE, A. (2009). Lexical systems : graph models of natural language lexicons. *Language resources and evaluation*, 43(1):41–55.

POLGUÈRE, A. (2011). Classification sémantique des lexies fondée sur le paraphrasage. *Cahiers de lexicologie*, 98:197–211.

PUSTEJOVSKY, J. (1991). The Generative Lexicon. *Computational Linguistics*, 17(4):409–441.

QUILLIAN, M. (1968). Semantic memory. *In* MINSKY, M. L., éditeur : *Semantic Information Processing*, chapitre 10, pages 227–270. MIT Press, Cambridge, MA.

ROUQUET, D. (2012). *Multilinguisation d'ontologies dans le cadre de la recherche d'information translingue dans des collections d'images accompagnées de textes spontanés*. Thèse de doctorat, Université Grenoble 1.

RUDOLPH, S. (2011). Foundations of description logics. *Reasoning Web. Semantic Technologies for the Web of Data*, pages 76–136.

SCHANK, R. C. et ABELSON, R. P. (1977). *Scripts, plans, goals, and understanding : An inquiry into human knowledge structures*. Artificial Intelligence. Hillsdale.

SÉRASSET, G. (1997a). Informatisation du dictionnaire Explicatif et Combinatoire. *Proc. TALN-97, Grenoble*.

SÉRASSET, G. (1997b). Le projet NADIA-DEC : vers un dictionnaire explicatif et combinatoire informatisé. *La mémoire des mots, Ve journées scientifiques du réseau LTT*, 97:149–159.

SEYE, O., FARON-ZUCKER, C., CORBY, O. et GAIGNARD, A. (2014). Publication, partage et réutilisation de règles sur le Web de données. *In Proc. 25èmes Journées francophones d'Ingénierie des Connaissances*, Clermont-Ferrand.

SINHA, M., REDDY, M. et BHATTACHARYYA, P. (2006). An approach towards construction and application of multilingual indo-wordnet. *In 3rd Global Wordnet Conference (GWC'06)*, Jeju Island, Korea.

SOWA, J. (1989). Using a lexicon of canonical graphs in a semantic interpreter. *In Relational models of the lexicon*, pages 113–137. Cambridge University Press New York, NY, USA.

SOWA, J. (1992). Logical structures in the lexicon. *Knowledge-Based Systems*, 5(3):173–182.

SOWA, J. F. (1984). *Conceptual structures : information processing in mind and machine*. System programming series. Addison-Wesley Pub., Reading, MA.

STAR, S. L. et GRIESEMER, J. R. (1989). Institutional Ecology, 'Translations' and Boundary Objects : Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3):387–420.

STEINLIN, J., KAHANE, S. et POLGUÈRE, A. (2005). Compiling a "classical" explanatory combinatorial lexicographic description into a relational database. *In Proceedings of the Second International Conference on the Meaning Text Theory (MTT'2005)*.

STEINLIN, J., KAHANE, S., POLGUÈRE, A. et EL GHALI, A. (2004). De l'article lexicographique à la modélisation objet du dictionnaire et des liens lexicaux. *Actes de EURALEX'2004*, pages 177–186.

TESNIÈRE, L. (1959). *Éléments de syntaxe structurale*. C. Klincksieck (Colombes, Impr. ITE).

VANIER, J., BASSAC, C., HENRY, P., MARLET, R. et RETORÉ, C. (2006). Toward a knowledge representation model dedicated to the semantic analysis of the sentence. Rapport technique, Inria.

VOSSEN, P. (1998). EuroWordNet : a multilingual database with lexical semantic networks. *Computational Linguistics*, 25(4).

ŽOLKOVSKIJ, A. et MEL'ČUK, I. A. (1967). O semantičeskom sinteze [on semantic synthesis (of texts)]. *Problemy kybernetiki [Problems of Cybernetics]*, 19:177–238.

WANNER, L. (2003). Definitions of Lexical Meanings : Some Reflections on Purpose and Structure. *In Proceedings of the First international conference on Meaning-Text Theory (MTT'2003)*, pages 16–28.

WIERZBICKA, A. (1996). *Semantics : Primes and Universals : Primes and Universals*. Oxford University Press.

Annexes

ANNEXE A

Preuves

Démonstration de la Proposition 6.1.1 . Sachant que les rôles f_1, f_2, et g sont fonctionnels, montrons que l'axiome 6.39 implique la règle de contraction 6.38. Le membre gauche de la règle n'a pas d'expression en LD. Réécrivons cette implication :

$$
\begin{aligned}
& (\forall x)\big[(\exists y,z)[A(x)\wedge f_1(x,y)\wedge f_2(y,z)\wedge g(x,z)]\rightarrow B(x)\big] && \text{(A.1)}\\
\Longleftrightarrow\ & (\forall x)\big[A(x)\wedge(\exists y,z)[f_1(x,y)\wedge f_2(y,z)\wedge g(x,z)]\rightarrow B(x)\big] && \text{(A.2)}\\
\Longleftrightarrow\ & (\forall x)\big[B(x)\vee\neg A(x)\vee\neg\big((\exists y,z)[f_1(x,y)\wedge f_2(y,z)\wedge g(x,z)]\big)\big] && \text{(A.3)}\\
\Longleftrightarrow\ & (\forall x)\big[A(x)\wedge\neg B(x)\rightarrow(\forall y,z)[\neg f_1(x,y)\vee\neg f_2(y,z)\vee\neg g(x,z)]\big] && \text{(A.4)}
\end{aligned}
$$

Le membre gauche de l'équation A.4 est équivalent à l'application du membre gauche de l'axiome 6.39 à x :

$$\big(A\sqcap\neg B\big)(x)\leftrightarrow A(x)\wedge\neg B(x) \tag{A.5}$$

Introduisons un concept auxiliaire R. Montrons que l'application du membre droit de l'axiome 6.39 à x est équivalent au membre droit de l'équation A.4.

$$
\begin{aligned}
& \Big((\forall g.R)\sqcap\big(\forall f_1.(\forall f_2.\neg R)\big)\Big)(x) && \text{(A.6)}\\
\Longleftrightarrow\ & \big((\forall z)[g(x,z)\rightarrow R(z)]\big)\wedge\Big((\forall y)\Big[f_1(x,y)\rightarrow\big((\forall z)\big[f_2(y,z)\rightarrow\neg R(z)\big]\big)\Big]\Big) && \text{(A.7)}\\
\Longleftrightarrow\ & \big((\forall z)[R(z)\vee\neg g(x,y)]\big)\wedge\Big((\forall y)\Big[\big((\forall z)\big[\neg R(z)\vee\neg f_2(y,z)\big]\big)\vee\neg f_1(x,y)\Big]\Big) && \text{(A.8)}\\
\Longleftrightarrow\ & \big((\forall z)[R(z)\vee\neg g(x,y)]\big)\wedge\Big((\forall y,z)\Big[\neg R(z)\vee\neg f_2(y,z)\vee\neg f_1(x,y)\Big]\Big) && \text{(A.9)}\\
\Longleftrightarrow\ & (\forall z)\Big[\big(R(z)\vee\neg g(x,y)\big)\wedge\Big(\neg R(z)\vee\big((\forall y)\big[\neg f_2(y,z)\vee\neg f_1(x,y)\big]\big)\Big)\Big] && \text{(A.10)}\\
\Longrightarrow\ & (\forall z)\big[\neg g(x,z)\vee\big((\forall y)\big[\neg f_2(y,z)\vee\neg f_1(x,y)\big]\big)\big] && \text{(A.11)}\\
\Longrightarrow\ & (\forall y,z)\big[\neg g(x,z)\vee\neg f_1(x,y)\vee\neg f_2(y,z)\big] && \text{(A.12)}
\end{aligned}
$$

Ainsi l'axiome 6.39 implique la règle de contraction 6.38 de $^{/}\mathrm{B}^{\backslash}$. □

Démonstration de la Proposition 6.1.2 . Sachant que les rôles f_1, f_2, et g sont fonctionnels, et ayant posé les axiomes 6.45 et 6.46, montrons que l'axiome 6.47 implique les règles d'expansion 6.40 et 6.41.

$$\begin{cases} & \big((\forall x,y,z)\big[f_1(x,y)\wedge f_2(y,z)\rightarrow h(x,z)\big]\big) \\ \wedge & \big((\forall x,z)\big[g(x,z)\rightarrow h(x,z)\big]\big) \\ \wedge & \big(B\sqsubseteq A\sqcap(\leq 1.h)\big) \\ \wedge & B(x)\wedge f_1(x,y)\wedge f_2(y,z) \end{cases} \tag{A.13}$$

$$\Longrightarrow \begin{cases} & \big((\forall x,y)\big[f_1(x,y)\wedge f_2(y,z)\rightarrow h(x,z)\big]\big) \\ \wedge & \big((\forall x)[B(x)\rightarrow A(x)]\big) \\ \wedge & B(x)\wedge f_1(x,y)\wedge f_2(y,z) \end{cases} \tag{A.14}$$

$$\Longrightarrow A(x)\wedge h(x,z) \tag{A.15}$$

Donc les axiomes de LD impliquent la règle 6.40.

$$\begin{cases} & \big((\forall x,y,z)\,[f_1(x,y)\wedge f_2(y,z)\rightarrow h(x,z)]\big) \\ \wedge & \big((\forall x,z)\,\big[g(x,z)\rightarrow h(x,z)\big]\big) \\ \wedge & \big(B\sqsubseteq A\sqcap(\leq 1.h)\big) \\ \wedge & B(x)\wedge f_1(x,y)\wedge f_2(y,z_1)\wedge g(x,z_2) \end{cases} \tag{A.16}$$

$$\Longrightarrow h(x,z_1)\wedge h(x,z_2)\wedge A(x)\wedge\big((\forall x,z_1,z_2)[h(x,z_1)\wedge h(x,z_2)\rightarrow z_1=z_2]\big) \tag{A.17}$$

$$\Longrightarrow A(x)\wedge z_1=z_2 \tag{A.18}$$

Donc les axiomes de LD impliquent la règle 6.41. □

Démonstration de la Proposition 6.1.3 . Sachant que les rôles f_1 et f_2 sont fonctionnels, montrons que l'axiome 6.51 implique la règle de contraction 6.50. Le membre gauche de la règle n'a pas d'expression en LD. Réécrivons cette implication :

$$(\forall x)\,[(\exists y)\,[A(x)\wedge f_1(x,y)\wedge f_2(y,x)]\rightarrow B(x)] \tag{A.19}$$

$$\Longleftrightarrow (\forall x)\,[A(x)\wedge((\exists y)\,[\wedge f_1(x,y)\wedge f_2(y,x)])\rightarrow B(x)] \tag{A.20}$$

$$\Longleftrightarrow (\forall x)\,\big[\big(B(x)\wedge\neg A(x)\big)\vee\neg\big((\exists y)\,[f_1(x,y)\wedge f_2(y,x)]\,\big)\big] \tag{A.21}$$

$$\Longleftrightarrow (\forall x)\,[A(x)\wedge\neg B(x)\rightarrow(\forall y)\,[\neg f_1(x,y)\vee\neg f_2(y,x)]] \tag{A.22}$$

L'application à x du membre gauche de l'axiome de LD est équivalent au membre gauche de la règle de contraction réécrite.

$$\big(A\sqcap\neg B\big)(x)\leftrightarrow A(x)\wedge\neg B(x) \tag{A.23}$$

Introduisons un concept auxiliaire R. Montrons que l'application du membre droit de l'axiome

de LD implique le membre droit de la règle de contraction réécrite.

$$\Big(R \sqcap \big(\forall f_1.(\forall f_2.\neg R)\big)\Big)(x) \tag{A.24}$$
$$\iff R(x) \wedge \Big((\forall y)\left[f_1(x,y) \rightarrow \big((\forall z)\left[f_2(y,z) \rightarrow \neg R(z)\right]\big)\right]\Big) \tag{A.25}$$
$$\iff R(x) \wedge \Big((\forall y)\left[\big((\forall z)\left[\neg R(z) \vee \neg f_2(y,z)\right]\big) \vee \neg f_1(x,y)\right]\Big) \tag{A.26}$$
$$\iff R(x) \wedge \Big((\forall y,z)\left[\neg R(z) \vee \neg f_2(y,z) \vee \neg f_1(x,y)\right]\Big) \tag{A.27}$$
$$\implies R(x) \wedge \Big(\neg R(x) \vee \big((\forall y)\left[\neg f_2(y,x) \vee \neg f_1(x,y)\right]\big)\Big) \tag{A.28}$$
$$\implies (\forall y)\left[R(x) \wedge \big(\neg f_2(y,x) \vee \neg f_1(x,y)\big)\right] \tag{A.29}$$
$$\implies (\forall y)\left[\neg f_1(x,y) \vee \neg f_2(y,x)\right] \tag{A.30}$$

Ainsi l'axiome 6.51 implique la règle de contraction 6.50. □

Démonstration de la Proposition 6.1.4 . Sachant que les rôles f_1 et f_2 sont fonctionnels, et ayant posé les axiomes 6.56 et 6.57, montrons que l'axiome 6.58 implique les règles d'expansion 6.52 et 6.53.

$$\begin{cases} & \big((\forall x,y,z)\left[f_1(x,y) \wedge f_2(y,z) \rightarrow g(x,z)\right]\big) \\ \wedge & \big((\forall x)\left[B(x) \rightarrow g(x,x)\right]\big) \\ \wedge & \big(B \sqsubseteq A \sqcap (\leq 1.g)\big) \\ \wedge & B(x) \wedge f_1(x,y) \wedge f_2(y,x) \end{cases} \tag{A.31}$$
$$\implies \begin{cases} & \big((\forall x)\left[B(x) \rightarrow g(x,x)\right]\big) \\ \wedge & \big((\forall x)\left[B(x) \rightarrow A(x)\right]\big) \\ \wedge & B(x) \end{cases} \tag{A.32}$$
$$\implies A(x) \wedge g(x,x) \tag{A.33}$$

Donc les axiomes de LD impliquent la règle 6.52.

$$\begin{cases} & \big((\forall x,y,z)\left[f_1(x,y) \wedge f_2(y,z) \rightarrow g(x,z)\right]\big) \\ \wedge & \big((\forall x)\left[B(x) \rightarrow g(x,x)\right]\big) \\ \wedge & \big(B \sqsubseteq A \sqcap (\leq 1.g)\big) \\ \wedge & B(x) \wedge f_1(x,y) \wedge f_2(y,x) \end{cases} \tag{A.34}$$
$$\implies g(x,x) \wedge g(x,z) \wedge A(x) \wedge \big(\forall x,z_1,z_2,g(x,z_1) \wedge g(x,z_2) \rightarrow z_1 = z_2\big) \tag{A.35}$$
$$\implies A(x) \wedge (x = z) \tag{A.36}$$

Donc les axiomes de LD impliquent la règle 6.41. □

Démonstration de la Proposition 7.1.1 . Par construction, chacun des points suivants est vrai :

- $\boldsymbol{C_T}$ contient C_A, donc le préordre $\boldsymbol{C_T^*}$ contient l'ensemble des comparaisons déclarées ;
- $\boldsymbol{C_T}$ contient $C_\top$, donc $\forall t \in \boldsymbol{T}, (\top, t) \in \boldsymbol{C_T^*}$, et $\top$ sont des éléments maximaux de $(\boldsymbol{T}, \boldsymbol{C_T^*})$;
- $\boldsymbol{C_T}$ contient $C_\bot$, donc $\forall t \in \boldsymbol{T}, (t, \bot) \in \boldsymbol{C_T^*}$, et $\bot$ est un élément minimal de $(\boldsymbol{T}, \boldsymbol{C_T^*})$;
- $\boldsymbol{C_T}$ contient $C_{\boldsymbol{\Gamma_1}}$, donc $\forall s \in \boldsymbol{S_\mathscr{T}}, (\boldsymbol{\gamma}(s), \boldsymbol{\gamma_1}(s)) \in \boldsymbol{C_T^*}$, i.e., pour tout SRelA s, la racine de PosAObl s est plus spécifique que la racine de PosA s ;

– $\boldsymbol{C_T}$ contient $C_{\boldsymbol{\Gamma_0}}$, donc $\forall s \in \boldsymbol{S}_{\mathscr{T}}, (\boldsymbol{\gamma}(s), \boldsymbol{\gamma_0}(s)) \in \boldsymbol{C_T^*}$, i.e., pour tout SRelA s, la racine de PosAInt s est plus spécifique que la racine de PosA s.

Finalement, $\boldsymbol{C_T}$ introduit aucune comparaison superflue, donc $\boldsymbol{C_T^*}$ est effectivement le plus petit préordre tel que les points de la définition 31 sont vrai, i.e.,

$$\forall t, t' \in \boldsymbol{T}, t' \lesssim t \iff (t, t') \in \boldsymbol{C_T^*}$$

□

Démonstration de la Proposition 7.1.2 . Soit $s \in \boldsymbol{S}_{\mathscr{T}}$.
Preuve de l'égalité 7.1 :
$\subseteq$: Soit $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha}(t)\}$. $s \in \boldsymbol{\alpha}(t)$, par la définition 34, $t \lesssim \boldsymbol{\gamma}(s)$, donc $t \in {\downarrow}\boldsymbol{\gamma}(s)$.
$\supseteq$: Soit $t \in {\downarrow}\boldsymbol{\gamma}(s)$. $t \lesssim \boldsymbol{\gamma}(s)$, et par la définition 34, $s \in \boldsymbol{\alpha}(t)$, donc $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha}(t)\}$.
Preuve de l'égalité 7.2 :
$\subseteq$: Soit $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_1}(t)\}$. $s \in \boldsymbol{\alpha_1}(t)$, par la définition 35, $t \lesssim \boldsymbol{\gamma_1}(s)$, donc $t \in {\downarrow}\boldsymbol{\gamma_1}(s)$.
$\supseteq$: Soit $t \in {\downarrow}\boldsymbol{\gamma_1}(s)$. $t \lesssim \boldsymbol{\gamma_1}(s)$, et par la définition 35, $s \in \boldsymbol{\alpha_1}(t)$, donc $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_1}(t)\}$.
Preuve de l'égalité 7.3 :
$\subseteq$: Soit $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_0}(t)\}$. $s \in \boldsymbol{\alpha_0}(t)$, par la définition 36, $t \lesssim \boldsymbol{\gamma_0}(s)$, donc $t \in {\downarrow}\boldsymbol{\gamma}(s)$.
$\supseteq$: Soit $t \in {\downarrow}\boldsymbol{\gamma_0}(s)$. $t \lesssim \boldsymbol{\gamma}(s)$, et par la définition 36, $s \in \boldsymbol{\alpha_0}(t)$, donc $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_0}(t)\}$. □

Démonstration de la Proposition 7.1.3 . Soit $t_x, t_y \in \boldsymbol{T}$ tel que $t_x \lesssim t_y$.
Soit $s \in \boldsymbol{\alpha}(t_y)$. La définition 34 implique que $t_y \lesssim \boldsymbol{\gamma}(s)$. Donc $t_x \lesssim t_y \lesssim \boldsymbol{\gamma}(s)$ et $s \in \boldsymbol{\alpha}(t_x)$.
Soit $s \in \boldsymbol{\alpha_1}(t_y)$. La définition 35 implique que $t_y \lesssim \boldsymbol{\gamma_1}(s)$. Donc $t_x \lesssim t_y \lesssim \boldsymbol{\gamma_1}(s)$ et $s \in \boldsymbol{\alpha_1}(t_x)$.
Soit $s \in \boldsymbol{\alpha_0}(t_y)$. La définition 36 implique que $t_y \lesssim \boldsymbol{\gamma_0}(s)$. Donc $t_x \lesssim t_y \lesssim \boldsymbol{\gamma_0}(s)$ et $s \in \boldsymbol{\alpha_0}(t_x)$. □

Démonstration de la Proposition 7.1.4 . Soit $t \in \boldsymbol{T}$, et $s \in \boldsymbol{\alpha_1}(t)$.
Par la définition 35, $t \lesssim \boldsymbol{\gamma_1}(s)$.
Puisque $\boldsymbol{\gamma_1}(s) \lesssim \boldsymbol{\gamma}(s)$ par la définition 31, nous savons que $t \lesssim \boldsymbol{\gamma}(s)$.
Donc par la définition 34, $s \in \boldsymbol{\alpha}(t)$ et $\boldsymbol{\alpha_1}(t) \subseteq \boldsymbol{\alpha}(t)$.
La preuve pour $\boldsymbol{\alpha_0}$ est la même en remplaçant $\boldsymbol{\gamma_1}$ par $\boldsymbol{\gamma_0}$, la définition 35 par la définition 36, et $\boldsymbol{\alpha_1}$ par $\boldsymbol{\alpha_0}$. □

Démonstration de la Proposition 7.1.5 . Puisque $\bot$ est un élément minimal de $\boldsymbol{T}$, alors $\forall s \in \boldsymbol{S}_{\mathscr{T}}, \bot \lesssim \boldsymbol{\gamma}(s)$. Donc $s \in \boldsymbol{\alpha}(\bot)$. Ainsi $\boldsymbol{\alpha}(\bot) = \boldsymbol{S}_{\mathscr{T}}$. Maintenant, pour tout $t \in \bot^{\sim}$ nous savons que $t \lesssim \bot$, et en utilisant la proposition 7.1.3, $\boldsymbol{S}_{\mathscr{T}} \subseteq \boldsymbol{\alpha}(t)$. Donc $\boldsymbol{\alpha}(t) = \boldsymbol{S}_{\mathscr{T}}$.
La preuve pour $\boldsymbol{\alpha_1}$ (resp. $\boldsymbol{\alpha_0}$) est la même en remplaçant $\boldsymbol{\gamma}$ par $\boldsymbol{\gamma_1}$ (resp. $\boldsymbol{\gamma_0}$), et $\boldsymbol{\alpha}$ par $\boldsymbol{\alpha_1}$ (resp. $\boldsymbol{\alpha_0}$). □

Démonstration de la Proposition 7.1.6 . Soit $s \in \boldsymbol{S}_{\mathscr{T}}$.
$\subseteq$: Soit $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_?}(t)\}$. $s \in \boldsymbol{\alpha_?}(t)$.
Par la définition 37, $s \in \boldsymbol{\alpha}(t) - \boldsymbol{\alpha_1}(t) - \boldsymbol{\alpha_0}(t)$.
Par les définitions 34, 35 et 36, $s \lesssim \boldsymbol{\gamma}(s)$ et $t \not\lesssim \boldsymbol{\gamma_1}(s)$ et $t \not\lesssim \boldsymbol{\gamma_0}(s)$.
Donc $t \in {\downarrow}\boldsymbol{\gamma}(s) - {\downarrow}\boldsymbol{\gamma_1}(s) - {\downarrow}\boldsymbol{\gamma_0}(s)$.
$\supseteq$: Soit $t \in {\downarrow}\boldsymbol{\gamma}(s) - {\downarrow}\boldsymbol{\gamma_1}(s) - {\downarrow}\boldsymbol{\gamma_0}(s)$.
$s \lesssim \boldsymbol{\gamma}(s)$ et $t \not\lesssim \boldsymbol{\gamma_1}(s)$ et $t \not\lesssim \boldsymbol{\gamma_0}(s)$.
Par la définition 34, 35 et 36, $s \in \boldsymbol{\alpha}(t)$ et $s \notin \boldsymbol{\alpha_1}(t)$ et $s \notin \boldsymbol{\alpha_0}(t)$.
Donc $s \in \boldsymbol{\alpha_?}(t)$, et $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha_?}(t)\}$. □

Démonstration de la Proposition 7.1.7 . Par la définition 34, $\boldsymbol{\alpha_?}(t) = \boldsymbol{\alpha}(t) - \boldsymbol{\alpha_1}(t) - \boldsymbol{\alpha_0}(t)$.
De plus, par la proposition 7.1.3, $\boldsymbol{\alpha}(t') = \boldsymbol{\alpha}(t)$, $\boldsymbol{\alpha_1}(t') = \boldsymbol{\alpha_1}(t)$ et $\boldsymbol{\alpha_0}(t') = \boldsymbol{\alpha_0}(t)$.
Donc $\boldsymbol{\alpha}(t') = \boldsymbol{\alpha}(t)$. □

Démonstration de la Proposition 7.1.8 . Soit $t \in \bot^{\sim}$.
Par la définition 37 et la proposition 7.1.5,
$\boldsymbol{\alpha_?}(t) = \boldsymbol{S}_{\mathscr{T}} - \boldsymbol{S}_{\mathscr{T}} - \boldsymbol{S}_{\mathscr{T}} = \varnothing$. □

Démonstration de la Proposition 7.1.9 . Par la proposition 7.1.5, $\forall t \in \bot^{\sim}, \boldsymbol{\alpha_1}(t) = \boldsymbol{\alpha_0}(t) = \boldsymbol{S}_{\mathscr{T}}$.
Ainsi si $\boldsymbol{S}_{\mathscr{T}} \neq \varnothing$, alors $\boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing$.
Ainsi $\bot^{\sim} \subseteq \{t \in \boldsymbol{T} \mid \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing\}$ □

Démonstration de la Proposition 7.2.1 . Soit $t \in \boldsymbol{T}$.
1) $\boldsymbol{\alpha}^{\cap}(\{t\}) = \bigcup_{t \in \{t\}} \boldsymbol{\alpha}(t) = \boldsymbol{\alpha}(t)$;
2) $\boldsymbol{\alpha_1}^{\cap}(\{t\}) = \bigcup_{t \in \{t\}} \boldsymbol{\alpha_1}(t) = \boldsymbol{\alpha_1}(t)$;
3) $\boldsymbol{\alpha_0}^{\cap}(\{t\}) = \bigcup_{t \in \{t\}} \boldsymbol{\alpha_0}(t) = \boldsymbol{\alpha_0}(t)$;
4) $\boldsymbol{\alpha_?}^{\cap}(\{t\}) = \boldsymbol{\alpha}^{\cap}(\{t\}) - \boldsymbol{\alpha_1}^{\cap}(\{t\}) - \boldsymbol{\alpha_0}^{\cap}(\{t\}) = \boldsymbol{\alpha}(t) - \boldsymbol{\alpha_1}(t) - \boldsymbol{\alpha_0}(t) = \boldsymbol{\alpha_?}(t)$.
5) soit $s \in \boldsymbol{\alpha}(t)$. $\{t \in \{t\} \mid s \in \boldsymbol{\alpha}(t)\} = \{t\}$, donc $\boldsymbol{\varsigma}^{\cap}_{\{t\}}(s) = \bigcup_{t \in \{t\}} \boldsymbol{\varsigma}_t(s) = \boldsymbol{\varsigma}_t(s)$. □

Démonstration de la Proposition 7.2.2 . Soit $t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\cap}$ tel que $t_x^{\cap} \subseteq t_y^{\cap}$. Alors $\forall t_x \in t_x^{\cap}, t_x \in t_y^{\cap}$.
Ainsi $\forall t_x \in t_x^{\cap}, \exists t_y (= t_x) \in t_y^{\cap} : t_y \lesssim t_x$, donc par la définition 47, $t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}$. □

Démonstration de la Proposition 7.2.3 . $\varnothing$:
Soit $t^{\cap} \in \boldsymbol{T}^{\cap}$. $\varnothing \subseteq t^{\cap}$, donc par la proposition 7.2.2 : $t^{\cap} \overset{\cap}{\lesssim} \varnothing$.
Puisque $\overset{\cap}{\lesssim}$ contient l'extension naturelle du préordre $\lesssim$, alors $\varnothing$ est un élément maximal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.
$\boldsymbol{T}$:
Soit $t^{\cap} \in \boldsymbol{T}^{\cap}$. $t^{\cap} \subseteq \boldsymbol{T}$, donc par la proposition 7.2.2 : $\boldsymbol{T} \overset{\cap}{\lesssim} t^{\cap}$.
Puisque $\overset{\cap}{\lesssim}$ contient l'extension naturelle du préordre $\lesssim$, alors $\boldsymbol{T}$ est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.
□

Démonstration de la Proposition 7.2.4 . Soit $t^{\cap} \subseteq \top^{\sim}$.
Si $t^{\cap} = \varnothing$, alors nous savons par la proposition 7.2.3 que $t^{\cap}$ est un élément maximal de $(t^{\cap}, \overset{\cap}{\lesssim})$.
Si $t^{\cap} \neq \varnothing$, alors soit $t_x^{\cap} \in \boldsymbol{T}^{\cap}$. Pour tout $t \in t^{\cap} \setminus \varnothing$, puisque t est un élément maximal de $\boldsymbol{T}$, alors pour tout $t_x \in t_x^{\cap}$, $t_x \lesssim t$. Donc $t_x^{\cap} \overset{\cap}{\lesssim} t^{\cap}$.
En particulier, $\top^{\cap} \overset{\cap}{\lesssim} t^{\cap}$. Maintenant puisque par la définition $\top^{\cap}$ est un élément maximal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, alors $\varnothing \overset{\cap}{\lesssim} \top^{\cap}$. Ainsi $\varnothing \overset{\cap}{\lesssim} t^{\cap}$.
Donc $t^{\cap}$ est un élément maximal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$. □

Démonstration de la Proposition 7.2.5 . Puisque $\top^{\cap}$ est un élément maximal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ par la proposition 7.2.4, alors l'ensemble $\{t^{\cap} \mid t^{\cap} \overset{\cap}{\cong} \top^{\cap}\}$ est l'ensemble des éléments maximaux de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, i.e., l'ensemble des TUC universels $\top^{\sqcap}$. □

Démonstration de la Proposition 7.2.6 . Prouvons chaque item un par un :
1 : Par la proposition 7.2.3 $\boldsymbol{T}$ est un élément minimal de $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, donc il est absurde.
2 : Nous savons par la définition 39, que $\bot^{\sim} \subseteq A^{\sim}$.
De plus, $\bot^{\cap} = \{\bot\} \subseteq \bot^{\sim}$. Donc il suffit de prouver que tout sous-ensemble non-vide de $A^{\sim}$ est

absurde. Soit $t^\cap \subseteq A^\sim \setminus \varnothing$ un tel sous-ensemble, et $t \in t^\cap$.
Par la définition 39, soit a) $t \in \bot^\sim$, soit b) $\boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing$.
a) Si $t \in \bot^\sim$: Soit $t_x^\cap \in \boldsymbol{T}^\cap$. $\forall t_x \in t_x^\cap, t \lesssim t_x$.
Donc $\forall t_x \in t_x^\cap, \exists t'(=t) \in t^\cap, t' \lesssim t_x$.
Donc par la définition 47, $t^\cap \overset{\cap}{\lesssim} t_x^\cap$, et $t^\cap$ est un élément minimal de $\boldsymbol{T}^\cap$.
b) Si $\boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing$:
Soit $s \in \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t)$. Par les définitions 35 et 36, $t \lesssim \boldsymbol{\gamma_1}(s)$ et $t \lesssim \boldsymbol{\gamma_0}(s)$.
Soit $t_x^\cap = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$. Alors $\forall t_x \in t_x^\cap, \exists t'(=t) \in t^\cap, t' \lesssim t_x$.
Donc $t^\cap \overset{\cap}{\lesssim} \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$, et puisque par la définition 47, $\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$ est un élément minimal de $\boldsymbol{T}^\cap$, alors $t^\cap$ l'est également.
Dans les deux cas, a) et b), $t^\cap$ est un élément minimal de $\boldsymbol{T}^\cap$ et est donc absurde.
3 : Soit $t^\cap \in \boldsymbol{T}^\cap$ et supposons que $\exists t \in t^\cap, t \lesssim \bot$. Alors par la définition 47 $t^\cap \overset{\cap}{\lesssim} \bot^\cap$. Puisque $\bot^\cap$ est absurde par l'item 2, alors $t^\cap \in \bot^\sqcap$ est également absurde.
4 : Soit $t^\cap \in \boldsymbol{T}^\cap$ et supposons que $\exists t_a^\cap \in \bot^\sqcap$ tel que $\forall t_a \in t_a^\cap, \exists t \in t^\cap, t \lesssim t_a$. Alors par la définition 47 $t^\cap \overset{\cap}{\lesssim} t_a^\cap$. Puisque $t_a^\cap$ est un élément minimal de $\boldsymbol{T}^\cap$ par la définition 47, alors $t^\cap$ l'est également. Ainsi $t^\cap$ est absurde.
5 : Soit $t^\cap \in \boldsymbol{T}^\cap$ tel que $\boldsymbol{\alpha_1^\cap}(t^\cap) \cap \boldsymbol{\alpha_0^\cap}(t^\cap) \neq \varnothing$. Soit $s \in \boldsymbol{\alpha_1^\cap}(t^\cap) \cap \boldsymbol{\alpha_0^\cap}(t^\cap)$.
Par les définitions 43 et 44, $\exists t_1 \in t^\cap : s \in \boldsymbol{\alpha_1}(t_1)$, et $\exists t_0 \in t^\cap : s \in \boldsymbol{\alpha_0}(t_0)$.
Par les définitions 35 et 36, $\exists t_1 \in t^\cap : t_1 \lesssim \boldsymbol{\gamma_1}(s)$, et $\exists t_0 \in t^\cap : t_0 \lesssim \boldsymbol{\gamma_0}(s)$.
Donc $\forall t_x \in \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}, \exists t \in t^\cap : t \lesssim t_x$.
Donc $t^\cap \overset{\cap}{\lesssim} \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$, et par la définition 47, $\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$ est un élément minimal de $\boldsymbol{T}^\cap$, alors $t^\cap$ est absurde.
6 : Supposons que $\exists s \in \boldsymbol{S}_\mathscr{T}$ tel que $\boldsymbol{\varsigma}_{t^\cap}^\cap(s) \in \bot^\sqcap$. Alors par la définition 49 $\boldsymbol{\varsigma}_{t^\cap}^\cap(s)$ est un élément minimal de $\boldsymbol{T}^\cap$, et par la définition 47 $t^\cap$ est un élément minimal de $\boldsymbol{T}^\cap$, et est absurde. □

Démonstration de la Proposition 7.2.7 . Puisque $\bot^\cap$ est un élément minimal de $(\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$ par la proposition 7.2.6, alors l'ensemble $\{t^\cap \mid t^\cap \mathrel{\hat{\simeq}} \bot^\cap\}$ est l'ensemble des éléments minimaux de $(\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$, i.e., l'ensemble des TUC absurdes $\bot^\sqcap$. □

Démonstration du Lemme 7.3.1 . Il est bien connu que $\boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$ définit un préordre sur $2^{\boldsymbol{T}}$ (i.e., il est réflexif et transitif), rappelons comment ce résultat est obtenu :
Réflexivité : Soit $t^\cap \in \boldsymbol{T}^\cap$.
$\forall t \in t^\cap, \exists t'(=t) \in t^\cap : t' \lesssim t$.
Donc $(t^\cap, t^\cap) \in \boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$.
Transitivité : Soit $t_x^\cap, t_y^\cap, t_z^\cap \in \boldsymbol{T}^\cap$ tel que $(t_z^\cap, t_y^\cap) \in \boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$, et $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$.
$\forall t_z \in t_z^\cap, \exists t_y \in t_y^\cap : t_y \lesssim t_z$, et $\forall t_y' \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y'$.
Soit $t_y' = t_y$, alors $\forall t_z \in t_z^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_z$,
et donc $(t_z^\cap, t_x^\cap) \in \boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$.

Maintenant, à partir de ce résultat et du processus de construction itérative de $\boldsymbol{C}^\cap$, on peut prouver la réflexivité et la transitivité de $\boldsymbol{C}^\cap$:
Puisque $\boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap \subseteq \boldsymbol{C}^\cap$ et $\boldsymbol{C}_{\overset{\cap}{\lesssim}}^\cap$ sont réflexifs, alors $\boldsymbol{C}^\cap$ l'est également.
La transitivité est obtenue de $\boldsymbol{C}_+^\cap$ dans le processus de construction itérative de $\boldsymbol{C}^\cap$. □

Démonstration du Lemme 7.3.2 . Montrons séquentiellement chacune des équations de la définition 47 :

Extension de $\lesssim$:
Pour tout $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$, si $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$, alors $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\lesssim}^\cap \subseteq \boldsymbol{C}^\cap$.
Donc $\forall t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap, \left(\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y \implies (t_y^\cap, t_x^\cap) \in \boldsymbol{C}^\cap\right)$
et $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$ contient l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$.

$\varnothing$ est un TUC maximal :
Prouvons ce résultat intermédiaire.
Soit $t^\cap \in \boldsymbol{T}^\cap$. $\forall t' \in \varnothing$, (i.e., none), $\exists t \in t^\cap, t \lesssim t'$. Ainsi puisque $\boldsymbol{C}^\cap$ contient l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$, alors $\forall t^\cap \in \boldsymbol{T}^\cap, (\varnothing, t^\cap) \in \boldsymbol{C}^\cap$
Donc $\varnothing$ est un élément maximal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$.

$\top^\cap$ est un TUC maximal :
$\varnothing$ est un élément maximal pour $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$. Puisque de plus $(\top^\cap, \varnothing) \in \boldsymbol{C}_\top^\cap \subseteq \boldsymbol{C}^\cap$, alors
$\forall t^\cap \in \boldsymbol{T}^\cap, (\top^\cap, t^\cap) \in \boldsymbol{C}^\cap$
Donc $\top^\cap$ est un élément maximal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$.

$\bot^\cap$ est un TUC minimal :
Prouvons ce résultat intermédiaire : Soit $t_x^\cap \in \boldsymbol{T}^\cap$. $\forall t_x \in t_x^\cap, \bot \lesssim t_x$.
Donc $\forall t_x \in t_x^\cap, \exists t (= \bot) \in \bot^\cap, t \lesssim t_x$.
Ainsi puisque $\boldsymbol{C}^\cap$ contient l'extension naturelle du préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$, $\forall t^\cap \in \boldsymbol{T}^\cap, (t^\cap, \bot^\cap) \in \boldsymbol{C}^\cap$
Et $\bot^\cap$ est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$.

TUC déclarés absurdes :
Pour tout $t^\cap \in \bot_A^\sqcap$, $(\bot^\cap, t^\cap) \in \boldsymbol{C}_\bot^\cap \subseteq \boldsymbol{C}^\cap$. Puisque $\bot^\cap$ est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$, alors $t^\cap$ l'est également.
Donc $\forall t_a^\cap \in \bot_A^\sqcap, \forall t^\cap \in \boldsymbol{T}^\cap, (t^\cap, t_a^\cap) \in \boldsymbol{C}^\cap$
Et tout TUC déclaré absurde est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$.

les racines de PosAObl et de PosAInt d'un même SRelA sont incompatibles :
Pour tout $s \in \boldsymbol{S}_\mathscr{T}$, $(\bot^\cap, \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}) \in \boldsymbol{C}_\boldsymbol{\Gamma}^\cap \subseteq \boldsymbol{C}^\cap$. Puisque $\bot^\cap$ est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$, alors $\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$ l'est également.
$\forall s \in \boldsymbol{S}_\mathscr{T}, \forall t^\cap \in \boldsymbol{T}^\cap, (t^\cap, \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}) \in \boldsymbol{C}^\cap$ pour tout SRelA, le TUC composé de ses racines de PosAObl et de PosAInt est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$.

Signatures absurdes :
Supposons $\exists s \in \boldsymbol{\alpha_1^\cap}(t^\cap), \left(\forall t_x^\cap \in \boldsymbol{T}^\cap, (t_x^\cap, \boldsymbol{\varsigma}_{t^\cap}^\cap(s)) \in \boldsymbol{C}^\cap\right)$
Alors $(\bot^\cap, \boldsymbol{\varsigma}_{t^\cap}^\cap(s)) \in \boldsymbol{C}^\cap$, et il existe $n \in \mathbb{N}$ tel que $(\bot^\cap, \boldsymbol{\varsigma}_{t^\cap}^\cap(s)) \in \boldsymbol{C}_n^\cap$.
Ainsi pour $n+1$, $(\bot^\cap, t^\cap) \in \boldsymbol{C}_{\boldsymbol{\varsigma}\,n+1}^\cap \subseteq \boldsymbol{C}^\cap$. Et puisque $\bot^\cap$ est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap)$, alors $t^\cap$ l'est également.
Ainsi si la signature d'un TUC pour une PosAObl données est un élément minimal de $\boldsymbol{T}^\cap$, alors ce TUC est un élément minimal de $\boldsymbol{T}^\cap$, i.e., □

Démonstration of Théorème 7.3.3 . Maintenant que nous savons que $\boldsymbol{C}^\cap$ est un préordre (i.e., réflexif et transitif) tel que toutes les conditions de la définition 47 sont satisfaites, il suffit de montrer que $\boldsymbol{C}^\cap$ est le plus parti de ces préordres. Nous devons donc seulement montrer que $(\boldsymbol{T}^\cap, \boldsymbol{C}^\cap) \subseteq (\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$:

- $(\boldsymbol{T}^\cap, \boldsymbol{C}_{\lesssim}^\cap) \subseteq (\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$ car $\boldsymbol{C}_{\lesssim}^\cap$ est l'expression de l'extension d'un préordre $\lesssim$ sur l'ensemble des parties de $\boldsymbol{T}$;
- $(\boldsymbol{T}^\cap, \boldsymbol{C}_\top^\cap) \subseteq (\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$ car il assure seulement que $\top^\cap$ est un élément maximal de $(\boldsymbol{T}^\cap, \boldsymbol{C}_\top^\cap)$.
- $(\boldsymbol{T}^\cap, \boldsymbol{C}_\bot^\cap) \subseteq (\boldsymbol{T}^\cap, \overset{\cap}{\lesssim})$ car il assure seulement que tout TUC déclaré absurde est un élément minimal de $(\boldsymbol{T}^\cap, \boldsymbol{C}_\top^\cap)$.

– $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{\boldsymbol{\Gamma}}) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ car il assure seulement que tout couple $\{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$ est un élément minimal de $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{\top})$.

Ainsi pour $i = 0$, $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_0) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$.

Maintenant si $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_i) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, alors

– $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{\boldsymbol{\varsigma}\, i}) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ car il assure simplement que si la signature d'un TUC pour une PosAObl donnée est minimale, alors ce TUC est un élément minimal de $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{\top})$.

– $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{+i}) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ car $(\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$ est un préordre.

Ainsi $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_{i+1}) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$

Ainsi par récusion, pour tout $i \in \mathbb{N}$, $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}_i) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$. Donc $(\boldsymbol{T}^{\cap}, \boldsymbol{C}^{\cap}) \subseteq (\boldsymbol{T}^{\cap}, \overset{\cap}{\lesssim})$, et l'ensemble des comparaisons de TUC $\boldsymbol{C}^{\cap}$ est effectivement égal à la relation de préordre $\overset{\cap}{\lesssim}$ sur $\boldsymbol{T}^{\cap}$. □

Démonstration de la Proposition 7.3.4 . Preuve que la condition est suffisante.
Supposons que $t^{\cap} \subseteq \top^{\sim}$. Donc $\forall t \in t^{\cap}, \top \lesssim t$.
Ainsi $\top^{\cap} \overset{\cap}{\lesssim} t^{\cap}_y$, et $t^{\cap}_y$ est maximal.

Preuve que la condition est nécessaire.
Supposons que $t^{\cap}$ est maximal.
Soit $t^{\cap}_m \subseteq \top^{\sim}$. Nous savons that $t^{\cap}_m \overset{\cap}{\lesssim} t^{\cap}$.
Donc il existe n tel que $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_n$.

Supposons que $n = 0$.
Si $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{\lesssim}$, alors $\forall t \in t^{\cap}, \exists t_m \in t^{\cap}_m, t_m \lesssim t$. Ainsi $t^{\cap} \subseteq \top^{\sim}$, ce qui satisfait la condition 1.
Si $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{\bot}$, alors $t^{\cap}_m \in \bot^{\sqcap}_A$, donc $\top^{\cap} \overset{\cap}{\lesssim} t^{\cap}_m \overset{\cap}{\lesssim} \bot^{\cap}$, ce qui satisfait la condition 2.
Ainsi $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_0$ implique la condition pour tout $t^{\cap}_m \subseteq \top^{\sim}$.

Supposons que $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_n$ implique la condition pour tout $t^{\cap}_m \subseteq \top^{\sim}$.
Soit $t^{\cap}_m \subseteq \top^{\sim}$.
Supposons que $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{n+1}$.
Si $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{\boldsymbol{\varsigma}\, n+1}$, alors $\top^{\cap} \overset{\cap}{\lesssim} t^{\cap}_m \overset{\cap}{\lesssim} \bot^{\cap}$, ce qui satisfait la condition 2.
Si $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{+n+1}$, alors il existe $t^{\cap}_1 \in \boldsymbol{T}^{\cap}$ tel que $(t^{\cap}, t^{\cap}_1) \in \boldsymbol{C}^{\cap}_n$, et $(t^{\cap}_1, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_n$.
$(t^{\cap}_1, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_n$, alors l'un des cas suivants est vrai :
Si $t^{\cap}_1 \subseteq \top^{\sim}$, alors la condition est satisfaite. Si $\top^{\cap} \subseteq \bot^{\cap}$, alors la condition est également satisfaite.
Ainsi $(t^{\cap}, t^{\cap}_m) \in \boldsymbol{C}^{\cap}_{n+1}$ implique la condition pour tout $t^{\cap}_m \subseteq \top^{\sim}$.

Ainsi la condition est satisfaite pour tout n. □

Démonstration de la Proposition 7.3.5 . Preuve que la condition est suffisante.
Nous savons que $(\bot^{\cap}, t^{\cap}_n) \in \boldsymbol{C}^{\cap}_0$.
Donc la condition satisfaite pour $n = 0$

Soit $1 \leq i \leq n$

Supposons que la condition suffisante est vraie pour $n-i+1$
$t^{\cap}_{n-i+1} \overset{\cap}{\lesssim} \bot^{\cap}$.
Considérons m_1 tel que $(\bot^{\cap}, t^{\cap}_{n-i+1}) \in \boldsymbol{C}^{\cap}_{m_1}$:
• Si $t^{\cap}_{n-i} \overset{\cap}{\lesssim} t^{\cap}_{n-i+1}$, alors il existe m_2 tel que $(t^{\cap}_{n-i+1}, t^{\cap}_{n-i}) \in \boldsymbol{C}^{\cap}_{m_2}$.
Ainsi pour $m = \max(m_1, m_2)+1$, $(\bot^{\cap}, t^{\cap}_{n-i}) \in \boldsymbol{C}^{\cap}_{+m} \subseteq \boldsymbol{C}^{\cap}_{m}$, et $t^{\cap}_{n-i} \overset{\cap}{\lesssim} \bot^{\cap}$.
• Si $\exists s_{n-i} \in \boldsymbol{\alpha}^{\cap}(t^{\cap}_{n-i}), \boldsymbol{\varsigma}^{\cap}_{t^{\cap}_{n-i}}(s_{n-i}) = t^{\cap}_{n-i+1}$.
Donc $\exists s_{n-i} \in \boldsymbol{\alpha}^{\cap}(t^{\cap}_{n-i}), (\bot^{\cap}, \boldsymbol{\varsigma}^{\cap}_{t^{\cap}_{n-i}}(s_{n-i})) \in \boldsymbol{C}^{\cap}_{m_1}$.
Et ainsi $(\bot^{\cap}, t^{\cap}_{n-i}) \in \boldsymbol{C}^{\cap}_{\boldsymbol{\varsigma}\, m_1+1} \subseteq \boldsymbol{C}^{\cap}_{m_1+1}$, et $t^{\cap}_{n-i} \overset{\cap}{\lesssim} \bot^{\cap}$.
Donc la condition est satisfaite pour $n-i$

Ainsi par récurrence, la condition suffisante est satisfaite pour tout $n=0$, et $t^{\cap}_0$ est absurde.

Preuve que la condition est nécessaire.
Supposons que $t^{\cap}_0 \overset{\cap}{\lesssim} \bot^{\cap}$.
Considérons m tel que $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_m$.

Supposons que $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_0$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{\overset{\sim}{\lesssim}}$, alors $\exists t \in t^{\cap}_0, t \lesssim \bot$. Cela implique l'item 2.a avec $n=0$.
$(\bot^{\cap}, t^{\cap}_0) \notin \boldsymbol{C}^{\cap}_{\top}$ car $\bot^{\cap} \neq \top^{\cap}$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{\bot}$, alors $t^{\cap}_0 \in \bot^{\sqcap}_A$. Cela implique l'item 2.b avec $n=0$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{\boldsymbol{\Gamma}}$, alors il existe $s \in \boldsymbol{S}_{\mathscr{T}}$ tel que $t^{\cap}_0 = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$. Cela implique l'item 2.c avec $n=0$.
Donc la condition nécessaire est vraie pour $m=0$, avec $n=0$.

Supposons que la condition nécessaire est satisfaite pour m, avec $n=m$.
Supposons que $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{m+1}$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_m$, alors Cela implique que la condition est satisfaite pour $m+1$ avec $n=m$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{\boldsymbol{\varsigma}\, m+1}$, alors il existe $s \in \boldsymbol{\alpha}^{\cap}_{t^{\cap}_0}$ tel que $(\bot^{\cap}, \boldsymbol{\varsigma}^{\cap}_{t^{\cap}_0}(s)) \in \boldsymbol{C}^{\cap}_m$.
Donc la condition est satisfaite pour $m+1$ avec $n=m$ et une liste $(\boldsymbol{\varsigma}^{\cap}_{t^{\cap}_0}(s), \ldots, t^{\cap}_n)$, et elle est ainsi satisfaite pour $n=m+1$ avec la liste $(t^{\cap}_0, \boldsymbol{\varsigma}^{\cap}_{t^{\cap}_0}(s), \ldots, t^{\cap}_n)$.
Si $(\bot^{\cap}, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_{+m+1}$, alors il existe $t^{\cap}_1 \in \boldsymbol{T}^{\cap}$ tel que $(\bot^{\cap}, t^{\cap}_1) \in \boldsymbol{C}^{\cap}_m$, et $(t^{\cap}_1, t^{\cap}_0) \in \boldsymbol{C}^{\cap}_m$. Donc la condition est satisfaite pour $n=m$ with a list $(t^{\cap}_1, \ldots, t^{\cap}_n)$, et elle est ainsi satisfaite pour $m+1$ avec $n=m$ et une liste $(t^{\cap}_0, t^{\cap}_1, \ldots, t^{\cap}_n)$.
Ainsi dans tous les cas, la condition nécessaire est satisfaite pour $m+1$, avec $n=m+1$.

Ainsi la condition nécessaire est satisfaite pour tout m, avec $n=m$. □

Démonstration de la Proposition 7.3.6 . **Preuve de la condition suffisante :**
La première condition suffisante découle directement de la définition 49 : les TUC absurdes sont des TUC minimaux, donc tout TUC absurde est comparable à tout autre TUC.
Ensuite, $\varnothing$ est un élément maximal, comme $\top^{\cap}$. Donc tout TUC similaire à $\top^{\cap}$ est similaire à $\varnothing$.
Finalement, par la définition 47, le préordre $\overset{\cap}{\lesssim}$ contient l'extension naturelle d'un préordre sur un ensemble à un préordre sur l'ensemble des parties de cet ensemble.

Preuve de la condition nécessaire : Soit $t^{\cap}_x, t^{\cap}_y \in \boldsymbol{T}^{\cap}$ tels que $t^{\cap}_x \overset{\cap}{\lesssim} t^{\cap}_y$.

il existe n tel que $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_n^\cap$.

case $n = 0$: $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_0^\cap$
Si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\lesssim}^\cap$, alors la condition 3 est satisfaite.
Si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\widetilde{\top}}^\cap$, alors $t_x^\cap = \varnothing$, et la condition 2 est satisfaite.
Si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_\bot^\cap$ ou $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\boldsymbol{\Gamma}}^\cap$, alors $t_y^\cap = \bot^\cap$, et $t_x^\cap \mathrel{\hat{\simeq}} \bot^\cap$, donc la condition 1 est satisfaite.
Ainsi si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_0^\cap$, donc la condition est satisfaite.

Supposons que la condition est satisfaite si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_n^\cap$
Supposons que $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{n+1}^\cap$. Si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\boldsymbol{\varsigma}\, n+1}^\cap$, alors $t_y^\cap = \bot^\cap$, et $t_x^\cap \mathrel{\hat{\simeq}} \bot^\cap$, donc la condition 1 est satisfaite.
Si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{+n+1}^\cap$, alors il existe $t_z^\cap \in \boldsymbol{T}^\cap$ tel que $(t_y^\cap, t_z^\cap) \in \boldsymbol{C}_n^\cap$ et $(t_z^\cap, t_x^\cap) \in \boldsymbol{C}_n^\cap$.
$(t_z^\cap, t_x^\cap) \in \boldsymbol{C}_n^\cap$ alors l'un des cas suivants est vrai :
• Si $t_x^\cap \mathrel{\hat{\simeq}} \bot^\cap$, la condition 1 est satisfaite.
• Si $t_x^\cap = \varnothing$, la condition 2 est satisfaite.
• Si $\forall t_z \in t_z^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_z$, alors on utilise le fait que $(t_y^\cap, t_z^\cap) \in \boldsymbol{C}_n^\cap$, et donc que l'un des cas suivants est vrai :
•• Si $t_z^\cap \mathrel{\hat{\simeq}} \bot^\cap$, alors $t_x^\cap \mathrel{\hat{\lesssim}} \bot^\cap$, et la condition 1 est satisfaite.
•• Si $t_z^\cap = \varnothing$, alors $t_y^\cap \mathrel{\hat{\simeq}} \top^\cap$. Donc par la proposition 7.3.4, l'un des cas suivants est vrai :
••• Si $t_y^\cap \subseteq \top^\sim$, alors $\forall t_x \in t_x^\cap, \forall t_y \in t_y^\cap, t_x \lesssim t_y$, et la condition 3 est satisfaite.
••• Si $\top^\cap \mathrel{\hat{\lesssim}} \bot^\cap$, alors $t_x^\cap$ est absurde, et la condition 1 est satisfaite.
•• Si $\forall t_y \in t_y^\cap, \exists t_z \in t_z^\cap : t_z \lesssim t_y$, alors $\forall t_y \in t_y^\cap, \exists t_z \in t_z^\cap, t_x \in t_x^\cap : t_x \lesssim t_z \lesssim t_y$. Donc la condition 3 est satisfaite. Ainsi la condition est satisfaite si $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{n+1}^\cap$.

Ainsi la condition est satisfaite pour tout n, donc la condition est satisfaite pour tout $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$. □

Démonstration de la Proposition 7.3.7 . Soit $t^\cap \in \boldsymbol{T}^\cap$ et $s \in \boldsymbol{\alpha}_1^\cap(t^\cap)$ (resp. $\boldsymbol{\alpha}_0^\cap(t^\cap)$).
Par la définition 43 (resp. 44), $\exists t \in t^\cap, s \in \boldsymbol{\alpha}_1(t)$ (resp. $\boldsymbol{\alpha}_0(t)$).
Par la proposition 7.1.4, $s \in \boldsymbol{\alpha}(t)$.
Par la définition 42, $s \in \boldsymbol{\alpha}^\cap(t^\cap)$. □

Démonstration de la Proposition 7.3.8 . Soit $s \in \boldsymbol{S}_{\mathscr{T}}$.
$\subseteq_1$: Soit $t^\cap \in \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\}$.
Par la définition 42, $\exists t \in t^\cap, s \in \boldsymbol{\alpha}(t)$. Donc $t^\cap \neq \varnothing$.
D'un autre côté, par la définition 34, $t \lesssim \boldsymbol{\gamma}(s)$.
Par la définition 47, $t^\cap \mathrel{\hat{\lesssim}} \{\boldsymbol{\gamma}(s)\}$, donc $t^\cap \in \downarrow^\cap \boldsymbol{\gamma}(s)$.
Donc $t^\cap \in \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing$, et $\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \subseteq \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing$.
$\subseteq_2$: Soit $t^\cap \in \bot^\sqcap \setminus \varnothing$. $t^\cap \mathrel{\hat{\lesssim}} \bot^\cap$ qui est un élément minimal de $\boldsymbol{T}^\cap$.
Donc $t^\cap \mathrel{\hat{\lesssim}} \{\boldsymbol{\gamma}(s)\}$ et $t^\cap \in \downarrow^\cap \boldsymbol{\gamma}(s)$ et $t^\cap \in \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing$.
Donc $\bot^\sqcap \setminus \varnothing \subseteq \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing$.
$\supseteq$: Soit $t^\cap \in \downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing$. $t^\cap \mathrel{\hat{\lesssim}} \{\boldsymbol{\gamma}(s)\}$.
Nous utilisons la proposition 7.3.6, donc au moins l'un des cas suivants est vrai :
i) $t^\cap \mathrel{\hat{\lesssim}} \bot^\cap$: alors $t^\cap \in \bot^\sqcap$. Et puisque $t^\cap \neq \varnothing$, $t^\cap \in \bot^\sqcap \setminus \varnothing$.
ii) $t^\cap = \varnothing$: cas impossible.
iii) $\exists t \in t^\cap : t \lesssim \boldsymbol{\gamma}(s)$: alors par la définition 34, $\exists t \in t^\cap : s \in \boldsymbol{\alpha}(t)$, et par la définition 42, $s \in \boldsymbol{\alpha}^\cap(t^\cap)$. Donc $t^\cap \in \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\}$.

Comme les définitions des PosA, des PosAObl et des PosAInt sont parallèles, les preuves pour les PosAObl et les PosAInt sont exactement les mêmes, sauf pour les symboles et les références aux définitions. □

Démonstration de la Proposition 7.3.9 . Si $t_x^\cap = \boldsymbol{T}$ ou $t_x^\cap = \bot^\cap$, alors $\boldsymbol{\alpha}^\cap(\boldsymbol{T}) = \boldsymbol{S}_\mathscr{T}$. Ainsi $\forall t_y^\cap \in \boldsymbol{T}^\cap, \boldsymbol{\alpha}^\cap(t_y^\cap) \subseteq \boldsymbol{\alpha}^\cap(t_x^\cap)$.

Soit $t_x^\cap \in \boldsymbol{T}^\cap \setminus \varnothing - \bot^\sqcap$ et $t_y^\cap \in \boldsymbol{T}^\cap$ tel que $t_x^\cap \overset{\cap}{\lesssim} t_y^\cap$.
Soit $s \in \boldsymbol{\alpha}^\cap(t_y^\cap)$. Par la définition 42, $\exists t \in t_y^\cap, s \in \boldsymbol{\alpha}(t)$.
Nous utilisons la proposition 7.3.6, mais puisque $t_x^\cap \notin \bot^\sqcap$ et $t_x^\cap \neq \varnothing$, $t_x^\cap \overset{\cap}{\lesssim} t_y^\cap$ implique que : $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t$. Donc $\exists t_x \in t_x^\cap : t_x \lesssim t_y$.
Nous utilisons la proposition 7.1.3, $s \in \boldsymbol{\alpha}(t_x)$, et par la définition 42, $s \in \boldsymbol{\alpha}^\cap(t_x^\cap)$.
Puisque les définitions des PosA, PosAObl et PosAInt sont parallèles, les preuves pour les PosAObl et les PosAInt sont les mêmes, sauf pour les symboles et les références aux définitions. □

Démonstration de la Proposition 7.3.10 . Nous utilisons les propositions 7.1.6 et 7.3.8 :

$$
\begin{aligned}
&\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}_?^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing \\
&= \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap) - \boldsymbol{\alpha}_\mathbf{1}^\cap(t^\cap) - \boldsymbol{\alpha}_\mathbf{0}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing \\
&= \Big(\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} - \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}_\mathbf{1}^\cap(t^\cap)\} - \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}_\mathbf{0}^\cap(t^\cap)\}\Big) \cup \bot^\sqcap \setminus \varnothing \\
&= \Big(\Big(\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing\Big) - \Big(\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}_\mathbf{1}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing\Big) \\
&\qquad\qquad - \Big(\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}_\mathbf{0}^\cap(t^\cap)\} \cup \bot^\sqcap \setminus \varnothing\Big)\Big) \cup \bot^\sqcap \setminus \varnothing \\
&= \Big(\downarrow^\cap \boldsymbol{\gamma}(s) \setminus \varnothing - \downarrow^\cap \boldsymbol{\gamma}_\mathbf{1}(s) \setminus \varnothing - \downarrow^\cap \boldsymbol{\gamma}_\mathbf{0}(s) \setminus \varnothing\Big) \cup \bot^\sqcap \setminus \varnothing
\end{aligned}
$$

□

Démonstration de la Proposition 7.3.11 . Si $t_x^\cap = \boldsymbol{T}$, alors par la proposition 7.2.2 $t_y^\cap \subseteq t_x^\cap$.
Par la définition 46, $\boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \subseteq \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$.
Ainsi par la proposition 7.2.2, $\boldsymbol{\varsigma}_{t_y^\cap}^\cap(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s)$.

Si $t_x^\cap = \bot^\cap$, alors $\boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) = \boldsymbol{\varsigma}_\bot(s)$. Par la définition 38, $\forall t_y \in \{t \in t_y^\cap \mid s \in \boldsymbol{\alpha}(t)\}$, $\forall t_y' \in \boldsymbol{\varsigma}_{t_y}(s), \exists t_x \in \boldsymbol{\varsigma}_\bot(s) : t_x \lesssim t_y'$.
Ainsi, $\forall t_y' \in \bigcup_{t_y \in t_y^\cap \mid s \in \boldsymbol{\alpha}(t_y)} \boldsymbol{\varsigma}_{t_y}(s), \exists t_x \in \boldsymbol{\varsigma}_\bot(s) : t_x \lesssim t_y'$.
Donc par la définition 47, $\boldsymbol{\varsigma}_\bot(s) = \boldsymbol{\varsigma}_{\bot^\cap}^\cap(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$.

Montrons les propositions pour tous les autres TUC régulier.
Par la définition 46, $\boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \subseteq \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$.
Ainsi par la proposition 7.2.2, $\boldsymbol{\varsigma}_{t_y^\cap}^\cap(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s)$.

Soit $t_x^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing - \bot^{\sqcap}$ et $t_y^{\cap} \in \boldsymbol{T}^{\cap}$ et $s \in \boldsymbol{S}_{\mathscr{T}}$ tel que $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$ et $s \in \boldsymbol{\alpha}^{\cap}(t_y^{\cap})$.
1) Soit $t_a \in \boldsymbol{\varsigma}^{\cap}_{t_y^{\cap}}(s)$. Par la définition 46, $\exists t_b \in t_y^{\cap} : s \in \boldsymbol{\alpha}(t_b)$ et $t_a \in \boldsymbol{\varsigma}_{t_b}(s)$.
2) Nous utilisons $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$ et la proposition 7.3.6.
Puisque $t_x^{\cap} \notin \bot^{\sqcap}$ et $t_x^{\cap} \neq \varnothing$, nous savons que $\forall t_y \in t_y^{\cap}, \exists t_x \in t_x^{\cap} : t_x \lesssim t_y$.
Nous restreignons ces t_y à être tels que $s \in \boldsymbol{\alpha}(t_y)$.
Nous utilisons les définitions 38 et 47 : $\forall t_y \in t_y^{\cap}$ tel que $s \in \boldsymbol{\alpha}(t_y)$, il existe $t_x \in t_x^{\cap}$ tel que $\forall t' \in \boldsymbol{\varsigma}_{t_y}(s), \exists t \in \boldsymbol{\varsigma}_{t_x}(s) : t \lesssim t'$
3) Ceci peut être réécrit : $\forall t_y \in t_y^{\cap}$ tel que $s \in \boldsymbol{\alpha}(t_y), \forall t' \in \boldsymbol{\varsigma}_{t_y}(s), \exists t_x, t$ tel que $t_x \in t_x^{\cap}, t \in \boldsymbol{\varsigma}_{t_x}(s)$ et $t \lesssim t'$
4) en combinant 1) et 3) avec $t_y = t_b$, et $t' = t_a$, on obtient :
$\exists t_b \in t_y^{\cap}$ tel que $s \in \boldsymbol{\alpha}(t_b)$ et $t_a \in \boldsymbol{\varsigma}_{t_b}(s), \exists t_x, t$ tel que $t_x \in t_x^{\cap}, t \in \boldsymbol{\varsigma}_{t_x}(s)$ et $t \lesssim t_a$
5) Donc il existe t et t_x tels que $t_x \in t_x^{\cap}$ et $t \in \boldsymbol{\varsigma}_{t_x}(s)$ et $t \lesssim t_a$,
6) Par la définition 46, $t \in \boldsymbol{\varsigma}^{\cap}_{t_x^{\cap}}(s)$. Donc $\forall t_a \in \boldsymbol{\varsigma}^{\cap}_{t_y^{\cap}}(s), \exists t \in \boldsymbol{\varsigma}^{\cap}_{t_x^{\cap}}(s) : t \lesssim t_a$, et par la définition 47, $\boldsymbol{\varsigma}^{\cap}_{t_x^{\cap}}(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}^{\cap}_{t_y^{\cap}}(s)$. □

Démonstration de la Proposition 7.3.12 . $\top^{\sqcap}$: Par la proposition 7.2.4, $\top^{\cap}$ est un élément maximal pour $\overset{\cap}{\lesssim}$.
Donc $\forall t^{\cap} \in \boldsymbol{T}^{\cap}, t^{\cap} \overset{\cap}{\lesssim} \top^{\cap}$. Donc $\forall t^{\sqcap} \in \boldsymbol{T}^{\sqcap}, t^{\sqcap} \overset{\sqcap}{\leq} \top^{\sqcap}$, et $\top^{\sqcap}$ est la borne supérieure de $(\boldsymbol{T}^{\sqcap}, \overset{\sqcap}{\leq})$.
$\bot^{\sqcap}$: Par la proposition 7.2.6, $\bot^{\cap}$ est un élément minimal de $\overset{\cap}{\lesssim}$. Donc $\forall t^{\cap} \in \boldsymbol{T}^{\cap}, \bot^{\cap} \overset{\cap}{\lesssim} t^{\cap}$.
Donc $\forall t^{\sqcap} \in \boldsymbol{T}^{\sqcap}, \bot^{\sqcap} \overset{\sqcap}{\leq} t^{\sqcap}$, donc $\bot^{\sqcap}$ est la borne inférieure de $(\boldsymbol{T}^{\sqcap}, \overset{\sqcap}{\leq})$. □

Démonstration de la Proposition 7.3.13 . Soit $t^{\sqcap} \in \boldsymbol{T}^{\sqcap}$, et $t_x^{\cap}, t_y^{\cap} \in t^{\sqcap}$. Nous prouverons que $t_x^{\cap} \cup t_y^{\cap} \overset{\cap}{\simeq} t_x^{\cap}$.
$\overset{\cap}{\lesssim}$ *:* Nous savons par la proposition 7.2.2 que $t_x^{\cap} \subseteq t_x^{\cap} \cup t_y^{\cap}$ implique que $t_x^{\cap} \cup t_y^{\cap} \overset{\cap}{\lesssim} t_x^{\cap}$.
$\overset{\cap}{\gtrsim}$ *:* Nous savons que $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$. Par la proposition 7.3.6, au moins l'un des cas suivants est vrai :
i) $t_x^{\cap} \overset{\cap}{\lesssim} \bot^{\cap}$: alors $t_x^{\cap} \cup t_y^{\cap} \overset{\cap}{\simeq} t_x^{\cap} \overset{\cap}{\simeq} \bot^{\cap}$;
ii) $t_x^{\cap} = \varnothing$: alors $t_x^{\cap} \cup t_y^{\cap} = t_y^{\cap} \overset{\cap}{\simeq} t_x^{\cap}$;
iii) $\forall t_y \in t_y^{\cap}, \exists t_x \in t_x^{\cap} : t_x \lesssim t_y$: Donc $\forall t \in t_x^{\cap} \cup t_y^{\cap}, \exists t_x \in t_x^{\cap} : t_x \lesssim t$, et $t_x^{\cap} \overset{\cap}{\lesssim} t_x^{\cap} \cup t_y^{\cap}$. □

Démonstration du Lemme 7.3.14 . Soit $t^{\cap} \in \boldsymbol{T}^{\cap}$ et $t \in \boldsymbol{T}$ tel que $\forall t' \in t^{\cap}, t' \lesssim t$.
$\overset{\cap}{\lesssim}$ *:* $t_y^{\cap} \subseteq t_y^{\cap} \cup \{t\}$, donc $t^{\cap} \cup \{t\} \overset{\cap}{\lesssim} t^{\cap}$;
$\overset{\cap}{\gtrsim}$ *:* $t^{\cap} \neq \varnothing$. Soit $t' \in t^{\cap} \cup \{t\}$.
Si $t' \in t^{\cap}$, alors il existe $t''(= t') \in t^{\cap} : t'' \lesssim t'$.
Si $t' = t$, alors il existe $t'' \in t^{\cap} t'' \lesssim t'$.
Donc $\forall t' \in t^{\cap} \cup \{t\}, \exists t'' \in t^{\cap} : t'' \lesssim t'$, et $t^{\cap} \cup \{t\} \overset{\cap}{\gtrsim} t^{\cap}$. □

Démonstration de la Proposition 7.3.15 . Soit $t^{\sqcap} \in \boldsymbol{T}^{\sqcap}$, et $t^{\cap} \in t^{\sqcap}$. Par définition, $\Uparrow t^{\cap}$ est l'union de tous les TUC dans $t^{\sqcap}$.
Nous savons par la proposition 7.3.13 que $t^{\sqcap}$ est clos par l'opération d'union, donc $\Uparrow t^{\cap} \in t^{\sqcap}$. □

Démonstration de la Proposition 7.3.16 . Par définition, $t_x^{\cap} \overset{\cap}{\simeq} t_y^{\cap}$, alors $[t_x^{\cap}] = [t_y^{\cap}]$, et $\Uparrow t_x^{\cap} = \Uparrow t_y^{\cap}$ □

Démonstration de la Proposition 7.3.17 . Soit $t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\cap}$. Par la proposition 7.3.15 nous savons que $t_x^{\cap} \overset{\cap}{\simeq} \Uparrow t_x^{\cap}$ et $t_y^{\cap} \overset{\cap}{\simeq} \Uparrow t_y^{\cap}$.
$\Rightarrow$ *:* Si $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$, alors $t_x^{\cap} \overset{\cap}{\simeq} \Uparrow t_x^{\cap} \overset{\cap}{\lesssim} \Uparrow t_y^{\cap} \overset{\cap}{\simeq} t_y^{\cap}$. Donc $\Uparrow t_x^{\cap} \overset{\cap}{\lesssim} \Uparrow t_y^{\cap}$.
$\Leftarrow$ *:* Si $\Uparrow t_x^{\cap} \overset{\cap}{\lesssim} \Uparrow t_y^{\cap}$, alors $\Uparrow t_x^{\cap} \overset{\cap}{\simeq} t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap} \overset{\cap}{\simeq} \Uparrow t_y^{\cap}$. Donc $t_x^{\cap} \overset{\cap}{\lesssim} t_y^{\cap}$. □

Démonstration de la Proposition 7.3.18 . Soit $t^\cap \in \boldsymbol{T}^\cap$:
1) si $t^\cap \in \boldsymbol{T}^\cap \setminus \varnothing - \bot^\sqcap$:
$\subseteq$ *:* Soit $t \in \Uparrow t^\cap$. Par définition, $\exists t_x^\cap \in [t^\cap], t \in t_x^\cap$. Donc $t^\cap \triangleq t_x^\cap$ et $t^\cap \overset{\cap}{\lesssim} t_x^\cap$.
Par la proposition 7.3.6, au moins l'un des cas suivants est vrai :
i) $t^\cap \overset{\cap}{\lesssim} \bot^\cap$ (impossible d'après nos hypothèses) ;
ii) $t^\cap = \varnothing$ (impossible d'après nos hypothèses) ;
iii) $\forall t_x \in t_x^\cap, \exists t' \in t^\cap : t' \lesssim t_x$. Donc pour $t \in t_x^\cap$, $\exists t' \in t^\cap : t' \lesssim t$. Et ainsi $t \in \uparrow t^\cap$.
$\supseteq$ *:* Soit $t \in \uparrow t^\cap$. Alors $\exists t' \in t^\cap, t' \lesssim t$. Par le lemme 7.3.14, $t^\cap \cup \{t\} \triangleq t^\cap$, donc $t^\cap \cup \{t\} \in [t^\cap]$, donc $t \in \Uparrow t^\cap$.
2) si $t^\cap \in \bot^\sqcap$, alors $\boldsymbol{T} \in [t^\cap]$ et $\Uparrow t^\cap = \boldsymbol{T}$;
3) si $\varnothing \triangleq \top^\cap$, alors par la proposition 7.3.16, $\Uparrow \varnothing = \Uparrow \top^\cap$. □

Démonstration de la Proposition 7.3.19 . Montrons les deux implications :
$\Rightarrow$ *:* Si $t^\cap$ est clos, alors il existe $t^{\cap\prime} \in \boldsymbol{T}^\cap, t^\cap = \Uparrow t^{\cap\prime}$.
i) si $t^{\cap\prime} \in \bot^\sqcap$, alors $t^\cap = \boldsymbol{T}$;
ii) si $t^{\cap\prime} = \varnothing$, alors $t^\cap = \Uparrow \bot^\cap$, et $\Uparrow \bot^\cap$ est soit $\boldsymbol{T}$, soit un ensemble supérieur non-vide de $\boldsymbol{T}$;
iii) si $t^{\cap\prime} \in \boldsymbol{T}^\cap_{regular}$, alors $t^\cap$ est un ensemble supérieur non-vide de $\boldsymbol{T}$.
$\Leftarrow$ *:*
i) si $t^\cap = \boldsymbol{T}$, alors pour tout $t^{\cap\prime} \in \bot^\sqcap$, $t^\cap = \Uparrow t^{\cap\prime}$, donc $t^\cap$ est clos ;
ii) si $t^\cap \notin \bot^\sqcap$, et $t^\cap$ est un ensemble supérieur non-vide de $\boldsymbol{T}$, alors par les propositions 7.3.16 et 7.3.15, $\Uparrow t^\cap = t^\cap$, donc $t^\cap$ est clos. □

Démonstration de la Proposition 7.3.20 . 1) *Extensif :* Soit $t^\cap \in \boldsymbol{T}^\cap$. $t^\cap \in [t^\cap]$ donc par la définition 54, $t^\cap \subseteq \Uparrow t^\cap$, et $\Uparrow$ est extensif.

2) *Croissant par rapport à* $\subseteq$ *:* Soit $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$ tel que $t_x^\cap \subseteq t_y^\cap$.
D'abord, nous savons par la proposition 7.2.2 que $t_y^\cap \overset{\cap}{\lesssim} t_x^\cap$.
i) Si $t_y^\cap \in \bot^\sqcap$, alors $\Uparrow t_y^\cap = \boldsymbol{T}$ (prop. 7.3.18) et $\Uparrow t_x^\cap \subseteq \Uparrow t_y^\cap$.
ii) Si $t_x^\cap \in \bot^\sqcap$, alors $t_y^\cap$ également, et $\Uparrow t_x^\cap = \Uparrow t_y^\cap = \boldsymbol{T}$ (prop. 7.3.18).
iii) Si $t_y^\cap = \varnothing$, alors $t_x^\cap = t_y^\cap = \varnothing$, et $\Uparrow t_x^\cap = \Uparrow t_y^\cap$.
iv) Si $t_y^\cap \notin \bot^\sqcap$, $t_y^\cap \neq \varnothing$, $t_x^\cap \notin \bot^\sqcap$, et $t_x^\cap \neq \varnothing$: Soit $t \in \Uparrow t_x^\cap = \uparrow t_x^\cap$. Nous savons que $\exists t' \in t_x^\cap, t' \lesssim t$. Puisque $t_x^\cap \subseteq t_y^\cap$, alors $\exists t' \in t_y^\cap, t' \lesssim t$, et $t' \in \uparrow t_y^\cap = \Uparrow t_y^\cap$.
v) Si $t_x^\cap = \varnothing$, alors $\Uparrow t_x^\cap = \Uparrow \top^\cap$. $\top$ est un élément maximal de $\boldsymbol{T}$, donc par le lemme 7.3.14, $t_y^\cap \triangleq t_y^\cap \cup \top^\cap$, et par la définition 54, $\Uparrow t_y^\cap = \Uparrow t_y^\cap \cup \top^\cap$. en utilisant les items i) to iv), nous savons que $\Uparrow t_x^\cap = \Uparrow \top^\cap \subseteq \Uparrow t_y^\cap \cup \top^\cap = \Uparrow t_y^\cap$.
Donc $\Uparrow t_x^\cap \subseteq \Uparrow t_y^\cap$, et $\Uparrow$ est croissant.

3) *Idempotent :* Soit $t^\cap \in \boldsymbol{T}^\cap$.
i) si $t^\cap \in \bot^\sqcap$, alors $\Uparrow t^\cap = \boldsymbol{T}$, Et puisque nous savons que $\boldsymbol{T} \in \bot^\sqcap$, $\Uparrow \Uparrow t^\cap = \Uparrow t^\cap = \boldsymbol{T}$.
ii) si $t^\cap \notin \bot^\sqcap$ et $t^\cap \neq \varnothing$, alors :
$\supseteq$ *:* Nous savons par l'extensivité de $\Uparrow$ que $\Uparrow t^\cap \subseteq \Uparrow \Uparrow t^\cap$.
$\subseteq$ *:* Soit $t \in \Uparrow \Uparrow t^\cap$. $\Uparrow t^\cap \triangleq t^\cap$ donc (prop : 7.3.13 $\Uparrow t^\cap \notin \bot^\sqcap$, et nous savons également que $t^\cap \subseteq \Uparrow t^\cap$, donc $\Uparrow t^\cap \neq \varnothing$. En utilisant les propriétés des ensembles supérieurs : $\Uparrow \Uparrow t^\cap = \uparrow \Uparrow t^\cap = \uparrow \uparrow t^\cap = \uparrow t^\cap = \Uparrow t^\cap$.
iii) si $t^\cap = \varnothing$, alors $\Uparrow t^\cap = \Uparrow \bot^\cap$, et en utilisant ii), $\Uparrow t^\cap = \Uparrow \bot^\cap = \Uparrow \Uparrow \bot^\cap = \Uparrow \Uparrow t^\cap$. □

Démonstration de la Proposition 7.3.21 . Montrons que $\overset{\cap}{\lesssim}$ est antisymétrique sur l'ensemble $\boldsymbol{T}^{\Uparrow}$.
Soit $t_x^{\Uparrow}, t_y^{\Uparrow} \in \boldsymbol{T}^{\Uparrow}$ tel que $t_x^{\Uparrow} \overset{\cap}{\simeq} t_y^{\Uparrow}$. Par la définition 55, nous savons qu'il existe $t_x^{\cap}, t_y^{\cap} \in \boldsymbol{T}^{\cap}$, tels que $\Uparrow t_x^{\cap} = t_x^{\Uparrow}$ et $\Uparrow t_y^{\cap} = t_y^{\Uparrow}$.
Ainsi $\Uparrow t_y^{\cap} \overset{\cap}{\simeq} \Uparrow t_x^{\cap}$
Puisque $\overset{\cap}{\simeq}$ est le noyau de $\Uparrow$ (cf., prop. 7.3.16), $\Uparrow\Uparrow t_x^{\cap} = \Uparrow\Uparrow t_y^{\cap}$
Puisque $\Uparrow$ est idempotent, (cf., prop. 7.3.20), $\Uparrow t_x^{\cap} = \Uparrow t_y^{\cap}$
Ainsi $t_x^{\Uparrow} = t_y^{\Uparrow}$, $\overset{\cap}{\lesssim}$ est antisymétrique et définit un ordre partiel sur $\boldsymbol{T}^{\Uparrow}$. □

Démonstration de la Proposition 7.3.22 . $\boldsymbol{T}^{\Uparrow} \setminus \boldsymbol{T} \subseteq \boldsymbol{T}^{\cap}_{regular}$, ainsi la proposition 7.3.9 s'applique sauf pour le cas où $t_x^{\cap} = \boldsymbol{T}$.
Par la définition 42, $\boldsymbol{\alpha}^{\cap}(\boldsymbol{T}) = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{\alpha}(t) \supseteq \bigcup_{t \in t_y^{\cap}} \boldsymbol{\alpha}(t) = \boldsymbol{\alpha}^{\cap}(t_y^{\cap})$.
Ainsi, $\boldsymbol{T} \overset{\cap}{\lesssim} t_y^{\cap} \implies \boldsymbol{\alpha}^{\cap}(t_y^{\cap}) \subseteq \boldsymbol{\alpha}^{\cap}(\boldsymbol{T})$.
Puisque les définitions 42, 43, et 44 sont similaires, la même preuve peut être réécirte pour les PosAObl et PosAInt. □

Démonstration de la Proposition 7.3.23 . $\boldsymbol{T}^{\Uparrow} \setminus \boldsymbol{T} \subseteq \boldsymbol{T}^{\cap}_{regular}$, ainsi la proposition 7.3.11 s'applique sauf pour le cas où $t_x^{\cap} = \boldsymbol{T}$.
Soit $s \in \boldsymbol{\alpha}^{\cap}(t_y^{\cap})$.
Par la définition 46, $\boldsymbol{\varsigma}_{\boldsymbol{T}}^{\cap}(s) = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{\varsigma}_t(s) \supseteq \bigcup_{t \in t_y^{\cap}} \boldsymbol{\varsigma}_t(s) = \boldsymbol{\varsigma}_{t_y^{\cap}}^{\cap}(s)$.
Par la proposition 7.2.2, $\boldsymbol{\varsigma}_{\boldsymbol{T}}^{\cap}(s) \overset{\cap}{\lesssim} \boldsymbol{\varsigma}_{t_y^{\cap}}^{\cap}(s)$ Ainsi, $\boldsymbol{T} \overset{\cap}{\leq} t_y^{\cap} \implies \Uparrow \boldsymbol{\varsigma}_{\boldsymbol{T}}^{\cap}(s) \overset{\cap}{\leq} \Uparrow \boldsymbol{\varsigma}_{t_y^{\cap}}^{\cap}(s)$.
□

Démonstration de la Proposition 8.2.1 . Un multigraphe étiqueté et orienté est fini si et seulement si à la fois son ensemble de nœuds et son ensemble d'arcs sont finis.
Soit $G = \langle U, \boldsymbol{l}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un GU, de graphe sous-jacent $graph(G)$.
$graph(G)$ peut avoir un arc (u,v) pour toute étiquette dans l'ensemble $\boldsymbol{S}_{\mathscr{T}} \cup \boldsymbol{S}_{\mathscr{C}} \cup \{=\}$.
Par les définitions 27 et 56, $\boldsymbol{S}_{\mathscr{T}}$ et $\boldsymbol{S}_{\mathscr{C}}$ sont finis. Ainsi le nombre d'arcs entre u et v est majoré par $|\boldsymbol{S}_{\mathscr{T}}| + |\boldsymbol{S}_{\mathscr{C}}| + 1$.
Un corollaire est que si l'ensemble des arcs de $graph(G)$ est fini, alors l'ensemble des nœuds de $graph(G)$ est fini.

Ainsi $graph(G)$ (resp. G) est fini si et seulement si son ensemble de nœuds (de nœuds d'unité) est fini. □

Démonstration de la Proposition 9.1.1 . Par la définition 34, $|\boldsymbol{\alpha}(t)| = valence(t)$. Comme de plus $0 \notin \boldsymbol{S}_{\mathscr{T}}$, alors $|\{0\} \cup \boldsymbol{\alpha}(t)| = 1 + valence(t)$. □

Démonstration de la Proposition 9.1.2 . Soit $t_1 \simeq t_2$.
$t_1 \lesssim t_2$, donc d'après la définition 85, $\pi_{\{0\} \cup \boldsymbol{\alpha}(t_2)} \delta(\{t_1\}) \subseteq \delta(\{t_2\})$.
Comme $\{0\} \cup \boldsymbol{\alpha}(t_2) = \{0\} \cup \boldsymbol{\alpha}(t_1)$ par la proposition 7.1.3, alors $\delta(\{t_1\}) \subseteq \delta(\{t_2\})$.
Finalement par la symétrie de t_1 et t_2 dans la preuve, $\delta(\{t_1\}) = \delta(\{t_2\})$. □

Démonstration de la Proposition 9.1.3 . Soit $t \in \boldsymbol{T}$ et $s \in \boldsymbol{\alpha_1}(t)$.
Par la proposition 35, $t \lesssim \boldsymbol{\gamma_1}(s)$.
Par la définition 85, $\pi_{\{0\} \cup \boldsymbol{\alpha}(\{\boldsymbol{\gamma_1}(s)\})} \delta(\{t\}) \subseteq \delta(\{\boldsymbol{\gamma_1}(s)\})$.
Donc $\pi_s \delta(\{t\}) \subseteq \pi_s \delta(\{\boldsymbol{\gamma_1}(s)\})$.
Par la définition 86, $\pi_s \delta(\{\boldsymbol{\gamma_1}(s)\}) \subseteq D \setminus \bullet$.
Ainsi $\pi_s \delta(\{t\}) \subseteq D \setminus \bullet$.

Soit $t \in \boldsymbol{T}$ et $s \in \boldsymbol{\alpha_0}(t)$.
Par la proposition 36, $t \lesssim \boldsymbol{\gamma_0}(s)$.
Par la définition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(\{\boldsymbol{\gamma_0}(s)\})}\delta(\{t\}) \subseteq \delta(\{\boldsymbol{\gamma_0}(s)\})$.
Donc $\pi_s\delta(\{t\}) \subseteq \pi_s\delta(\{\boldsymbol{\gamma_0}(s)\})$.
Par la définition 86, $\pi_s\delta(\{\boldsymbol{\gamma_0}(s)\}) \subseteq \{\bullet\}$.
Ainsi $\pi_s\delta(\{t\}) \subseteq \{\bullet\}$. □

Démonstration de la Proposition 9.1.4 . Soit $t \in A^{\sim}$. Par la définition 39, soit $t \in \bot^{\sim}$, ou t possède une PosA à la fois obligatoire et interdite.

Si $t \in \bot^{\sim}$, alors $t \triangleq \bot$. Par la définition 88 et la proposition 9.1.2, nous savons que $\delta(\{t\}) = \delta(\{\bot\}) = \varnothing$.

Si $t \in \{t \in \boldsymbol{T} \mid \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t) \neq \varnothing\}$, alors soit $s \in \boldsymbol{\alpha_1}(t) \cap \boldsymbol{\alpha_0}(t)$.
Par les définitions 35 et 36, $t \lesssim \boldsymbol{\gamma_1}(s)$ et $t \lesssim \boldsymbol{\gamma_0}(s)$.
Par la définition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(\boldsymbol{\gamma_1}(s))}\delta(\{t\}) \subseteq \delta(\{\boldsymbol{\gamma_1}(s)\})$ et $\pi_{\{0\}\cup\boldsymbol{\alpha}(\boldsymbol{\gamma_0}(s))}\delta(\{t\}) \subseteq \delta(\{\boldsymbol{\gamma_0}(s)\})$.
Puisque $s \in \boldsymbol{\alpha}(\boldsymbol{\gamma_1}(s))$ et $s \in \boldsymbol{\alpha}(\boldsymbol{\gamma_0}(s))$, alors $\pi_s\delta(\{t\}) \subseteq \pi_s\delta(\{\boldsymbol{\gamma_1}(s)\})$, et $\pi_s\delta(\{t\}) \subseteq \pi_s\delta(\{\boldsymbol{\gamma_0}(s)\})$.
Finalement par la définition 86, $\pi_s\delta(\{t\}) \subseteq D \setminus \bullet$ et $\pi_s\delta(\{t\}) \subseteq \{\bullet\}$
Ainsi, $\pi_s\delta(\{t\}) = \varnothing$, ce qui implique $\delta(\{t\}) = \varnothing$.

Dans le deux cas, $\delta(\{t\}) = \varnothing$. □

Démonstration de la Proposition 9.1.5 . Soit $t^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing$. Par la définition 89 et par la définition de l'opérateur de jointure, l'ensemble d'attributs de $\delta(t^{\cap})$ est l'union des ensembles d'attributs des TUP dont est composé $t^{\cap}$. Par les définitions 42 et 84, cet ensemble d'attributs est :

$$\bigcup_{t\in t^{\cap}} \{0\} \cup \boldsymbol{\alpha}(t) = \{0\} \cup \bigcup_{t\in t^{\cap}} \boldsymbol{\alpha}(t) = \{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap})$$

Et comme $0 \notin \boldsymbol{S}_{\mathscr{T}}$, nous avons $|\{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap})| = 1 + |\boldsymbol{\alpha}^{\cap}(t^{\cap})| = 1 + valence(t^{\cap})$.
Ainsi $\delta(t^{\cap}) \subseteq D^{1+valence(t^{\cap})}$, d'attributs $\{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap})$. □

Démonstration de la Proposition 9.1.6 . Soit $t^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing$, $r_1, r_2 \in \delta(t^{\cap})$ tels que $r_1(0) = r_2(0)$.
Soit $t \in t^{\cap}$
Par la définition 89 et par la définition de l'opérateur de jointure,
$\pi_{\{0\}\cup\boldsymbol{\alpha}(t)}\delta(\{t\}) \subseteq \delta(t^{\cap})$. Ainsi $r_{1|\{0\}\cup\boldsymbol{\alpha}(t)} \in \delta(\{t\})$, et $r_{2|\{0\}\cup\boldsymbol{\alpha}(t)} \in \delta(\{t\})$.
Nous savons que $r_{1|\{0\}\cup\boldsymbol{\alpha}(t)}(0) = r_1(0) = r_2(0) = r_{2|\{0\}\cup\boldsymbol{\alpha}(t)}(0)$.
Alors par la définition 84, $r_{1|\{0\}\cup\boldsymbol{\alpha}(t)} = r_{2|\{0\}\cup\boldsymbol{\alpha}(t)}$.
Ou de manière équivalente, $\forall s \in \{0\} \cup \boldsymbol{\alpha}(t)$, $r_1(s) = r_2(s)$.
Ceci vaut pour tout $t \in t^{\cap}$.
Comme de plus $\{0\} \cup \bigcup_{t\in t^{\cap}} \boldsymbol{\alpha}(t) = \{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap})$, alors $\forall s \in \{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap}), r_1(s) = r_2(s)$.
Ensuite, $r_1(0) = r_2(0)$, et comme l'ensemble des attributs de $\delta(t^{\cap})$ est $\{0\} \cup \boldsymbol{\alpha}^{\cap}(t^{\cap})$, alors $r_1 = r_2$, et 0 est une clé unique pour $\delta(t^{\cap})$.
Finalement, comme l'interprétation de $\varnothing$ est égale à l'interprétation de $\top^{\cap}$, et comme 0 est une clé unique pour $\top^{\cap}$, alors 0 est une clé unique pour $\varnothing$ et donc pour tout TUC. □

Démonstration de la Proposition 9.1.7 . Soit $t^\cap \in \boldsymbol{T}^\cap \setminus \varnothing$. Par la définition 89 et par la propriété de l'opérateur de jointure, et comme 0 est une clé unique pour tout TUC, alors
$\pi_0\delta(t^\cap) \subseteq \bigcap_{t\in t^\cap} \pi_0\delta(\{t\})$
Donc pour un $t \in t^\cap$. $\pi_0\delta(t^\cap) \subseteq \pi_0\delta(\{t\})$.
Comme nous savons que $\top$ est un élément maximal (prop. 7.2.4), et par la proposition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(\top)}\delta(\{t\}) \subseteq \delta(\{\top\})$. Ainsi $\pi_0\delta(\{t\}) \subseteq \pi_0\delta(\{\top\})$.
Par la définition 88, $\pi_0\delta(t^\cap) \subseteq \pi_0\delta(\{t\}) \subseteq D \setminus \bullet$.
Finalement, comme l'interprétation de $\varnothing$ est égale à l'interprétation de $\top^\cap$, alors ceci vaut pour tout TUC. □

Démonstration de la Proposition 9.1.8 . Montrons que $\pi_0\delta(\{t_1,t_2\}) = \pi_0\delta(\{t_1\}) \cap \pi_0\delta(\{t_2\})$.

$\subseteq$:

$$\pi_0\delta(\{t_1,t_2\}) = \pi_0\Big(\delta(\{t_1\}) \bowtie \delta(\{t_2\})\Big) \tag{A.37}$$
$$\subseteq \pi_0\delta(\{t_1\}) \cap \pi_0\delta(\{t_2\}) \tag{A.38}$$

$\supseteq$:
Soit $i \in \pi_0\delta(\{t_1\}) \cap \pi_0\delta(\{t_2\})$.
$\exists r_1 \in \delta(\{t_1\})$ et $r_2 \in \delta(\{t_2\})$ tels que $r_1(0) = r_2(0) = i$.
Soit $s \in \boldsymbol{\alpha}(t_1) \cap \boldsymbol{\alpha}(t_2)$.
$t_1 \lesssim \boldsymbol{\gamma}(s)$. Donc par la définition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(\{\boldsymbol{\gamma}(s)\})}\delta(\{t_1\}) \subseteq \delta(\{\boldsymbol{\gamma}(s)\})$.
Puisque $s \in \boldsymbol{\alpha}(\{\boldsymbol{\gamma}(s)\})$, alors $\pi_{0,s}\delta(\{t_1\}) \subseteq \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Donc $(r_1(0), r_1(s)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par symétrie de 1 et 2 dans la preuve, $(r_2(0), r_2(s)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par la définition 84, 0 est une clé unique pour $\delta(\{\boldsymbol{\gamma}(s)\})$, so as $r_1(0) = r_2(0) = i$, alors $r_1(s) = r_2(s)$.
Ainsi, pour tout $s \in \boldsymbol{\alpha}(t_1) \cap \boldsymbol{\alpha}(t_2)$, $r_1(s) = r_2(s)$. De plus, $r_1(0) = r_2(0)$
$\{0\} \cup \boldsymbol{\alpha}(t_1) \cap \boldsymbol{\alpha}(t_2)$ sont exactement les attributs partagés par $\delta(\{t_1\})$ et $\delta(\{t_2\})$.
Ainsi par la définition de l'opérateur de jointure, $r_1 \in \pi_{\{0\}\cup\boldsymbol{\alpha}(\{t_1\})}\delta(\{t_1\}) \bowtie \delta(\{t_2\})$.
Ainsi $i \in \pi_0\big(\delta(\{t_1\}) \bowtie \delta(\{t_2\})\big)$.
Donc $\pi_0\big(\delta(\{t_1\}) \cap \pi_0\delta(\{t_2\})\big) \subseteq \pi_0\delta(\{t_1,t_2\})$.
Finalement, puisque l'interprétation de $\varnothing$ est la même que l'interprétation de $\top^\cap$, alors ceci est vrai pour tout TUC.

Donc $\pi_0\delta(\{t_1,t_2\}) = \pi_0\delta(\{t_1\}) \cap \pi_0\delta(\{t_2\})$. □

Démonstration de la Proposition 9.1.9 . Soit $t^\cap \in \boldsymbol{T}^\cap \setminus \varnothing$, et $s \in \boldsymbol{\alpha}_\mathbf{1}^\cap(t^\cap)$.
Par la définition 43, il existe $t \in t^\cap$ tel que $s \in \boldsymbol{\alpha}_\mathbf{1}(t)$.
Par la proposition 9.1.3, $\pi_s\delta(\{t\}) \subseteq D \setminus \bullet$.
Par une propriété de l'opérateur de jointure, $\pi_s\delta(t^\cap) \subseteq \bigcap_{t\in t^\cap | s\in\boldsymbol{\alpha}(t)} \pi_s\delta(\{t\}) \subseteq \pi_s\delta(\{t\})$.
Ainsi, $\pi_s\delta(t^\cap) \subseteq D \setminus \bullet$.
Finalement, comme $\varnothing$ ne possède pas de PosAObl, ceci vaut pour tout TUC.

Soit $t^\cap \in \boldsymbol{T}^\cap \setminus \varnothing$, et $s \in \boldsymbol{\alpha}_\mathbf{0}^\cap(t^\cap)$.

Par la définition 44, il existe $t \in t^{\cap}$ tel que $s \in \boldsymbol{\alpha_0}(t)$.
Par la proposition 9.1.3, $\pi_s\delta(\{t\}) \subseteq \{\bullet\}$.
Par une propriété de l'opérateur de jointure, $\pi_s\delta(t^{\cap}) \subseteq \bigcap_{t\in t^{\cap}|s\in\boldsymbol{\alpha}(t)} \pi_s\delta(\{t\}) \subseteq \pi_s\delta(\{t\})$.
Ainsi, $\pi_s\delta(t^{\cap}) \subseteq \{\bullet\}$.
Finalement, comme $\varnothing$ ne possède pas de PosAInt, ceci vaut pour tout TUC.

□

Démonstration de la Proposition 9.1.10 . Soit $t^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing$, et $s \in \boldsymbol{\alpha}^{\cap}(t^{\cap})$.

$$\pi_s\delta(t^{\cap}) \setminus \bullet = \left(\pi_s \bowtie_{t\in t^{\cap}} \delta(\{t\})\right) \setminus \bullet \text{ (def. 89)} \tag{A.39}$$

$$\subseteq \left(\bigcap_{t\in t^{\cap}|s\in\boldsymbol{\alpha}(s)} \pi_s\delta(\{t\})\right) \setminus \bullet \text{ (prop. de l'opérateur de jointure)} \tag{A.40}$$

$$\subseteq \bigcap_{t\in t^{\cap}|s\in\boldsymbol{\alpha}(s)} \left(\pi_s\delta(\{t\}) \setminus \bullet\right) \tag{A.41}$$

$$\subseteq \bigcap_{t\in t^{\cap}|s\in\boldsymbol{\alpha}(s)} \pi_0\delta(\boldsymbol{\varsigma}_t(s)) \text{ (def. 87)} \tag{A.42}$$

$$\subseteq \pi_0\delta(\boldsymbol{\varsigma}_{t^{\cap}}^{\cap}(s)) \text{ (def. 46 et prop. 9.1.8)} \tag{A.43}$$

□

Démonstration de la Proposition 9.1.11 . Soit $t_2^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing$, et $t_1^{\cap} = \boldsymbol{T}$ ou $t_1^{\cap} = \bot^{\cap}$, alors $t_1^{\cap} \overset{\cap}{\lesssim} t_2^{\cap}$, et $\bot \in t_1^{\cap}$. Nous savons que $\pi_0 t_1^{\cap} \subseteq \pi_0 \bot^{\cap} = \varnothing$. Ainsi $\pi_{\{0\}\cup\boldsymbol{\alpha}^{\cap}(t_2^{\cap})}\delta(t_1^{\cap}) \subseteq \delta(t_2^{\cap})$.

Soit $t_1^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing - \bot^{\sqcap}$ et $t_2^{\cap} \in \boldsymbol{T}^{\cap} \setminus \varnothing$ tels que $t_1^{\cap} \overset{\cap}{\lesssim} t_2^{\cap}$.
Soit $r \in \delta(t_1^{\cap})$.
Soit $t_2 \in t_2^{\cap}$.
Par la proposition 7.3.6, et comme $t_x^{\cap}$ est ni absurde ni le TUC vide, alors : $\exists t_1 \in t_1^{\cap}, t_1 \lesssim t_2$.
Par la définition 89, $\pi_{\{0\}\cup\boldsymbol{\alpha}(t_1)} r \in \delta(\{t_1\})$.
Par la définition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)}\pi_{\{0\}\cup\boldsymbol{\alpha}(t_1)} r \in \delta(\{t_2\})$.
Par la proposition 7.1.3, $\{0\}\cup\boldsymbol{\alpha}(t_2) \subseteq \{0\}\cup\boldsymbol{\alpha}(t_1)$.
Ainsi $\pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)}\pi_{\{0\}\cup\boldsymbol{\alpha}(t_1)} r = \pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)} r \in \delta(\{t_2\})$.
Donc pour tout $t_2 \in t_2^{\cap}$, $\pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)} r \in \delta(\{t_2\})$.
Donc par la définition 89, $\pi_{\{0\}\cup\boldsymbol{\alpha}^{\cap}(t_2^{\cap})} r \in \delta(t_2^{\cap}))$.
Ainsi $\pi_{\{0\}\cup\boldsymbol{\alpha}^{\cap}(t_2^{\cap})}\delta(t_1^{\cap}) \subseteq \delta(t_2^{\cap})$.
Si $t_2^{\cap} = \top^{\cap}$, alors $\pi_{\{0\}\cup\boldsymbol{\alpha}^{\cap}(\top^{\cap})}\delta(t_1^{\cap}) \subseteq \delta(\top^{\cap})$.
Donc comme $\delta(\varnothing)\delta(\top^{\cap})$, alors si $t_2^{\cap} = \varnothing$, $\pi_{\{0\}\cup\boldsymbol{\alpha}^{\cap}(\top^{\cap})}\delta(t_1^{\cap}) \subseteq \delta(\varnothing)$. □

Démonstration de la Proposition 9.1.12 . Soit $t_0^{\cap} \in \bot^{\sqcap}$.
Par la proposition 7.3.5, il existe une liste de TUC $(t_0^{\cap}, \ldots, t_n^{\cap})$, avec $0 \leq n$, telle que tous les poins suivants sont vrais :

1. pour tout $0 \leq i < n$, un des cas suivants est vrai :
   a. $(t_{i+1}^{\cap}, t_i^{\cap}) \in \boldsymbol{C}_i^{\cap}$;

b. $\exists s_i \in \boldsymbol{\alpha}^{\cap}(t_i^{\cap}), \boldsymbol{\varsigma}_{t_i^{\cap}}^{\cap}(s_i) = t_{i+1}^{\cap}$;

2. un des cas suivants est vrai :
   a. $\exists t \in t_n^{\cap}, t \lesssim \bot$;
   b. $t_n^{\cap} \in \bot_A^{\sqcap}$;
   c. $\exists s \in \boldsymbol{S}_{\mathscr{T}}, t_n^{\cap} = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$.

Cas $n = 0$:
• Si $\exists t \in t_0^{\cap}, t \lesssim \bot$, Alors par la définition 85 et 88, $\pi_{\{0\}\cup\boldsymbol{\alpha}(\bot)}\delta(\{t\}) \subseteq \delta(\bot^{\cap}) = \varnothing$. Donc $\delta(\{t\}) = \varnothing$.
Nous savons que $\pi_{\{0\}\cup\boldsymbol{\alpha}(t)}\delta(t_0^{\cap}) \subseteq \delta(\{t\})$. Donc $\pi_{\{0\}\cup\boldsymbol{\alpha}(t)}\delta(t_0^{\cap}) = \varnothing$
Ainsi $\delta(t_0^{\cap}) = \varnothing$.
• Si $t_0^{\cap} \in \bot_A^{\sqcap}$, alors par la définition 90, $\delta(t_0^{\cap}) = \varnothing$.
• Si $\exists s \in \boldsymbol{S}_{\mathscr{T}}, t_0^{\cap} = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$.
Nous savons que $\pi_s\delta(t_0^{\cap}) \subseteq \pi_s\boldsymbol{\gamma_1}(s) \cap \pi_s\boldsymbol{\gamma_0}(s)$.
Et par la définition 86, $\pi_s\delta(t_0^{\cap}) \subseteq D \setminus \bullet \cap \{\bullet\} = \varnothing$. Donc $\delta(t_0^{\cap}) = \varnothing$.
Donc si la condition vaut pour $n = 0$, alors $\delta(t_0^{\cap}) = \varnothing$.

Supposons que la condition vaille pour n, alors $\delta(t_0^{\cap}) = \varnothing$.
Supposons qu'il existe une liste de TUC $(t_0^{\cap}, \ldots, t_{n+1}^{\cap})$, telle que tous les points suivants sont vrais :

1. pour tout $0 \leq i < n+1$, un des cas suivants est vrai,
   a. $(t_{i+1}^{\cap}, t_i^{\cap}) \in \boldsymbol{C}_i^{\cap}$;
   b. $\exists s_i \in \boldsymbol{\alpha}^{\cap}(t_i^{\cap}), \boldsymbol{\varsigma}_{t_i^{\cap}}^{\cap}(s_i) = t_{i+1}^{\cap}$;
2. un des cas suivants est vrai,
   a. $\exists t \in t_{n+1}^{\cap}, t \lesssim \bot$;
   b. $t_{n+1}^{\cap} \in \bot_A^{\sqcap}$;
   c. $\exists s \in \boldsymbol{S}_{\mathscr{T}}, t_{n+1}^{\cap} = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$.

Donc nous savons que la condition vaut pour n et la liste $(t_1^{\cap}, \ldots, t_{n+1}^{\cap})$. Ainsi $\delta(t_1^{\cap}) = \varnothing$.
• Si $(t_1^{\cap}, t_0^{\cap}) \in \boldsymbol{C}_0^{\cap}$, alors :
•• Si $(t_1^{\cap}, t_0^{\cap}) \in \boldsymbol{C}_{\lesssim}^{\cap}$, alors $\forall t_1 \in t_1^{\cap}, \exists t_0 \in t_x^{\cap} : t_0 \lesssim t_1$. Soit T la fonction qui associe à tout $t_1 \in t_1^{\cap}$ un tel $t_0 \in t_0^{\cap}$.
Par la proposition 9.1.8, $\pi_0\delta(t_0^{\cap}) = \bigcap_{t_0 \in t_0^{\cap}} \pi_0\delta(\{t_0\}) \subseteq \bigcap_{t_1 \in t_1^{\cap}} \pi_0\delta(\{T(t_1)\})$.
Comme $\forall t_1 \in t_1^{\cap}, T(t_1) \lesssim t_1$, et par la proposition 85, $\pi_{\{0\}\cup\boldsymbol{\alpha}(t_1)}\delta(\{T(t_1)\}) \subseteq \delta(\{t_1\})$.
Donc $\pi_0\delta(\{T(t_1)\}) \subseteq \delta(\{t_1\})$.
Par la proposition 9.1.8, $\bigcap_{t_1 \in t_1^{\cap}} \pi_0\delta(\{T(t_1)\}) \subseteq \bigcap_{t_1 \in t_1^{\cap}} \delta(\{t_1\}) = \delta(t_1^{\cap}) = \varnothing$.
Ainsi $\delta(t_0^{\cap}) = \varnothing$.
•• Si $(t_1^{\cap}, t_0^{\cap}) \in \boldsymbol{C}_{\top}^{\cap}$, alors $t_1^{\cap} = \top^{\cap}$ et $t_0^{\cap} = \varnothing$. Par la définition 89, $\delta(t_0^{\cap}) = \delta(t_1^{\cap}) = \varnothing$.
•• Si $(t_1^{\cap}, t_0^{\cap}) \in \boldsymbol{C}_{\bot}^{\cap}$, alors $t_0^{\cap} \in \bot_A^{\sqcap}$, et par la définition 90, $\delta(t_0^{\cap}) = \varnothing$..
•• Si $(t_1^{\cap}, t_0^{\cap}) \in \boldsymbol{C}_{\boldsymbol{\Gamma}}^{\cap}$, alors $\exists s \in \boldsymbol{S}_{\mathscr{T}}, t_0^{\cap} = \{\boldsymbol{\gamma_1}(s), \boldsymbol{\gamma_0}(s)\}$. Ceci correspond à un sous-cas de $n = 0$, et $\delta(t_0^{\cap}) = \varnothing$.
• Si $\exists s \in \boldsymbol{\alpha}^{\cap}(t_0^{\cap}), \boldsymbol{\varsigma}_{t_0^{\cap}}^{\cap}(s) = t_1^{\cap}$, alors par la proposition 9.1.10, $\pi_s\delta(t_0^{\cap}) \subseteq \pi_0\delta(t_1^{\cap}) = \varnothing$, donc $\delta(t^{\cap}) = \varnothing$.
Ainsi si la condition vaut pour $n+1$, alors $\delta(t_0^{\cap}) = \varnothing$.

Ainsi si la condition vaut pour n, alors $\delta(t_0^{\cap}) = \varnothing$. □

Démonstration de la Proposition 9.1.13 . Soit $G = \langle U, \boldsymbol{I}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$, et soit $u \in U$ tel que $type(u) \in \bot^{\sqcap}$. Soit (D, δ, β) un modèle de G. L'assignation $\beta(u)$ devrait être une valeur pour l'attribut 0 de $\delta(\{\bot\})$, or cette interprétation est vide par la proposition 9.1.12. □

Démonstration de la Proposition 9.3.1 . Soit $G \in \mathscr{G}(\mathscr{S})$ un GU clos.
Les règles **eq-ref**, **eq-sym**, et **eq-trans** ne sont pas applicables, donc la relation Eq est réflexive, symétrique et transitive. □

Démonstration de la Proposition 9.3.2 . Soit $G = \langle U, \boldsymbol{I}, A, C, Eq \rangle \in \mathscr{G}(\mathscr{S})$ un GU clos.
Nous devons montrer que le modèle (D, δ) est un modèle du support $\mathscr{S}$, et qu'il satisfait le GU G.
Nous savons déjà que $\forall m \in \boldsymbol{M}, \delta(m) \in D$.

Montrons que (D, δ) est un modèle de $\mathscr{T}$.
Définition 85 :
Vérifions que 0 est une clé unique. $\delta(\{t\})$, qui possède un tuple r pour chaque classe d'équivalence $[u]$. $r(0) = [u]$ et est donc unique.
Définition 85 : Soit $t_1 \lesssim t_2$. r un tuple de $\delta(\{t_1\})$, et soit u un membre de $r(0)$. $t_1 \in type(u)$. Par ailleurs, G est clos, donc l'inapplicabilité de la règle **typ-comp** implique que $t_2 \in type(u)$. Donc il existe également un tuple r' dans $\delta(\{t_2\})$ qui possède les mêmes valeurs que r pour leurs attributs communs.
Définition 86 :
Racine de PosAInt : Soit $s \in \boldsymbol{S}_{\mathscr{T}}$. Soit r un tuple de $\delta(\{\boldsymbol{\gamma_1}(s)\})$, et soit u un membre de $r(0)$. G est clos, donc l'inapplicabilité de la règle **a-oblaslot-root** impose qu'il existe $(u, s, v) \in A$. Donc $r(s) \neq \bullet$
Racine de PosAInt : Soit $s \in \boldsymbol{S}_{\mathscr{T}}$. Soit r un tuple de $\delta(\{\boldsymbol{\gamma_0}(s)\})$, et soit u un membre de $r(0)$. G est clos, donc l'inapplicabilité de la règle **a-int** impose :

- Soit $\bot \in type(u)$, auquel cas G est absurde et n'est pas clos, ce qui est une contradiction ;
- Soit il n'existe pas de $(u, s, v) \in A$. Donc $r(s) = \bullet$.

Définition 87 :
Soit $t \in \boldsymbol{T}$ et $s \in \boldsymbol{\alpha}(t)$. Soit r un tuple de $\delta(\{t\})$, et soit u un membre de $r(0)$. G est clos, donc l'inapplicabilité de la règle **a-sig** impose que :

- Soit il existe $(u, s, v) \in A$. $r(s) = [v]$, et $\boldsymbol{\varsigma}_t(s) \subseteq type(v)$. Soit r' le tuple de $\delta(type(v))$, tel que $r'(0) = [v]$. Puisque $\pi_0\delta(type(v)) \subseteq \pi_0\delta(\boldsymbol{\varsigma}_t(s))$, alors il existe r'' dans $\delta(\boldsymbol{\varsigma}_t(s))$ tel que $r''(0) = [v]$, et $r(s) = r''(0)$.
- Soit il n'existe pas de $(u, s, v) \in A$. Donc $r(s) = \bullet$.

Donc $\pi_s\delta(\{t\}) \setminus \bullet \subseteq \pi_0\delta(\boldsymbol{\varsigma}_t(s))$.
Définition 88 :
$\top$ **:** L'inapplicabilité de la règle **a-top** impose que pour tout $u \in U$, $\top \in type(u)$. Donc il existe un tuple pour tout élément de $U/Eq = D \setminus \bullet$, et seulement pour ceux là.
$\bot$ **:** s'il existe $u \in U$ tel que $\bot \in type(u)$, alors G est absurde et n'est pas clos, ce qui est une contradiction.
Définition 90 :
L'inapplicabilité de la règle **typ-absurd** pour tout $t_a^{\cap} \in \bot_A^{\sqcap}$ impose que pour tout $u \in U$,

- Soit $\bot \in type(u)$, auquel cas G est absurde et n'est pas clos, ce qui est une contradiction ;
- Soit il n'existe pas de $u \in U$ tel que $t_a^{\cap} \subseteq type(u)$. Puisque $\pi_0\delta(t_a^{\cap}) \subseteq \bigcap_{t \in t_a^{\cap}} \pi_0\delta(\{t\})$, alors $\delta(t_a^{\cap}) = \varnothing$.

Donc (D,δ) est bien un modèle de $\mathscr{T}$.
Montrons maintenant que (D,δ) est un modèle de $\mathscr{C}$.

Définition 93 :
Soit $s_1,s_2 \in \boldsymbol{S}_{\mathscr{C}}$. Soit $(g,c) \in \delta(s_1)$. Il existe donc $(u,s,v) \in C$ tel que $g=[u]$, et $c=[v]$. G est clos, donc l'inapplicabilité de la règle **c-comp** impose qu'il existe $(u,s_2,v) \in C$. Donc $(g,c) \in \delta(s_2)$, et $\delta(s_1) \subseteq \delta(s_2)$.
Définition 94 :
Soit $s \in \boldsymbol{S}_{\mathscr{C}}$. Soit $(g,c) \in \delta(s)$. Il existe donc $(u,s,v) \in C$ tel que $g=[u]$, et $c=[v]$. G est clos, donc l'inapplicabilité de la règle **c-sig** impose que $domaine(s) \subseteq type(u)$, et $codomaine(s) \subseteq type(v)$.
Soit r_{gov} le tuple de $\delta(type(u))$, tel que $r_{gov}(0)=[u]$. Puisque $\pi_0\delta(type(u)) \subseteq \pi_0\delta(domaine(s))$, alors il existe r'_{gov} dans $\delta(domaine(s))$ tel que $r'_{gov}(0)=[u]$, et $g=r'_{gov}(0)$.
Soit r_{circ} le tuple de $\delta(type(v))$, tel que $r_{circ}(0)=[v]$. Puisque $\pi_0\delta(type(v)) \subseteq \pi_0\delta(codomaine(s))$, alors il existe r'_{circ} dans $\delta(codomaine(s))$ tel que $r'_{circ}(0)=[v]$, et $g=r'_{circ}(0)$.

Donc (D,δ) est bien un modèle de $\mathscr{C}$.
Montrons maintenant que $(D,\delta,[\cdot])$ satisfait G.

Définition 9.30 :
Soit $u \in U$ et $m \in marker(u)$, $\beta(u)=[u]=\delta(m)$.
Définition 9.31 :
Soit $u \in U$, $\beta(u)=[u]$, et pour tous les TUP $t \in type(u)$, il existe un tuple $r_t \in \delta(\{t\})$ avec les mêmes valeurs pour les attributs communs, en particulier $r_t(0)=[u]$. Donc il existe un tuple $r \in \delta(type(u))$ avec $r(0)=[u]$. Donc $\beta(u) \in \pi_0\delta(type(u))$
Définition 9.32 :
Soit $(u,s,v) \in A$. G est clos, donc l'inapplicabilité de **a-aslot-root** impose que $\boldsymbol{\gamma}(s) \in type(u)$. Il existe un type $r \in \delta(\{\boldsymbol{\gamma}(s)\})$ tel que $r(0)=[u]$, et $r(s)=[v]$. Par ailleurs, $\beta(u)=[u]$, et $\beta(v)=[v]$. Donc $(\beta(u),\beta(v)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Définition 9.33 :
Soit $(u,s,v) \in C$. $([u],[v]) \in \delta(s)$. Par ailleurs, $\beta(u)=[u]$, et $\beta(v)=[v]$. Donc $(\beta(u),\beta(v)) \in \delta(s)$.
Définition 9.34 :
Soit $(u,v) \in Eq$. $\beta(u)=[u]=[v]=\beta(v)$.

En conclusion, le modèle canonique de G satisfait bien G, $(D,\delta,[\cdot]) \vDash G$. □

Démonstration de la Proposition 9.3.3 . Soit $G=\langle U,\boldsymbol{l},A,C,Eq\rangle \in \mathscr{G}(\mathscr{S})$ et $u \in U$.
Supposons que $cl_2(G_i)$ ajoute un nœud d'unité v et un triplet (u,s,v) dans G_i. Le type de v est l'ensemble vide au début.
Ensuite la règle **a-sig** est applicable à u,v, et impose que $\boldsymbol{\varsigma}^{\cap}_{type(u)}(s) \subseteq type(v)$.
La règle **typ-top** est applicable à v, et impose que $\top^{\cap} \subseteq type(v)$, donc que $type(v)$ est non-vide.
Les règles **typ-comp** imposent que $type(v)$ est un ensemble supérieur de TUP.
Puisque v n'est connecté au reste du graphe que par $(u,s,v) \in A$, les seules autres règles qui peuvent avoir un effet sur le type de v le rendraient absurde. Or, si le type de v est absurde, alors $G_\perp$ peut être déduit de G, ce qui contredit notre hypothèse.
Donc le type de v est l'ensemble supérieur non-absurde et non-vide de TUP généré par $\boldsymbol{\varsigma}^{\cap}_{type(u)}(s)$.
Par la proposition 7.3.19, il s'agit donc d'un TUC clos : $type(v)=\Uparrow\boldsymbol{\varsigma}^{\cap}_{type(u)}(s)$. □

Démonstration de la Proposition 9.3.4 . Si $i \geq 1$, alors G_i est partiellement clos.
cl_2 ajoute seulement des couples d'un nœud d'unité et d'un triplet actanciel. Soit $(u', s, v) \in A(cl_2(G_i)) - A_{i-1}$ un tel triplet actanciel, avec $u' \in U(cl_2(G_i)) - U_{i-1}$.
Alors parmi les règles de cl_1, et parmi les nœuds d'unité de U_i, la seule configuration pour laquelle le type de u est modifiée est si $u = u'$: la règle **a-aslot-root** impose que $type_{i+1}(u) = type_i(u) \cup \{\boldsymbol{\gamma}(s)\}$.
Cependant, nous savons que $\boldsymbol{\gamma_1}(s) \in type_i(u)$, que $type_i(u)$ est clos, et que $\boldsymbol{\gamma_1}(s) \lesssim \boldsymbol{\gamma}(s)$. Donc $\boldsymbol{\gamma}(s)$ appartient également à $type_i(u)$. Ainsi $type_i(u) = type_{i+1}(u)$. □

Démonstration du Lemme 9.3.5 . Soit $i \geq 1$, et $u_{i+1} \in U_{i+1} - U_i$.
u_{i+1} peut seulement avoir été introduit par la règle **a-oblaslot-root**. Ainsi il existe $u_i \in U_i$, $s \in \boldsymbol{S}_{\mathscr{T}}$, tel que $\boldsymbol{\gamma}(s) \in type_i(u_i)$, et $(u_i, s_i, u_{i+1}) \in A_{i+1} - A_i$.
Si $u_i \in U_{i-1}$, alors par la proposition 9.3.4, $type_{i-1}(u_i) = type_i(u_i)$, donc $\boldsymbol{\gamma}(s_i) \in type_{i-1}(u_i)$, et $\exists v \in U_i$ tel que $(u_i, s_i, v) \in A_{i-1}$. Cette règle **a-oblaslot-root** n'aurait pas été applicable pour u_i et s_i. Ceci contredit notre hypothèse. Ainsi $u_i \notin U_{i-1}$. □

Démonstration of Théorème 9.3.6 . *Condition nécessaire.* Supposons que $G_{\mathscr{T}}$ possède un cycle, alors il existe une boucle de TUC clos $\{t_0^{\cap}, \dots t_n^{\cap}\}$, tels que $t_0^{\cap} = t_n^{\cap}$, et $\forall i < n, type(t_{i+1}^{\cap}) = \Uparrow \boldsymbol{\varsigma}_{t_i^{\cap}}^{\cap}(s)$.
Considérons le GU $G = (u_0, \boldsymbol{l}, \varnothing, \varnothing, \varnothing)$ tel que $type(u_0) = t_0^{\cap}$. Alors pour tout $i \geq 2$, G_i introduit un nœud d'unité u_i avec $type_i(u_i) = t_{i-1\%n}^{\cap}$. Ainsi l'expansion de G est infinie. Ainsi si tout GU fini possède une expansion finie, alors $G_{\mathscr{T}}$ est acyclique.
Condition suffisante. Supposons qu'il existe un GU fini ayant une expansion infinie. Alors G_{i+1} diffère toujours de G_i par au moins un nouveau nœud d'unité u_{i+1} et un nouveau triplet actanciel (u_i, s_i, u_{i+1}). Et G_i diffère toujours de G_{i-1} par un nœud d'unité u_i. Ainsi il existe une suite infinie de nœuds d'unité (u_i) dans $cl(G)$ tels que

$$\forall i \geq 2, u_i \in U_i - U_{i-1} \tag{A.44}$$
$$\forall i \geq 2, type(u_i) = \Uparrow \boldsymbol{\varsigma}_{type(u_{i-1})}^{\cap}(s) \tag{A.45}$$

Cependant le nombre de TUC clos est fini car l'ensemble des TUP est fini. Ainsi il existe une boucle de TUC clos $\{t_0^{\cap}, \dots t_n^{\cap}\}$, telle que $t_0^{\cap} = t_n^{\cap}$, et $\forall i < n, type(t_{i+1}^{\cap}) = \Uparrow \boldsymbol{\varsigma}_{t_i^{\cap}}^{\cap}(s)$. Ou de manière équivalente, $G_{\mathscr{T}}$ possède un cycle.
Ainsi, si $G_{\mathscr{T}}$ est acyclique, alors tout GU fini possède une expansion finie.

□

Démonstration du Lemme 9.3.7 . Soit $G = \langle U^g, \boldsymbol{l}^g, A^g, C^g, Eq^g \rangle$ et $H = \langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h \rangle$ deux GU définis sur le même support $\mathscr{S}$, et π un homomorphisme de H vers G
Soit (D, δ) un modèle de $\mathscr{S}$.
Soit $\beta^g : U^g \to D \setminus \bullet$ une affectation des nœuds d'unité de G telle que $(D, \delta, \beta^g) \models G$.
Considérons $\beta^h : U^h \to D \setminus \bullet$ une affectation des nœuds d'unité de H telle que $\beta^h(u) = \beta^g \circ \pi(u)$.
Montrons que (D, δ, β^h) satisfait H, i.e., que β^h satisfait les équations 9.30 à 9.34.
Eq. 9.30 : Soit $u \in U^h$ et $m \in marker^h(u)$.
Par la définition 68, $m \in marker^g(\pi(u))$
Ainsi par l'équation 9.30, $\beta^g(\pi(u)) = \delta(m)$.
Ainsi $\beta^h(u) = \delta(m)$.
Eq. 9.31 : Soit $u \in U^h$.
$\beta^h(u) = \beta^g(\pi(u)) \in \pi_0 \delta(type^g(\pi(u)))$.
Par la définition 68, $type^g(\pi(u)) \overset{\cap}{\lesssim} type^h(u)$.
Si $type^g(\pi(u)) \in \bot^{\sqcap}$, alors par la proposition 9.1.12, $\delta(type^g(\pi(u))) = \varnothing$. Cela mène à une contra-

diction, et implique qu'il n'existe aucun modèle qui satisfasse G, donc G est absurde.
Sinon, par la proposition 9.1.11, $\pi_0\delta(type^g(\pi(u))) \subseteq \pi_0\delta(type^h(u))$.
Ainsi $\beta^h(u) \in \pi_0\delta(type^h(u))$
Eq. 9.32 : Soit $(u,s,v) \in A^h$.
$(\pi(u),s,\pi(v)) \in A^g$, donc $\pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\}) = \{(\beta^g(\pi(u)),\beta^g(\pi(u)))\}$.
Ainsi, $\pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\}) = \{(\beta^h(u),\beta^h(u))\}$.
Eq. 9.33 : Soit $(u,s,v) \in C^h$.
$\exists c \in C^g, (\pi(u),\pi(v)) = arc(c)$, et $symbol(c) \overset{\mathscr{C}}{\lesssim} s$.
Par ailleurs, $(\beta^g(\pi(u)),\beta^g(\pi(v))) \in \delta(symbol^h(c))$.
Comme $symbol^h(c) \overset{\mathscr{C}}{\lesssim} s$, alors $\delta(symbol^h(c)) \subseteq \delta(s)$.
Ainsi $(\beta^h(u),\beta^h(v)) \in \delta(s)$.
Eq. 9.34 : Soit $(u_1,u_2) \in Eq^h$.
$\beta^g(\pi(u_1)) = \beta^g(\pi(u_2))$, ainsi $\beta^h(u_1) = \beta^h(u_2)$.

□

Démonstration du Lemme 9.3.8 . Soient $G = \langle U,\boldsymbol{I},A,C,Eq\rangle$, et $G' = \langle U',\boldsymbol{I}',A',C',Eq'\rangle$ deux GU définis sur le support $\mathscr{S}$. Montrons que si G' est une dérivation immédiate de G, alors $G \vDash G'$. Soit $(D,\delta,\beta) \vDash G$.
Nous étudions séquentiellement l'application de chacune des règles de la base de règles axiomatiques d'inférence $\mathscr{R}(\mathscr{S})$.
Si **mrk-ref** est appliqué à $u \in U$. Trivialement, $\beta(u) = \beta(u)$.
Dans G', $(u,u) \in Eq'$, et $\beta(u) = \beta(u)$, donc β satisfait l'équation 9.34, et $(D,\delta,\beta) \vDash G'$.
Si **eq-sym** est appliqué à $u,v \in U$, on sait que $(u,v) \in Eq$.
Par l'équation 9.34, $\beta(u) = \beta(v)$.
Dans G', $(v,u) \in Eq'$, et $\beta(v) = \beta(u)$, donc β satisfait l'équation 9.34, et $(D,\delta,\beta) \vDash G'$.
Si **eq-trans** est appliqué à $u,v,w \in U$, on sait que $(u,v) \in Eq$, et $(v,w) \in Eq$.
Par l'équation 9.34, $\beta(u) = \beta(v) = \beta(w)$.
Dans G', $(u,w) \in Eq'$, et $\beta(u) = \beta(w)$, donc β satisfait l'équation 9.34, et $(D,\delta,\beta) \vDash G'$.
Si **mrk-eq** est appliqué à $u,v \in U$, on sait que $m \in marker(u) \cap marker(v)$.
Par l'équation 9.30, on sait que $\delta(m) = \beta(u)$, et $\delta(m) = \beta(v)$.
Dans G', $(u,v) \in Eq'$, et $\beta(u) = \beta(w)$, donc β satisfait l'équation 9.34, et $(D,\delta,\beta) \vDash G'$.
Si **eq-mrk** est appliqué à $u,v \in U$, on sait que $m \in marker(u)$, et $(u,v) \in Eq$.
Par les équations 9.30 et 9.34, $\delta(m) = \beta(u)$, et $\beta(u) = \beta(v)$.
Dans G', $m \in marker'(v)$, et $\delta(m) = \beta(v)$, donc β satisfait l'équation 9.30, et $(D,\delta,\beta) \vDash G'$.
Si **eq-typ** est appliqué à $u,v \in U$, on sait que $t \in type(u)$, et $(u,v) \in Eq$.
Par les équations 9.31 et 9.34, $\beta(u) \in \pi_0\delta(type(u))$, $\beta(v) \in \pi_0\delta(type(v))$, et $\beta(u) = \beta(v)$.
Par la proposition 9.1.8, $\pi_0\delta(type(u)) \subseteq \pi_0\delta(\{t\})$. Donc $\beta(v) \in \pi_0\delta(\{t\})$. Dans G', $type'(v) = type(v) \cup \{t\}$. Par la proposition 9.1.8, $\pi_0\delta(type'(v)) = \pi_0\delta(type(v)) \cap \pi_0\delta(\{t\})$.
Donc $\beta(v) \in \pi_0\delta(type'(v))$, β satisfait l'équation 9.31, et $(D,\delta,\beta) \vDash G'$.
Si **typ-top** est appliqué à $u \in U$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la définition 88 et la proposition 9.1.8, $\pi_0\delta(type(u) \cup \{\top\}) = \pi_0\delta(type(u)) \cap D \setminus \bullet = \pi_0\delta(type(u))$.
Dans G', $type'(u) = type(u) \cup \{\top\}$, donc $\beta(u) \in \pi_0\delta(type'(u))$, β satisfait l'équation 9.31, et $(D,\delta,\beta) \vDash G'$.
Si **typ-comp** est appliqué à $u \in U$ pour $(t_2,t_1) \in \boldsymbol{C_T}$. $t_1 \in type(u)$.

Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u)) \subseteq \pi_0\delta(\{t_1\})$. Donc $\beta(u) \in \pi_0\delta(\{t_1\})$.
Par la définition 85, $\pi_0\delta(\{t_1\}) \subseteq \pi_0\delta(\{t_2\})$. Donc $\beta(u) \in \pi_0\delta(\{t_2\})$.
Par la proposition 9.1.8, $\pi_0\delta(type(u) \cup \{t_2\}) = \pi_0\delta(type(u)) \cap \pi_0\delta(\{t_2\})$. Donc $\beta(u) \in \pi_0\delta(type(u) \cup \{t_2\})$.
Dans G', $type'(u) = type(u) \cup \{t_2\}$, donc $\beta(u) \in \pi_0\delta(type'(u))$, β satisfait l'équation 9.31, et $(D,\delta,\beta) \vDash G'$.
Si **typ-absurd** est appliqué à $u \in U$ pour $t_a^{\cap} \in \bot_A^{\sqcap}$. $t_a^{\cap} \in type(u)$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u)) \subseteq \pi_0\delta(t_a^{\cap})$. Donc $\beta(u) \in \pi_0\delta(t_a^{\cap})$.
Par la définition 90, $\pi_0\delta(t_a^{\cap}) = \varnothing$. Cela mène à une contradiction, et implique qu'il n'existe aucun modèle qui satisfasse G, donc G est absurde.
Si **a-eq-g** est appliqué à $u_1, u_2, v \in U$, $(u_1,u_2) \in Eq$, et $(u_1,s,v) \in A$.
Par les équations 9.34 et 9.32, $\beta(u_1) = \beta(u_2)$, et $(\beta(u_1),\beta(v)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Dans G', $(u_2,s,v) \in A'$, et $(\beta(u_2),\beta(v)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$, donc β satisfait l'équation 9.32, et $(D,\delta,\beta) \vDash G'$.
Si **a-eq-a** est appliqué à $u, v_1, v_2 \in U$, $(v_1,v_2) \in Eq$, et $(u,s,v_1) \in A$.
Par les équations 9.34 et 9.32, $\beta(v_1) = \beta(v_2)$, et $(\beta(u),\beta(v_1)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Dans G', $(u,s,v_2) \in A'$, et $(\beta(u),\beta(v_2)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$, donc β satisfait l'équation 9.32, et $(D,\delta,\beta) \vDash G'$.
Si **a-fp** est appliqué à $u, v_1, v_2 \in U$, $(u,s,v_1) \in A$, et $(u,s,v_2) \in A$.
Par l'équation 9.32, $(\beta(u),\beta(v_1)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$, et $(\beta(u),\beta(v_2)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par la proposition 9.1.6, $\beta(v_1) = \beta(v_2)$.
Dans G', $(v_1,v_2) \in Eq'$, et $\beta(v_1) = \beta(v_2)$, donc β satisfait l'équation 9.34, et $(D,\delta,\beta) \vDash G'$.
Si **a-aslot-root** est appliqué à $u, v \in U$, $(u,s,v) \in A$.
Par l'équation 9.32, $(\beta(u),\beta(v_1)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$. Donc $\beta(u) \in \pi_0\delta(\{\boldsymbol{\gamma}(s)\})$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u) \cup \{\boldsymbol{\gamma}(s)\}) = \pi_0\delta(type(u)) \cap \pi_0\delta(\{\boldsymbol{\gamma}(s)\})$. Donc $\beta(u) \in \pi_0\delta(type(u) \cup \{\boldsymbol{\gamma}(s)\})$.
Dans G', $type'(u) = type(u) \cup \{\boldsymbol{\gamma}(s)\}$, donc $\beta(u) \in \pi_0\delta(type'(u))$, β satisfait l'équation 9.31, et $(D,\delta,\beta) \vDash G'$.
Si **a-oblaslot-root** est appliqué à $u \in U$, $\boldsymbol{\gamma_1}(s) \in type(u)$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u)) \subseteq \pi_0\delta(\{\boldsymbol{\gamma_1}(s)\})$. Donc $\beta(u) \in \pi_0\delta(\{\boldsymbol{\gamma_1}(s)\})$.
On sait que s est un attribut de $\delta(\{\boldsymbol{\gamma_1}(s)\})$, et que 0 est une clé unique. Posons donc $(\beta(u),a) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma_1}(s)\})$.
Par la définition 85, $\pi_{0,s}\delta(\{\boldsymbol{\gamma_1}(s)\}) \subseteq \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$. Donc $(\beta(u),a) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par la définition 86, on sait que $a \in D \setminus \bullet$.
Dans G', $(u,s,v) \in A'$. Considérons β' tel que $\beta'(u) = \beta(u)$ pour tout $u \neq v$, et $\beta'(v) = a$.
Alors $(\beta(u),\beta(v)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$, β satisfait l'équation 9.32, et $(D,\delta,\beta) \vDash G'$.
Si **a-proaslot-root** est appliqué à $(u,s,v) \in A$, $\boldsymbol{\gamma_0}(s) \in type(u)$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(\{\boldsymbol{\gamma_0}(s)\})$.
Par l'équation 9.32, $\{\beta(u),\beta(v)\} \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
On sait que s est un attribut de $\delta(\{\boldsymbol{\gamma_1}(s)\})$, et que 0 est une clé unique. Posons donc $(\beta(u),a) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma_1}(s)\})$.
Par la définition 85, $\pi_{0,s}\delta(\{\boldsymbol{\gamma_0}(s)\}) \subseteq \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$. Donc $(\beta(u),a) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par la définition 84, $a = \beta(v) \in D \setminus \bullet$. Or, par la définition 86, $a = \bullet$. Cela mène à une contradiction,

et implique qu'il n'existe aucun modèle qui satisfasse G, donc G est absurde.
Si **a-sig** est appliqué à $(u,s,v) \in A$, $t \in type(u)$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u)) \subseteq \pi_0\delta(\{t\})$. Donc $\beta(u) \in \pi_0\delta(\{t\})$.
Par l'équation 9.32, $\{\beta(u),\beta(v)\} \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Par la définition 85, $\pi_{0,s}\delta(\{t\}) \subseteq \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$. Et on sait que 0 est une clé unique, donc $\{\beta(u),\beta(v)\} \in \pi_{0,s}\delta(\{t\})$.
Donc $\beta(v) \in \pi_s\delta(\{t\})$. Par ailleurs, $\beta(v) \neq \bullet$. Ainsi par la définition 87, $\beta(v) \in \pi_0\delta(\boldsymbol{\varsigma}_t(s))$.
Par la proposition 9.1.8, $\pi_0\delta(type(v) \cup \boldsymbol{\varsigma}_t(s)) = \pi_0\delta(type(v)) \cap \pi_0\delta(\boldsymbol{\varsigma}_t(s))$. Donc $\beta(v) \in \pi_0\delta(type(v) \cup \boldsymbol{\varsigma}_t(s))$.
Dans G', $type'(v) = type(v) \cup \boldsymbol{\varsigma}_t(s)$, donc $\beta(v) \in \pi_0\delta(type'(v))$, β satisfait l'équation 9.31, et $(D,\delta,\beta) \models G'$.
Si **c-eq-g** est appliqué à $u_1,u_2,v \in U$, $(u_1,u_2) \in Eq$, et $(u_1,s,v) \in C$.
Par les équations 9.34 et 9.33, $\beta(u_1) = \beta(u_2)$, et $(\beta(u_1),\beta(v)) \in \delta(s)$.
Dans G', $(u_2,s,v) \in C'$, et $(\beta(u_2),\beta(v)) \in \delta(s)$, donc β satisfait l'équation 9.33, et $(D,\delta,\beta) \models G'$.
Si **c-eq-c** est appliqué à $u,v_1,v_2 \in U$, $(v_1,v_2) \in Eq$, et $(u,s,v_1) \in C$.
Par les équations 9.34 et 9.33, $\beta(v_1) = \beta(v_2)$, et $(\beta(u),\beta(v_1)) \in \delta(s)$.
Dans G', $(u,s,v_2) \in C'$, et $(\beta(u),\beta(v_2)) \in \delta(s)$, donc β satisfait l'équation 9.33, et $(D,\delta,\beta) \models G'$.
Si **c-sig** est appliqué à $(u,s,v) \in C$.
Par l'équation 9.31, $\beta(u) \in \pi_0\delta(type(u))$ et $\beta(v) \in \pi_0\delta(type(v))$.
Par l'équation 9.33 et $(\beta(u),\beta(v)) \in \delta(s)$.
Par la définition 94, $\beta(u) \in \pi_0\delta(domaine(s))$, et $\beta(v) \in \pi_0\delta(codomaine(s))$. Par la proposition 9.1.8, $\pi_0\delta(type(u) \cup domaine(s)) = \pi_0\delta(type(u)) \cap \pi_0\delta(domaine(s))$. Donc $\beta(u) \in \pi_0\delta(type(u) \cup domaine(s))$.
Par la proposition 9.1.8, $\pi_0\delta(type(v) \cup codomaine(s)) = \pi_0\delta(type(v)) \cap \pi_0\delta(codomaine(s))$. Donc $\beta(v) \in \pi_0\delta(type(v) \cup codomaine(s))$.
Dans G', $type'(u) = type(u) \cup domaine(s)$ et $type'(v) = type(v) \cup codomaine(s)$. Donc $\beta(u) \in \pi_0\delta(type(u))$ et $\beta(v) \in \pi_0\delta(type(v))$, donc β satisfait l'équation 9.33, et $(D,\delta,\beta) \models G'$.
Si **c-comp** est appliqué à $u,v \in U$ pour $(s_2,s_1) \in \boldsymbol{C}_{\boldsymbol{S}_{\mathscr{C}}}$.
Par l'équation 9.33, $(\beta(u),\beta(v)) \in \delta(s_1)$.
Par la définition 93, $\delta(s_1) \subseteq \delta(s_2)$. Donc $(\beta(u),\beta(v)) \in \delta(s_2)$.
Dans G', $(u,s_2,v) \in C'$, et $(\beta(u),\beta(v)) \in \delta(s_2)$. Donc β satisfait l'équation 9.33, et $(D,\delta,\beta) \models G'$.
Si une règle **def+** est appliquée,
Soit $D_t^- = \{u_t^-,v_1^-,\ldots,v_n^-\}(U^-,\boldsymbol{I}^-,A^-,\varnothing,\varnothing)$
Soit $D_t^+ = \{u_t^+,v_1^+,\ldots,v_n^+\}(U^+,\boldsymbol{I}^+,A^+,\varnothing,\varnothing)$
Il existe un homomorphisme π de D_t^- vers G.
Par la proposition 9.3.7, $(D,\delta,\beta \circ \pi) \models D_t^-$.
Comme (D,δ) satisfait (D_t^-,D_t^+,κ), alors par les définitions 100 et 101, il existe β^+ tel que $(D,\delta,\beta^+) \models D_t^+$, et $\forall u \in \{u_t^-,v_1^-,\ldots,v_n^-\}, \beta^+ \circ \kappa(u) = \beta \circ \pi(u)$.
Considérons l'addition de D_t^+ à G.

$$\text{Soit } \beta' \begin{cases} \beta'(u) = \beta^+(u) \text{ si } u \in U^+ \\ \beta'(u) = \beta(u) \text{ si } u \in U \end{cases} \tag{A.46}$$

(D,δ,β') satisfait l'addition de D_t^+ à G.
Nous devons fusionner les nœuds d'unité. Soit $(u^+,u^i) \in \pi \circ \kappa^{-1}$.
$\kappa^{-1}(u^+) \in \{u_t^-,v_1^-,\ldots,v_n^-\}$, donc $\beta^+ \circ \kappa \circ \kappa^{-1}(u^+) = \beta \circ \pi \circ \kappa^{-1}(u^+) = \beta(u^i)$. Comme κ est une

bijection, $\kappa \circ \kappa^{-1} = Id$. Ainsi $\beta^+(u^+) = \beta(u^i)$.
Fusionnons u^+ et u^i, en un nouveau nœud w. Soit $\beta'(w) = \beta^+(u^+) = \beta(u^i)$.
Équation 9.30 : Nous savons que $\forall m \in marker^+(u^+), \delta(m) = \beta^+(u^+) = \beta'(w)$ et $\forall m \in marker(u^i), \delta(m) = \beta(u^i) = \beta'(w)$.
Ainsi $\forall m \in marker(w) = marker^+(u^+) \cup marker(u^i), \delta(m) = \beta'(w)$.
Équation 9.31 : Nous savons que $\beta^+(u^+) \in \pi_0\delta(type^+(u^+))$ et $\beta(u^i) \in \pi_0\delta(type(u^i))$.
Par la proposition 9.1.8, $\pi_0\delta(type'(w)) = \pi_0\delta(type^+(u^+) \cup type(u^i)) = \pi_0\delta(type^+(u^+)) \cap \pi_0\delta(type^i(u^i))$, donc $\beta'(w) \in \pi_0\delta(type'(w))$.
Équation 9.32 :
• Pour tout $(w,s,v) \in A'$, soit $(u^+,s,v) \in A^+$, ou $(u^i,s,v) \in A$. Dans tous les cas, $(\beta(u^+) = \beta(u^i) = \beta(w), \beta(v)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
• Pour tout $(u,s,w) \in A'$, soit $(u,s,u^+) \in A^+$, ou $(u,s,u^i) \in A$. Dans tous les cas, $(\beta(u), \beta(u^+) = \beta(u^i) = \beta(w)) \in \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\})$.
Équation 9.33 :
• Pour tout $(w,s,v) \in C'$, soit $(u^+,s,v) \in C^+$, ou $(u^i,s,v) \in C$. Dans tous les cas, $(\beta(u^+) = \beta(u^i) = \beta(w), \beta(v)) \in \delta(s)$.
• Pour tout $(u,s,w) \in C'$, soit $(u,s,u^+) \in C^+$, ou $(u,s,u^i) \in C$. Dans tous les cas, $(\beta(v), \beta(u^+) = \beta(u^i) = \beta(w)) \in \delta(s)$.
Équation 9.34 :
• Pour tout $(w,v) \in Eq'$, soit $(u^+,v) \in Eq^+$, ou $(u^i,v) \in Eq$. Dans tous les cas, $\beta'(v) = \beta^+(u^+) = \beta(u^i) = \beta'(w)$.
• Pour tout $(u,w) \in Eq'$, soit $(u,u^+) \in Eq^+$, ou $(u,u^i) \in Eq$. Dans tous les cas, $\beta'(u) = \beta^+(u^+) = \beta(u^i) = \beta'(w)$.
Ainsi, nous avons $(D,\delta,\beta') \vDash G'$.
Si une règle **def-** est appliquée,
La démonstration de ce point est la même que pour la règle **def+**, où $+$ et $-$ sont inversés.
Conclusion de la preuve. Ainsi dans tous les cas, il existe une affectation β' telle que $(D,\delta,\beta') \vDash G'$. Donc $G \vDash G'$. En réutilisant ce résultat pour toute une séquence de dérivations immédiates, nous prouvons le résultat final. □

Démonstration of Théorème 9.3.9 . Si H est déductible de G, alors il existe un homomorphisme de H vers une dérivation G' de G.
Ainsi par le lemme 9.3.7, $G' \vDash H$.
Et par le lemme 9.3.8, nous savons que $G \vDash G'$.
Ainsi $G \vDash H$. □

Démonstration du Lemme 9.3.10 . Soient $G = \langle U, \boldsymbol{I}, A, C, Eq \rangle$, et $G' = \langle U', \boldsymbol{I}', A', C', Eq' \rangle$ deux GU définis sur le support $\mathscr{S}$. Montrons que si G' est une dérivation immédiate de G, alors $G' \vDash G$. Soit $(D,\delta,\beta') \vDash G'$, et considérons la restriction de l'assignation $\beta = \beta'|_U$.
Si la règle a ajouté une relation d'équivalences déclarées, une relation actancielle, une relation circonstancielle, ou un marqueur dans une étiquette d'un nœud d'unité, les conditions de satisfaction de G par (D,δ,β) sont plus lâches que pour G'.
Si la règle a augmenté d'un TUC $t^\cap$ le type d'un nœud d'unité u, i.e., $type'(u) = type(u) \cup t^\cap$, nous savons que $\beta'(u) \in \pi_0\delta(type'(u))$.
Par la proposition 9.1.8, $\pi_0\delta(type(u) \cup t^\cap) = \pi_0\delta(type(u)) \cap \pi_0\delta(t^\cap)$. Donc $\pi_0\delta(type(u) \cup t^\cap) \subseteq \pi_0\delta(type(u))$, et $\beta'(u) \in \pi_0\delta(type(u))$. Ainsi, donc β satisfait l'équation 9.31 pour G.
Ainsi, $(D,\delta,\beta) \vDash G$, et donc $G' \vDash G$. En réutilisant ce résultat pour toute une séquence de dérivations immédiates, nous prouvons le résultat final. □

Démonstration du Lemme 9.3.11 . Soit $G = \langle U^g, \boldsymbol{l}^g, A^g, C^g, Eq^g \rangle$ et $H = \langle U^h, \boldsymbol{l}^h, A^h, C^h, Eq^h \rangle$.
G est clos, donc Eq^g définit une relation d'équivalence sur U^g.
Soit $(D, \delta, [\cdot])$ le modèle canonique de G. $D = U^g/Eq^g \cup \{\bullet\}$.
Comme $G \vDash H$, alors il existe une affectation β telle que $(D, \delta, \beta) \vDash H$.
β est une application de U^h vers $D \setminus \bullet = U^g/Eq^g$.

Soit $tr : U^g/Eq^g \to U^g$ une fonction de choix, qui associe à chaque classe d'équivalence de nœud d'unité de G un nœud d'unité de cette classe d'équivalence.

Montrons que $tr \circ \beta$ définit un homomorphisme de H vers G.
Eq. 8.3 : Soit $u \in U^h$ et $m \in marker^h(u)$.
Par l'équation 9.30, $\beta(u) = \delta(m)$.
On sait que $[tr \circ \beta(u)] = \beta(u)$.
Donc $[tr \circ \beta(u)] = \delta(m) \neq \bullet$.
Finalement par la définition 109, $m \in marker^g(tr \circ \beta(u))$.
Eq. 8.4 : Soit $u \in U^h$.
Par l'équation 9.31, $\beta(u) \in \pi_0 \delta(type^h(u))$.
On sait que $[tr \circ \beta(u)] = \beta(u)$. Donc $[tr \circ \beta(u)] \in \pi_0 \delta(type^h(u))$
Par la proposition 9.1.8, $[tr \circ \beta(u)] \in \bigcap_{t \in type^h(u)} \pi_0 \delta(\{t\})$.
Par la définition 109, $type^h(u) \subseteq type^g(tr \circ \beta(u))$.
Finalement par la proposition 7.2.2, $type^g(tr \circ \beta(u)) \overset{\cap}{\lesssim} type^h(u)$.
Eq. 8.5 : Soit $(u, s, v) \in A^h$.
Par l'équation 9.32, $(\beta(u), \beta(v)) \in \pi_{0,s} \delta(\{\boldsymbol{\gamma}(s)\})$.
On sait que $[tr \circ \beta(u)] = \beta(u)$.
Donc il existe $r \in \delta(\{\boldsymbol{\gamma}(s)\})$ tel que $r(0) = [tr \circ \beta(u)]$, et $r(s) = [tr \circ \beta(v)] \neq \bullet$.
Finalement par la définition 109, $(tr \circ \beta(u), s, tr \circ \beta(v)) \in A^g$.
Eq. 8.6 : Soit $(u, s, v) \in C^h$;
Par l'équation 9.33, $(\beta(u), \beta(v)) \in \delta(s)$.
On sait que $[tr \circ \beta(u)] = \beta(u)$.
Donc $([tr \circ \beta(u)], [tr \circ \beta(v)]) \in \delta(s)$.
Finalement par la définition 109, $(tr \circ \beta(u), s, tr \circ \beta(v)) \in C^g$.
Eq. 8.7 : Soit $(u, v) \in Eq^h$.
Par l'équation 9.34, $\beta(u) = \beta(v)$. On sait que $[tr \circ \beta(u)] = \beta(u)$.
Donc $[tr \circ \beta(u)] = [tr \circ \beta(v)]$, et $(tr \circ \beta(u), tr \circ \beta(v)) \in Eq^g$.
Conclusion de la preuve π est un homomorphisme de H vers G. □

Démonstration of Théorème 9.3.12 . Soient $G, H \in \mathscr{G}(\mathscr{S})$. Si $G \vDash H$, et que G possède une expansion finie G',
Par le lemme 9.3.10, $G' \vDash G$, donc $G' \vDash H$.
Par le lemme 9.3.11, $G' \vDash H$ et G' est clos, donc il existe un homomorphisme de H vers G'.
Ainsi, il existe un homomorphisme de H vers une dérivation G' de G, donc H est déductible de G. □

ANNEXE B

Quelques transformées entre les logiques de description et la logique du premier ordre

---

Cette annexe précise la transformée entre les logiques de description et la logique du premier ordre pour quelques axiomes et constructeurs de classes et de propriétés. Nous nous limitons aux transformées nécessaires pour comprendre le chapitre 6 de ce mémoire.

ObjectIntersectionOf(A B)
Syntaxe des logiques de description : $A \sqcap B$

$$(A \sqcap B)(x) \iff A(x) \wedge B(x)$$

ObjectUnionOf(A B)
Syntaxe des logiques de description : $A \sqcup B$

$$(A \sqcup B)(x) \iff A(x) \vee B(x)$$

ObjectComplementOf(A)
Syntaxe des logiques de description : $\neg A$

$$(\neg A)(x) \iff \neg A(x)$$

ObjectAllValuesFrom(f C)
Syntaxe des logiques de description : $\forall f.C$

$$(\forall f.C)(x) \iff (\forall y)[f(x,y) \rightarrow C(y)]$$

ObjectSomeValuesFrom(f C)
Syntaxe des logiques de description : $\exists f.C$

$$(\exists f.C)(x) \iff (\exists y)[f(x,y) \wedge C(y)]$$

ObjectHasSelf(f)
Syntaxe des logiques de description : $\mathbf{HasSelf}.f$

$$(\mathbf{HasSelf}.f)(x) \iff f(x,x)$$

ObjectMaxCardinality(1 f)
Syntaxe des logiques de description : $\leq 1.f$

$$(\leq 1.f)(x) \iff (\forall y,z)[f(x,y) \wedge f(x,z) \rightarrow y = z]$$

ObjectMinCardinality(1 f)
Syntaxe des logiques de description : $\geq 1.f$

$$(\geq 1.f)(x) \iff (\exists y)[f(x,y)]$$

ObjectMaxCardinality(0 f)
Syntaxe des logiques de description : $\leq 0.f$

$$(\leq 0.f)(x) \iff (\forall y)[\neg f(x,y)]$$

subClassOf(A B)
Syntaxe des logiques de description : $A \sqsubseteq B$

$$A \sqsubseteq B \iff (\forall x)[A(x) \rightarrow B(x)]$$

equivalentClasses(A B)
Syntaxe des logiques de description : $A \equiv B$

$$A \equiv B \iff (\forall x)[A(x) \leftrightarrow B(x)]$$

subObjectPropertyOf(f g)
Syntaxe des logiques de description : $f \sqsubseteq g$

$$f \sqsubseteq g \iff (\forall x,y)[f(x,y) \rightarrow g(x,y)]$$

objectPropertyDomain(f C)
Syntaxe des logiques de description : $(\geq 1.f) \sqsubseteq C$

$$(\geq 1.f) \sqsubseteq C \iff (\forall x)\big[(\exists y)[f(x,y)] \rightarrow C(x)\big]$$

PropertyChainAxiom(f_1 f_2 g)
Syntaxe des logiques de description : $f_1 \circ f_2 \sqsubseteq g$

$$f_1 \circ f_2 \sqsubseteq g \iff (\forall x,y,z)[f_1(x,y) \wedge f_2(y,z) \rightarrow g(x,z)]$$

ANNEXE C

Base de règles SPARQL OWL 2 RL/RDF

C.1 The Semantics of Equality

```
# eq-ref
CONSTRUCT {
    ?s owl:sameAs ?s .
    ?p owl:sameAs ?p .
    ?o owl:sameAs ?o .
}
WHERE {
  ?s ?p ?o .
}
```

```
# eq-sym
CONSTRUCT {
    ?y owl:sameAs ?x .
}
WHERE {
    ?x owl:sameAs ?y .
}
```

```
# eq-trans
CONSTRUCT {
    ?x owl:sameAs ?z .
}
WHERE {
    ?x owl:sameAs ?y .
    ?y owl:sameAs ?z .
}
```

```
# eq-rep-s
CONSTRUCT {
    ?s2 ?p ?o .
}
WHERE {
    ?s owl:sameAs ?s2 .
    ?s ?p ?o .
}
```

```
# eq-rep-p
CONSTRUCT {
    ?s ?p2 ?o .
}
WHERE {
    ?p owl:sameAs ?p2 .
    ?s ?p ?o .
}
```

```
# eq-rep-o
CONSTRUCT {
    ?s ?p ?o2 .
}
WHERE {
    ?o owl:sameAs ?o2 .
    ?s ?p ?o .
}
```

```
# eq-diff1
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath owl:sameAs .
    _:b0 rdfs:label "Violates owl:differentFrom" .
}
WHERE {
    ?x owl:sameAs ?y .
    ?x owl:differentFrom ?y .
}
```

```
# eq-diff2
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?y1 .
    _:b0 rdfs:label "Violation of owl:AllDifferent" .
}
WHERE {
    ?x a owl:AllDifferent .
    ?x owl:members ?list1 .
    ?list1 rdf:rest* ?list2 .
    ?list2 rdf:first ?zi .
    ?list2 rdf:rest+ ?list3 .
    ?list3 rdf:first ?zj .
    ?zi owl:sameAs ?zj .
}
```

```
# eq-diff3
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?y1 .
    _:b0 rdfs:label "Violation of owl:AllDifferent" .
}
WHERE {
    ?x a owl:AllDifferent .
    ?x owl:distinctMembers ?list1 .
    ?list1 rdf:rest* ?list2 .
    ?list2 rdf:first ?zi .
    ?list2 rdf:rest+ ?list3 .
    ?list3 rdf:first ?zj .
    ?zi owl:sameAs ?zj .
}
```

C.2 The Semantics of Axioms about Properties

```
# prp-ap
#!!! What to do ? T(ap, rdf:type, owl:AnnotationProperty) for each built-in annotation
     property of OWL 2 RL
```

```
# prp-dom
CONSTRUCT {
    ?x a ?c .
}
WHERE {
    ?p rdfs:domain ?c .
    ?x ?p ?y .
}
```

```
# prp-rng
CONSTRUCT {
    ?y a ?c .
}
WHERE {
    ?p rdfs:range ?c .
    ?x ?p ?y .
}
```

```
# prp-fp
CONSTRUCT {
    ?y1 owl:sameAs ?y2 .
}
WHERE {
    ?p a owl:FunctionalProperty .
    ?x ?p ?y1 .
    ?x ?p ?y2 .
#? FILTER (?y1 != ?y2) . ?# does this filter optimize the query time ?
}
```

```
# prp-ifp
CONSTRUCT {
    ?x1 owl:sameAs ?x2 .
}
WHERE {
    ?p a owl:InverseFunctionalProperty .
    ?x1 ?p ?y .
    ?x2 ?p ?y .
#? FILTER (?x1 != ?x2) . ?# does this filter optimize the query time ?
}
```

```
# prp-irp
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Irreflexive property" .
}
WHERE {
    ?p a owl:IrreflexiveProperty .
    ?x ?p ?x .
}
```

```
# prp-symp
CONSTRUCT {
    ?y ?p ?x .
}
WHERE {
    ?p a owl:SymmetricProperty .
```

```
    ?x ?p ?y .
}
```

```
# prp-asyp
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Antisymmetric property" .
}
WHERE {
    ?p a owl:AsymmetricProperty .
    ?x ?p ?y .
    ?y ?p ?x .
}
```

```
# prp-trp
CONSTRUCT {
    ?x ?p ?z .
}
WHERE {
    ?p a owl:TransitiveProperty .
    ?x ?p ?y .
    ?y ?p ?z .
}
```

```
# prp-spo1
CONSTRUCT {
    ?x ?p2 ?y .
}
WHERE {
    ?p1 rdfs:subPropertyOf ?p2 .
    ?x ?p1 ?y .
}
```

```
# prp-spo2
prefix c: <http://www.inria.fr/acacia/comma#>
CONSTRUCT {
  ?u ?p ?v .
}
WHERE {
  ?p owl:propertyChainAxiom ?x .
  {select
    (group_concat(concat('<', ?pi, '>'); separator = '/') as ?exp)
    (concat('select * where { ?u ', ?exp, ' ?v }') as ?query)
    where {
      ?x rdf:rest*/rdf:first ?pi
    }
  }
  {select (unnest(kg:sparql(?query)) as (?u, ?v)) where {}}
}
```

```
# prp-eqp1
CONSTRUCT {
    ?x ?p2 ?y .
}
WHERE {
    ?p1 owl:equivalentProperty ?p2 .
```

```
    ?x ?p1 ?y .
}
```

```
# prp-eqp2
CONSTRUCT {
    ?x ?p1 ?y .
}
WHERE {
    ?p1 owl:equivalentProperty ?p2 .
    ?x ?p2 ?y .
}
```

```
# prp-pdw
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath ?p1 .
    _:b0 rdfs:label "Property declared disjoint with" .
}
WHERE {
    ?p1 owl:propertyDisjointWith ?p2 .
    ?x ?p1 ?y .
    ?x ?p2 ?y .
}
```

```
# prp-adp
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath ?p1 .
    _:b0 rdfs:label "Violation of owl:AllDisjointProperties" .
}
WHERE {
    ?x a owl:AllDisjointProperties .
    ?x owl:members ?list1 .
    ?list1 rdf:rest* ?list2 .
    ?list2 rdf:first ?pi .
    ?list2 rdf:rest+ ?list3 .
    ?list3 rdf:first ?pj .
    ?u ?pi ?v .
    ?u ?pj ?v .
}
```

```
# prp-inv1
CONSTRUCT {
    ?y ?p2 ?x .
}
WHERE {
    ?p1 owl:inverseOf ?p2 .
    ?x ?p1 ?y .
}
```

```
# prp-inv2
CONSTRUCT {
    ?y ?p1 ?x .
}
WHERE {
    ?p1 owl:inverseOf ?p2 .
```

```
    ?x ?p2 ?y .
}
```

```
#prp-key
#!!! SPARQL Query for Corese... CHECK BEFORE USE !!!
prefix c: <http://www.inria.fr/acacia/comma#>
CONSTRUCT {
  ?x owl:sameAs ?y .
}
WHERE {
  ?c owl:hasKey ?u .
  {SELECT
    (group_concat(concat('?x <', ?pi, '> ?', pi); separator = ' \n') as ?expx)
    (group_concat(concat('?y <', ?pi, '> ?', pi); separator = ' \n') as ?expy)
    (concat('select ?x ?y where {', ?expx, '\n', ?expy, '}') as ?query)
    WHERE {
      ?u rdf:rest*/rdf:first ?pi
    }
  }
  {SELECT (unnest(kg:sparql(?query)) as (?x, ?y)) WHERE {}}
}
```

```
# prp-npa1
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?i1 .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Negative Property Assertion" .
}
WHERE {
    ?x owl:sourceIndividual ?i1 .
    ?x owl:assertionProperty ?p .
    ?x owl:targetIndividual ?i2 .
    ?i1 ?p ?i2 .
}
```

```
# prp-npa2
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?i .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Negative Property Assertion" .
}
WHERE {
    ?x owl:sourceIndividual ?i .
    ?x owl:assertionProperty ?p .
    ?x owl:targetValue ?lt .
    ?i ?p ?lt .
}
```

C.3 The Semantics of Classes

```
# cls-thing
CONSTRUCT {
  owl:Thing a owl:Class .
}
WHERE {}
```

```
# cls-nothing1
CONSTRUCT {
  owl:Nothing a owl:Class .
}
WHERE {}
```

```
# cls-nothing2
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 rdfs:label "There exists an instance of owl:Nothing" .
}
WHERE {
    ?x a owl:Nothing .
}
```

```
# cls-int1
#!!! CORESE SPARQL QUERY - CHECK BEFORE USE
prefix c: <http://www.inria.fr/acacia/comma#>
CONSTRUCT {
  ?y a ?c .
}
WHERE {
    ?c owl:intersectionOf ?x .
  {SELECT
    (group_concat(concat('?y a <', ?ci, '>'); separator = ' \n') as ?exp)
    (concat('select ?y where {', ?exp, '}') as ?query)
    WHERE {
      ?x rdf:rest*/rdf:first ?ci
    }
  }
  {SELECT (unnest(kg:sparql(?query)) as (?y)) WHERE {}}
}
```

```
# cls-int2
CONSTRUCT {
    ?y a ?ci .
}
WHERE {
    ?c owl:intersectionOf ?x .
    ?x rdf:rest*/rdf:first ?ci .
    ?y a ?c .
}
```

```
# cls-uni
CONSTRUCT {
    ?y a ?c .
}
WHERE {
    ?c owl:unionOf ?x .
    ?x rdf:rest*/rdf:first ?ci .
    ?y a ?ci .
}
```

```
# cls-com
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
```

```
    _:b0 rdfs:label "ComplementOf Violation" .
}
WHERE {
    ?c1 owl:complementOf ?c2 .
    ?x a ?c1 .
    ?x a ?c2 .
}
```

```
# cls-com
CONSTRUCT {
    ?u a ?x .
}
WHERE {
    ?x owl:someValuesFrom ?y .
    ?x owl:onProperty ?p .
    ?u ?p ?v .
    ?v a ?y .
}
```

```
# cls-svf2
CONSTRUCT {
    ?u a ?x .
}
WHERE {
    ?x owl:someValuesFrom owl:Thing .
    ?x owl:onProperty ?p .
    ?u ?p ?v .
}
```

```
# cls-avf
CONSTRUCT {
    ?v a ?y .
}
WHERE {
    ?x owl:allValuesFrom ?y .
    ?x owl:onProperty ?p .
    ?u a ?x .
    ?u ?p ?v .
}
```

```
# cls-hv1
CONSTRUCT {
    ?u ?p ?y .
}
WHERE {
    ?x owl:hasValue ?y .
    ?x owl:onProperty ?p .
    ?u a ?x .
}
```

```
# cls-hv2
CONSTRUCT {
    ?u a ?x .
}
WHERE {
    ?x owl:hasValue ?y .
    ?x owl:onProperty ?p .
    ?u ?p ?y .
}
```

```
# cls-maxc1
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "owl:maxCardinality of 0" .
}
WHERE {
    ?x owl:maxCardinality "0"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?u a ?x .
    ?u ?p ?y .
}
```

```
# cls-maxc2
CONSTRUCT {
    ?y1 owl:sameAs ?y2 .
}
WHERE {
    ?x owl:maxCardinality "1"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?u a ?x .
    ?u ?p ?y1 .
    ?u ?p ?y2 .
}
```

```
# cls-maxqc1
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?u .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Maximum qualified cardinality of 0" .
}
WHERE {
    ?x owl:maxQualifiedCardinality "0"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?x owl:onClass ?c .
    ?u a ?x .
    ?u ?p ?y .
    ?y a ?c .
}
```

```
# cls-maxqc2
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?u .
    _:b0 spin:violationPath ?p .
    _:b0 rdfs:label "Qualified max cardinality 0" .
}
WHERE {
    ?x owl:maxQualifiedCardinality "0"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?x owl:onClass owl:Thing .
    ?u a ?x .
    ?u ?p ?y .
}
```

```
# cls-maxqc3
```

```
CONSTRUCT {
    ?y1 owl:sameAs ?y2 .
}
WHERE {
    ?x owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?x owl:onClass ?c .
    ?u a ?x .
    ?u ?p ?y1 .
    ?y1 a ?c .
    ?u ?p ?y2 .
    ?y2 a ?c .
}
```

```
# cls-maxqc4
CONSTRUCT {
    ?y1 owl:sameAs ?y2 .
}
WHERE {
    ?x owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger .
    ?x owl:onProperty ?p .
    ?x owl:onClass owl:Thing .
    ?u a ?x .
    ?u ?p ?y1 .
    ?u ?p ?y2 .
}
```

```
# cls-oo
CONSTRUCT {
    ?yi a ?c .
}
WHERE {
    ?c owl:oneOf ?x .
    ?x rdf:rest*/rdf:first ?yi .
}
```

C.4 The Semantics of Class Axioms

```
# cax-sco
CONSTRUCT {
    ?x a ?c2 .
}
WHERE {
    ?c1 rdfs:subClassOf ?c2 .
    ?x a ?c1 .
}
```

```
# cax-eqc1
#!!! useless with cax-sco and scm-eqc2
CONSTRUCT {
    ?x a ?c2 .
}
WHERE {
    ?c1 owl:equivalentClass ?c2 .
    ?x a ?c1 .
}
```

```
# cax-eqc2
#!!! useless with cax-sco and scm-eqc2
CONSTRUCT {
    ?x a ?c1 .
}
WHERE {
    ?c1 owl:equivalentClass ?c2 .
    ?x a ?c2 .
}
```

```
# cax-dw
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 rdfs:label "Shared instance of disjoint classes" .
}
WHERE {
    ?c1 owl:disjointWith ?c2 .
    ?x a ?c1 .
    ?x a ?c2 .
}
```

```
# cax-adc
CONSTRUCT {
    _:b0 a spin:ConstraintViolation .
    _:b0 spin:violationRoot ?x .
    _:b0 rdfs:label "Shared instance of classes from an AllDisjointClasses block" .
}
WHERE {
    ?y a owl:AllDisjointClasses .
  ?y owl:members ?list1 .
    ?list1 rdf:rest* ?list2 .
    ?list2 rdf:first ?ci .
    ?list2 rdf:rest+ ?list3 .
    ?list3 rdf:first ?cj .
    ?x a ?ci .
    ?x a ?cj .
}
```

C.5 The Semantics of Schema Vocabulary

```
# scm-cls
CONSTRUCT {
    ?c rdfs:subClassOf ?c .
    ?c owl:equivalentClass ?c .
    ?c rdfs:subClassOf owl:Thing .
    owl:Nothing rdfs:subClassOf ?c .
}
WHERE {
    ?c a owl:Class .
}
```

```
# scm-sco
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c3 .
}
WHERE {
```

```
    ?c1 rdfs:subClassOf ?c2 .
    ?c2 rdfs:subClassOf ?c3 .
}
```

```
# scm-eqc1
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c2 .
    ?c2 rdfs:subClassOf ?c1 .
}
WHERE {
    ?c1 owl:equivalentClass ?c2 .
}
```

```
# scm-eqc2
CONSTRUCT {
    ?c1 owl:equivalentClass ?c2 .
}
WHERE {
    ?c1 rdfs:subClassOf ?c2 .
    ?c2 rdfs:subClassOf ?c1 .
}
```

```
# scm-op
CONSTRUCT {
    ?p rdfs:subPropertyOf ?p .
    ?p owl:equivalentProperty ?p .
}
WHERE {
    ?p a owl:ObjectProperty .
}
```

```
# scm-dp
CONSTRUCT {
    ?p owl:equivalentProperty ?p .
}
WHERE {
    ?p a owl:DatatypeProperty .
}
```

```
# scm-eqp1
CONSTRUCT {
    ?p1 rdfs:subPropertyOf ?p2 .
    ?p2 rdfs:subPropertyOf ?p1 .
}
WHERE {
    ?p1 owl:equivalentProperty ?p2 .
}
```

```
# scm-eqp2
CONSTRUCT {
    ?p1 owl:equivalentProperty ?p2 .
}
WHERE {
    ?p1 rdfs:subPropertyOf ?p2 .
    ?p2 rdfs:subPropertyOf ?p1 .
}
```

```
# scm-dom1
CONSTRUCT {
    ?p rdfs:domain ?c2 .
}
WHERE {
    ?p rdfs:domain ?c1 .
    ?c1 rdfs:subClassOf ?c2 .
}
```

```
# scm-dom2
CONSTRUCT {
    ?p1 rdfs:domain ?c .
}
WHERE {
    ?p2 rdfs:domain ?c .
  ?p1 rdfs:subPropertyOf ?p2 .
}
```

```
# scm-rng1
CONSTRUCT {
    ?p rdfs:range ?c2 .
}
WHERE {
    ?p rdfs:range ?c1 .
    ?c1 rdfs:subClassOf ?c2 .
}
```

```
# scm-rng2
CONSTRUCT {
    ?p1 rdfs:range ?c .
}
WHERE {
    ?p2 rdfs:range ?c .
    ?p1 rdfs:subPropertyOf ?p2 .
}
```

```
# scm-hv
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c2 .
}
WHERE {
    ?c1 owl:hasValue ?i .
    ?c1 owl:onProperty ?p1 .
    ?c2 owl:hasValue ?i .
    ?c2 owl:onProperty ?p2 .
    ?p1 rdfs:subPropertyOf ?p2 .
}
```

```
# scm-svf1
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c2 .
}
WHERE {
    ?c1 owl:someValuesFrom ?y1 .
    ?c1 owl:onProperty ?p .
    ?c2 owl:someValuesFrom ?y2 .
    ?c2 owl:onProperty ?p .
    ?y1 rdfs:subClassOf ?y2 .
}
```

```
# scm-svf2
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c2 .
}
WHERE {
    ?c1 owl:someValuesFrom ?y .
    ?c1 owl:onProperty ?p1 .
    ?c2 owl:someValuesFrom ?y .
    ?c2 owl:onProperty ?p2 .
    ?p1 rdfs:subPropertyOf ?p2 .
}
```

```
# scm-avf1
CONSTRUCT {
    ?c1 rdfs:subClassOf ?c2 .
}
WHERE {
    ?c1 owl:allValuesFrom ?y1 .
    ?c1 owl:onProperty ?p .
    ?c2 owl:allValuesFrom ?y2 .
    ?c2 owl:onProperty ?p .
    ?y1 rdfs:subClassOf ?y2 .
}
```

```
# scm-avf2
CONSTRUCT {
    ?c2 rdfs:subClassOf ?c1 .
}
WHERE {
    ?c1 owl:allValuesFrom ?y .
    ?c1 owl:onProperty ?p1 .
    ?c2 owl:allValuesFrom ?y .
    ?c2 owl:onProperty ?p2 .
    ?p1 rdfs:subPropertyOf ?p2 .
}
```

```
# scm-int
CONSTRUCT {
    ?c rdfs:subClassOf ?ci .
}
WHERE {
    ?c owl:intersectionOf ?x .
    ?x rdf:rest*/rdf:first ?ci .
}
```

```
# scm-uni
CONSTRUCT {
    ?ci rdfs:subClassOf ?c .
}
WHERE {
    ?c owl:unionOf ?x .
    ?x rdf:rest*/rdf:first ?ci .
}
```

ANNEXE D

Schéma du Formalisme des Graphes d'Unités

Les ontologies décrites dans cette annexe ont été publiées suivant les bonnes pratiques du Web des données : http://lov.okfn.org/dataset/lov/Recommendations_Vocabulary_Design.pdf.

D.1 Modélisation du formalisme des Graphes d'Unités

Les sections suivantes définissent le modèle du formalisme des Graphes d'Unités :

- l'ontologie <http://ns.inria.org/ug/v1#> (§D.1.1) ;
- la base de règles de transfert entre les deux modèles, accessible à l'URL <http://ns.inria.org/ug/v1/ug-transfer.rul> (§D.1.2) ;
- la base de règles complémentaires génériques pour le modèle **gu-langue**, accessible à l'URL <http://ns.inria.org/ug/v1/ug-language.rul> (§D.1.3) ;
- la base de règles complémentaires génériques pour le modèle **gu-usage**, accessible à l'URL <http://ns.inria.org/ug/v1/ug-usage.rul> (§D.1.4) ;
- la base de règles pour la génération d'une base de règles d'expansion et de contraction, accessible à l'URL <http://ns.inria.org/ug/v1/ug-definition-rules-generation-rules.rul> (§D.1.5).

D.1.1 Ontologie

L'ontologie suivante décrit le modèle du formalisme des GU sur le Web des données. Elle est accessible à l'URL <http://ns.inria.org/ug/v1#>.

```
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://ns.inria.org/ug/v1#> a owl:Ontology ;
    dc:title "The Unit Graphs ontology (UG)"@en .

# Required to read in Protege
dc:title a owl:AnnotationProperty .

ug:UnitType a owl:Class ;
    rdfs:label "UnitType" ;
    rdfs:comment "The class of unit types."@en ;
    rdfs:subClassOf [ a owl:Restriction ;
```

```
19      owl:onProperty ug:specializationOf ;
20      owl:hasValue ug:Unit ] ;
21     rdfs:subClassOf [ a owl:Restriction ;
22      owl:onProperty [ owl:inverseOf ug:specializationOf ] ;
23      owl:hasValue ug:AbsurdUnit ] .
24
25 ug:Unit a ug:UnitType ;
26     a owl:Class ;
27     rdfs:label "Unit" ;
28     rdfs:comment "The prime universal unit type."@en ;
29     rdfs:comment "The class of all units."@en .
30
31 ug:AbsurdUnit a ug:UnitType ;
32     a owl:Class ;
33     rdfs:label "AbsurdUnit" ;
34     rdfs:comment "The prime absurd unit type."@en ;
35     rdfs:comment "The class of no units."@en .
36
37 ug:specializationOf a owl:ObjectProperty ;
38     a owl:ReflexiveProperty ;
39     a owl:TransitiveProperty ;
40     rdfs:label "specializationOf" ;
41     rdfs:comment "The subject, a unit type, is a specialization of the object, a unit
           type."@en ;
42     rdfs:domain ug:UnitType ;
43     rdfs:range ug:UnitType ;
44     owl:propertyChainAxiom ( ug:obligatoryActantSlotRootOf ug:actantSlotRoot ) ;
45     owl:propertyChainAxiom ( ug:prohibitedActantSlotRootOf ug:actantSlotRoot ) .
46
47 ug:conjunctionOf a owl:ObjectProperty ;
48     rdfs:label "conjunctionOf" ;
49     rdfs:comment "The subject, a unit type, is the conjunction of the object, a list of unit
           types."@en ;
50     rdfs:domain ug:UnitType ;
51     rdfs:range ug:UnitType .
52
53 ug:ActantialRelationSymbol a owl:Class ;
54     rdfs:label "ActantialRelationSymbol" ;
55     rdfs:comment "The class of actantial relation symbols."@en .
56
57 ug:actantialRelation a owl:ObjectProperty ;
58     a owl:FunctionalProperty ;
59     rdfs:label "actantialRelation" ;
60     rdfs:comment "The generic actantial relation."@en ;
61     rdfs:domain ug:Unit ;
62     rdfs:range ug:Unit .
63
64 ug:actantSlotRoot a owl:ObjectProperty ;
65     a owl:FunctionalProperty ;
66     rdfs:label "actantSlotRoot" ;
67     rdfs:comment "The subject, an actantial symbol relation, has for actant slot root the
           object, a unit type."@en ;
68     rdfs:domain ug:ActantialRelationSymbol ;
69     rdfs:range ug:UnitType ;
70     owl:inverseOf ug:actantSlotRootOf .
71
72 ug:actantSlotRootOf a rdf:Property ;
73     a owl:InverseFunctionalProperty ;
```

```
74      rdfs:label "actantSlotRootOf" ;
75      rdfs:comment "The subject, a unit type, is the actant slot root of the object, an
            actantial symbol relation."@en ;
76      rdfs:domain ug:UnitType ;
77      rdfs:range ug:ActantialRelationSymbol ;
78      owl:inverseOf ug:actantSlotRoot .
79
80  ug:obligatoryActantSlotRoot a owl:ObjectProperty ;
81      a owl:FunctionalProperty ;
82      rdfs:label "obligatoryActantSlotRoot" ;
83      rdfs:comment "The subject, an actantial symbol relation, has for obligatory actant slot
            root the object, a unit type."@en ;
84      rdfs:domain ug:ActantialRelationSymbol ;
85      rdfs:range ug:UnitType ;
86      owl:inverseOf ug:obligatoryActantSlotRootOf .
87
88  ug:obligatoryActantSlotRootOf a rdf:Property ;
89      a owl:InverseFunctionalProperty ;
90      rdfs:label "obligatoryActantSlotRootOf" ;
91      rdfs:comment "The subject, a unit type, is the obligatory actant slot root of the object,
            an actantial symbol relation."@en ;
92      rdfs:domain ug:UnitType ;
93      rdfs:range ug:ActantialRelationSymbol ;
94      owl:inverseOf ug:obligatoryActantSlotRoot .
95
96  ug:prohibitedActantSlotRoot a owl:ObjectProperty ;
97      a owl:FunctionalProperty ;
98      rdfs:label "prohibitedActantSlotRoot" ;
99      rdfs:comment "The subject, an actantial symbol relation, has for prohibited actant slot
            root the object, a unit type."@en ;
100     rdfs:domain ug:ActantialRelationSymbol ;
101     rdfs:range ug:UnitType ;
102     owl:inverseOf ug:prohibitedActantSlotRootOf .
103
104 ug:prohibitedActantSlotRootOf a rdf:Property ;
105     a owl:InverseFunctionalProperty ;
106     rdfs:label "prohibitedActantSlotRootOf" ;
107     rdfs:comment "The subject, a unit type, is the prohibited actant slot root of the object,
            an actantial symbol relation."@en ;
108     rdfs:domain ug:UnitType ;
109     rdfs:range ug:ActantialRelationSymbol ;
110     owl:inverseOf ug:prohibitedActantSlotRoot .
111
112 ug:actantSlot a owl:ObjectProperty ;
113     rdfs:label "actantSlot" ;
114     rdfs:comment "The subject, a unit type, has for actant slot the object, an actantial
            symbol relation." ;
115     rdfs:domain ug:UnitType ;
116     rdfs:range ug:ActantialRelationSymbol ;
117     owl:propertyChainAxiom ( ug:specializationOf ug:actantSlotRootOf ) .
118
119 ug:obligatoryActantSlot a owl:ObjectProperty ;
120     rdfs:subPropertyOf ug:actantSlot ;
121     rdfs:label "obligatoryActantSlot" ;
122     rdfs:comment "The subject, a unit type, has for obligatory actant slot the object, an
            actantial symbol relation." ;
123     rdfs:domain ug:UnitType ;
124     rdfs:range ug:ActantialRelationSymbol ;
```

```
125     owl:propertyChainAxiom ( ug:specializationOf ug:obligatoryActantSlotRootOf ) .
126
127 ug:prohibitedActantSlot a owl:ObjectProperty ;
128     rdfs:subPropertyOf ug:actantSlot ;
129     rdfs:label "prohibitedActantSlot" ;
130     rdfs:comment "The subject, a unit type, has for prohibited actant slot the object, an
            actantial symbol relation." ;
131     rdfs:domain ug:UnitType ;
132     rdfs:range ug:ActantialRelationSymbol ;
133     owl:propertyChainAxiom ( ug:specializationOf ug:prohibitedActantSlotRootOf ) .
134
135 ug:Signature a owl:Class ;
136     rdfs:subClassOf ug:UnitType ;
137     rdfs:label "Signature" ;
138     rdfs:comment "The class of signatures."@en .
139
140 ug:onActantSlot a owl:ObjectProperty ;
141     rdfs:label "onActantSlot" ;
142     rdfs:comment "The subject, a signature, affect the object, an actant slot." ;
143     rdfs:domain ug:Signature ;
144     rdfs:range ug:ActantialRelationSymbol .
145
146 ug:allUnitsFrom a owl:ObjectProperty ;
147     rdfs:label "allUnitsFrom" ;
148     rdfs:comment "The subject, a signature, is restricted to the object, a unit type." ;
149     rdfs:domain ug:Signature ;
150     rdfs:range ug:UnitType .
151
152 ug:circumstantialRelation a owl:ObjectProperty ;
153     rdfs:label "circumstantialRelation" ;
154     rdfs:comment "The generic circumstantial relation."@en ;
155     rdfs:domain ug:Unit ;
156     rdfs:range ug:Unit .
157
158 ug:Participant a owl:Class ;
159     rdfs:subClassOf ug:Unit ;
160     rdfs:label "Participant" ;
161     rdfs:comment "The class of units that represent participants of formal lexicographic
            definitions."@en .
162
163 ug:ObligatoryParticipant a owl:Class ;
164     rdfs:subClassOf ug:Participant ;
165     rdfs:label "ObligatoryParticipant" ;
166     rdfs:comment "The class of obligatory participants."@en ;
167     owl:disjointWith ug:ProhibitedParticipant .
168
169 ug:ProhibitedParticipant a owl:Class ;
170     rdfs:subClassOf ug:Participant ;
171     rdfs:label "ProhibitedParticipant" ;
172     rdfs:comment "The class of prohibited participants."@en ;
173     owl:disjointWith ug:ObligatoryParticipant .
174
175 ug:marker a owl:ObjectProperty ;
176     rdfs:label "marker" ;
177     rdfs:comment "The subject participant has for marker the object actantial relation
            symbol." ;
178     rdfs:domain ug:Participant ;
179     rdfs:range ug:ActantialRelationSymbol .
```

```
ug:definition a owl:ObjectProperty ;
    rdfs:label "definition" ;
    rdfs:comment "The subject unit type has a for central participant of its formal
        lexicographic definition the object participant." ;
    rdfs:domain ug:UnitType ;
    rdfs:range ug:ObligatoryParticipant .
```

D.1.2 Base de règles de transfert

La base de règles suivante complémente la base de règles OWL 2 RL/RDF. Elle permet d'assurer l'interopérabilité entre les modèles **gu-langue** et **gu-usage**, et contribue à simuler la sémantique du formalisme des GU. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/ug-transfer.rul>.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY rul "http://ns.inria.fr/edelweiss/2011/rule#">
]>

<rdf:RDF xmlns:rdfs="&rdfs;" xmlns:rdf="&rdf;" xmlns = '&rul;' >
  <rule><body><![CDATA[
    # language-usage transfer rule: unit types
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    CONSTRUCT {
     ?t a owl:Class ;
       rdfs:subClassOf ug:Unit .
    }
    WHERE {
     ?t a ug:UnitType .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # usage-language transfer rule: unit types
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    CONSTRUCT {
     ?t a ug:UnitType .
    }
    WHERE {
     ?t a owl:Class ;
       rdfs:subClassOf ug:Unit .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # language-usage transfer rule: specialization relation
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    CONSTRUCT {
```

```
    ?t1 rdfs:subClassOf ?t2 .
   }
   WHERE {
    ?t1 ug:specializationOf ?t2 .
   }
]]></body></rule>
<rule><body><![CDATA[
   # usage-language transfer rule: specialization relation
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?t1 ug:specializationOf ?t2 .
   }
   WHERE {
    ?t1 rdfs:subClassOf ?t2 .
    ?t2 rdfs:subClassOf ug:Unit .
   }
]]></body></rule>
<rule><body><![CDATA[
   # language-usage transfer rule: conjunctive unit types
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?tc owl:intersectionOf ?list .
   }
   WHERE {
    ?tc ug:conjunctionOf ?list .
   }
]]></body></rule>
<rule><body><![CDATA[
   # usage-language transfer rule: conjunctive unit types
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?tc ug:conjunctionOf ?list .
   }
   WHERE {
    ?tc rdfs:subClassOf ug:Unit .
    ?tc owl:intersectionOf ?list .
   }
]]></body></rule>
<rule><body><![CDATA[
   # language-usage transfer rule: actantial relation symbols
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?s a owl:ObjectProperty ;
      rdfs:subPropertyOf ug:actantialRelation .
   }
   WHERE {
```

```
    ?s a ug:ActantialRelationSymbol .
   }
 ]]></body></rule>
 <rule><body><![CDATA[
   # usage-language transfer rule: actantial relation symbols
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?s a ug:ActantialRelationSymbol .
   }
   WHERE {
    ?s a owl:ObjectProperty ;
      rdfs:subPropertyOf ug:actantialRelation .
   }
 ]]></body></rule>
 <rule><body><![CDATA[
   # language-usage transfer rule: PUT that introduce ASlot
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?s rdfs:domain ?t .
   }
   WHERE {
    ?s ug:actantSlotRoot ?t .
   }
 ]]></body></rule>
 <rule><body><![CDATA[
   # usage-language transfer rule: PUT that introduce ASlot
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?aslotr ug:specializationOf ?t .
   }
   WHERE {
    ?s rdfs:domain ?t .
    ?s ug:actantSlotRoot ?aslotr .
   }
 ]]></body></rule>
 <rule><body><![CDATA[
   # language-usage transfer rule: ProASlotRoots
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?t a owl:Restriction ;
      owl:onProperty ?s ;
      owl:maxCardinality "0"^^xsd:nonNegativeInteger .
   }
   WHERE {
    ?s ug:prohibitedActantSlotRoot ?t .
   }
```

```
]]></body></rule>


<rule><body><![CDATA[
  # usage-language transfer rule: ProASlotRoots
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
  CONSTRUCT {
   ?s ug:prohibitedActantSlotRoot ?t .
  }
  WHERE {
   ?s rdfs:subPropertyOf ug:actantialRelation .
   ?t a owl:Restriction ;
      owl:onProperty ?s ;
      owl:maxCardinality "0"^^xsd:nonNegativeInteger .
  }
]]></body></rule>


<rule><body><![CDATA[
  # language-usage transfer rule: prohibited actant slot
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
  CONSTRUCT {
   ?t rdfs:subClassOf ?sroot .
  }
  WHERE {
   ?t ug:prohibitedActantSlot ?s .
   ?s ug:prohibitedActantSlotRoot ?sroot .
  }
]]></body></rule>


<rule><body><![CDATA[
  # usage-language transfer rule: prohibited actant slot
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
  CONSTRUCT {
   ?t ug:prohibitedActantSlot ?s .
  }
  WHERE {
   ?s rdfs:subPropertyOf ug:actantialRelation .
   ?t rdfs:subClassOf ug:Unit ;
      rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ?s ;
       owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
  }
]]></body></rule>
<rule><body><![CDATA[
  # language-usage transfer rule: signatures
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
218     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
219     Prefix owl: <http://www.w3.org/2002/07/owl#>
220     CONSTRUCT {
221      ?sig a owl:Restriction ;
222         owl:onProperty ?s ;
223         owl:allValuesFrom ?t .
224     }
225     WHERE {
226      ?sig a ug:Signature ;
227        ug:onActantSlot ?s ;
228        ug:allUnitsFrom ?t .
229     }
230   ]]></body></rule>
231   <rule><body><![CDATA[
232     # usage-language transfer rule: signatures
233     Prefix ug: <http://ns.inria.org/ug/v1#>
234     Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
235     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
236     Prefix owl: <http://www.w3.org/2002/07/owl#>
237     CONSTRUCT {
238      ?sig a ug:Signature ;
239        ug:onActantSlot ?s ;
240        ug:allUnitsFrom ?t .
241     }
242     WHERE {
243      ?s rdfs:subClassOf ug:actantialRelation .
244      ?t rdfs:subClassOf ug:Unit .
245      ?sig rdfs:subClassOf ug:Unit ;
246         a owl:Restriction ;
247         owl:onProperty ?s ;
248         owl:allValuesFrom ?t .
249     }
250   ]]></body></rule>
251 </rdf:RDF>
```

D.1.3 Base de règles complémentaires d'inférence pour le modèle gu-langue

La base de règles suivante complémente la base de règles OWL 2 RL/RDF. Elle permet d'axiomatiser la sémantique du modèle **gu-langue**, et contribue à simuler la sémantique du formalisme des GU. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/ug-language.rul>.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE rdf:RDF [
3  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
5  <!ENTITY rul "http://ns.inria.fr/edelweiss/2011/rule#">
6  ]>
7
8  <rdf:RDF xmlns:rdfs="&rdfs;" xmlns:rdf="&rdf;" xmlns = '&rul;' >
9    <rule><body><![CDATA[
10     # two unit types that are the same specialize each other.
11     Prefix ug: <http://ns.inria.org/ug/v1#>
12     Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
13     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
14     Prefix owl: <http://www.w3.org/2002/07/owl#>
15     CONSTRUCT {
16      ?t1 ug:specializationOf ?t2 .
```

```
    ?t2 ug:specializationOf ?t1 .
   }
   WHERE {
    ?t1 owl:sameAs ?t2 .
    ?t1 a ug:UnitType .
    ?t2 a ug:UnitType .
   }
]]></body></rule>
<rule><body><![CDATA[
   # a conjunctive unit type is more specific than any unit type it is composed of
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?tc ug:specializationOf ?t .
   }
   WHERE {
    ?tc (ug:conjunctionOf/(rdf:rest)*)/rdf:first ?t .
   }
]]></body></rule>
<rule><body><![CDATA[
   # if a unit type is more specific than all the unit types a conjunctive unit type is
       composed of, then it is more specific than the conjunctive unit type
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?t ug:specializationOf ?tc .
   }
   WHERE {
    ?tc ug:conjunctionOf ?list .
    ?t a ug:UnitType .
    FILTER NOT EXISTS {
      ?list (rdf:rest*)/rdf:first ?tn .
      FILTER NOT EXISTS {
       ?t ug:specializationOf ?tn .
      }
    }
   }
]]></body></rule>
<rule><body><![CDATA[
   # the conjunction of the obligatory actant slot root and the prohibited actant slot root
       of a given actantial relation symbol is an absurd unit type
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   CONSTRUCT {
    ?t ug:specializationOf ug:AbsurdUnit .
   }
   WHERE {
    ?t ug:specializationOf/ug:obligatoryActantSlotRootOf ?s .
    ?t ug:specializationOf/ug:prohibitedActantSlotRootOf ?s .
   }
]]></body></rule>
<rule><body><![CDATA[
```

```
73    Prefix ug: <http://ns.inria.org/ug/v1#>
74    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
75    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
76    Prefix owl: <http://www.w3.org/2002/07/owl#>
77    CONSTRUCT {
78     ?sig ug:specializationOf ?aslotr .
79    }
80    WHERE {
81     ?sig ug:onActantSlot/ug:actantSlotRoot ?aslotr .
82    }
83  ]]></body></rule>
84  <rule><body><![CDATA[
85    Prefix ug: <http://ns.inria.org/ug/v1#>
86    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
87    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
88    Prefix owl: <http://www.w3.org/2002/07/owl#>
89    CONSTRUCT {
90     ?sig1 ug:specializationOf ?sig2 .
91    }
92    WHERE {
93     ?sig1 ug:onActantSlot ?s .
94     ?sig1 ug:allUnitsFrom ?t1 .
95     ?sig2 ug:onActantSlot ?s .
96     ?sig2 ug:allUnitsFrom ?t2 .
97     ?t1 ug:specializationOf ?t2 .
98    }
99  ]]></body></rule>
100 <rule><body><![CDATA[
101   Prefix ug: <http://ns.inria.org/ug/v1#>
102   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
103   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
104   Prefix owl: <http://www.w3.org/2002/07/owl#>
105   CONSTRUCT {
106    ?t ug:specializationOf ug:AbsurdUnit .
107   }
108   WHERE {
109    ?t ug:obligatoryActantSlot ?s .
110    ?t ug:specializationOf ?sig .
111    ?sig ug:onActantSlot ?s .
112    ?sig ug:allUnitsFrom/ug:specializationOf ug:AbsurdUnit .
113   }
114 ]]></body></rule>
115 <rule><body><![CDATA[
116   Prefix ug: <http://ns.inria.org/ug/v1#>
117   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
118   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
119   Prefix owl: <http://www.w3.org/2002/07/owl#>
120   CONSTRUCT {
121    GRAPH ?p0 { ?p ug:marker ?s . }
122   }
123   WHERE {
124    ?s rdfs:subPropertyOf ug:actantialRelation .
125    ?t ug:definition ?p0 .
126    GRAPH ?p0 { ?p0 ?s ?p . }
127   }
128 ]]></body></rule>
129 <rule><body><![CDATA[
130   Prefix ug: <http://ns.inria.org/ug/v1#>
```

```
131    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
132    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
133    Prefix owl: <http://www.w3.org/2002/07/owl#>
134    CONSTRUCT {
135     GRAPH ?p0 {
136       ?p a ug:ObligatoryParticipant .
137     }
138    }
139    WHERE {
140     ?t rdfs:subClassOf [
141        a owl:Restriction ;
142        owl:onProperty ?s ;
143        owl:minCardinality "1"^^xsd:nonNegativeInteger ] ;
144       ug:definition ?p0 .
145     GRAPH ?p0 { ?p ug:marker ?s . }
146    }
147  ]]></body></rule>
148  <rule><body><![CDATA[
149    Prefix ug: <http://ns.inria.org/ug/v1#>
150    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
151    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
152    Prefix owl: <http://www.w3.org/2002/07/owl#>
153    CONSTRUCT {
154     GRAPH ?p0 {
155       ?p a ug:ProhibitedParticipant .
156     }
157    }
158    WHERE {
159     ?t rdfs:subClassOf [
160        a owl:Restriction ;
161        owl:onProperty ?s ;
162        owl:maxCardinality "0"^^xsd:nonNegativeInteger ] ;
163      ug:definition ?p0 .
164     GRAPH ?p0 { ?p ug:marker ?s . }
165    }
166  ]]></body></rule>
167  <rule><body><![CDATA[
168    Prefix ug: <http://ns.inria.org/ug/v1#>
169    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
170    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
171    Prefix owl: <http://www.w3.org/2002/07/owl#>
172    CONSTRUCT {
173     GRAPH ?p0 {
174       ?p2 a ug:ObligatoryParticipant .
175     }
176    }
177    WHERE {
178     ?t0 ug:definition ?p0 .
179     ?t1 rdfs:subClassOf [
180        a owl:Restriction ;
181        owl:onProperty ?s ;
182        owl:minCardinality "1"^^xsd:nonNegativeInteger ] .
183     ?s rdfs:subPropertyOf ug:actantialRelation .
184     GRAPH ?p0 {
185       ?p1 a ?t1 .
186       ?p2 a ?t2 .
187       ?p1 ?s ?p2 .
188       ?p1 a ug:ObligatoryParticipant .
```

```
    }
   }
]]></body></rule>
<rule><body><![CDATA[
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
  CONSTRUCT {
   GRAPH ?p0 {
     ?p2 a ug:ProhibitedParticipant .
   }
  }
  WHERE {
   ?t0 ug:definition ?p0 .
   ?t1 rdfs:subClassOf [
      a owl:Restriction ;
      owl:onProperty ?s ;
      owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
   ?s rdfs:subPropertyOf ug:actantialRelation .
   GRAPH ?p0 {
     ?p1 a ?t1 .
     ?p2 a ?t2 .
     ?p1 ?s ?p2 .
     ?p1 a ug:ObligatoryParticipant .
   }
  }
]]></body></rule>
<rule><body><![CDATA[
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
  CONSTRUCT {
   GRAPH ?p0 {
     ?p1 a ug:ProhibitedParticipant .
   }
  }
  WHERE {
   ?t0 ug:definition ?p0 .
   ?t1 rdfs:subClassOf [
      a owl:Restriction ;
      owl:onProperty ?s ;
      owl:minCardinality "1"^^xsd:nonNegativeInteger ] .
   ?s rdfs:subPropertyOf ug:actantialRelation .
   GRAPH ?p0 {
     ?p1 a ?t1 .
     ?p2 a ?t2 .
     ?p1 ?s ?p2 .
     ?p2 a ug:ProhibitedParticipant .
   }
  }
]]></body></rule>
<rule><body><![CDATA[
  Prefix ug: <http://ns.inria.org/ug/v1#>
  Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
  Prefix owl: <http://www.w3.org/2002/07/owl#>
```

```
247     CONSTRUCT {
248      GRAPH ?p0 {
249        ?p1 a ug:ProhibitedParticipant .
250      }
251     }
252     WHERE {
253      ?t0 ug:definition ?p0 .
254      ?t1 rdfs:subClassOf [
255         a owl:Restriction ;
256         owl:onProperty ?s ;
257         owl:maxCardinality "0"^^xsd:nonNegativeInteger ] .
258      ?s rdfs:subPropertyOf ug:actantialRelation .
259      GRAPH ?p0 {
260        ?p1 a ?t1 .
261        ?p2 a ?t2 .
262        ?p1 ?s ?p2 .
263        ?p2 a ug:ObligatoryParticipant .
264      }
265     }
266   ]]></body></rule>
267 </rdf:RDF>
```

D.1.4 Base de règles complémentaires d'inférence pour le modèle gu-usage

La base de règles suivante complémente la base de règles OWL 2 RL/RDF. Elle permet d'axiomatiser la sémantique du modèle **gu-usage**, et contribue à simuler la sémantique du formalisme des GU. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/ug-usage.rul>.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE rdf:RDF [
3  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
5  <!ENTITY rul "http://ns.inria.fr/edelweiss/2011/rule#">
6  ]>
7
8  <rdf:RDF xmlns:rdfs="&rdfs;" xmlns:rdf="&rdf;" xmlns = '&rul;' >
9    <rule><body><![CDATA[
10     Prefix ug: <http://ns.inria.org/ug/v1#>
11     Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
12     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
13     Prefix owl: <http://www.w3.org/2002/07/owl#>
14     CONSTRUCT {
15      ?t rdfs:subClassOf ug:AbsurdUnit .
16     }
17     WHERE {
18      ?t rdfs:subClassOf [ ug:obligatoryActantSlotRootOf ?s ] .
19      ?t rdfs:subClassOf ?sig .
20      ?sig owl:onProperty ?s .
21      ?sig owl:allValuesFrom/rdfs:subClassOf ug:AbsurdUnit .
22     }
23   ]]></body></rule>
24   <rule><body><![CDATA[
25     # Rule UG/RDF eq-typ
26     Prefix ug: <http://ns.inria.org/ug/v1#>
27     Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
28     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
29     Prefix owl: <http://www.w3.org/2002/07/owl#>
```

```
    CONSTRUCT {
     ?v a ?t .
    }
    WHERE {
     ?u a ?t .
     ?t rdfs:subClassOf ug:Unit .
     ?u owl:sameAs ?v .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # Rule UG/RDF a-aslot-root
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    CONSTRUCT {
     ?u a ?aslotr .
    }
    WHERE {
     ?s rdfs:subPropertyOf ug:actantialRelation .
     ?s ug:actantSlotRoot ?aslotr .
     ?u ?s ?v .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # Rule UG/RDF a-oblaslot-root
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    CONSTRUCT {
     ?u ?s [] .
    }
    WHERE {
     ?u a ?t .
     ?s ug:obligatoryActantSlotRoot ?t .
     FILTER NOT EXISTS {
        ?u ?s ?v .
     }
    }
  ]]></body></rule>
</rdf:RDF>
```

D.1.5 Base de règles pour la génération d'une base de règles d'expansion et de contraction des définitions formelles

Appliquée à une ontologie dans laquelle sont décrites des définitions formelles, la base de règles suivante génère une base des règles d'expansion et de contraction. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/ug-definition-rules-generation-rules.rul>. Les bases de règles générées permettent d'axiomatiser la sémantique du modèle **gu-usage**, et contribuent à simuler la sémantique du formalisme des GU.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://ns.inria.fr/edelweiss/2011/rule#"
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rule><body><![CDATA[
```

```
5     # generates the contraction and expansion rules set
6     # starting point, called once.
7     Prefix ug: <http://ns.inria.org/ug/v1#>
8     Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
9     Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
10    Prefix owl: <http://www.w3.org/2002/07/owl#>
11    Prefix st: <http://ns.inria.fr/sparql-template/>
12    TEMPLATE st:start {
13     "<?xml version='1.0' encoding='UTF-8'?> \n"
14     " <rdf:RDF xmlns='http://ns.inria.fr/edelweiss/2011/rule#'
            xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'> \n"
15     " "st:call-template(ug:printContractionRule)
16     " "st:call-template(ug:printExpansionRule)
17     " </rdf:RDF> \n"
18    }
19    WHERE { }
20  ]]></body></rule>
21  <rule><body><![CDATA[
22    # generates the contraction rule
23    # called for every formal definition unit type.
24    Prefix ug: <http://ns.inria.org/ug/v1#>
25    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
26    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
27    Prefix owl: <http://www.w3.org/2002/07/owl#>
28    Prefix st: <http://ns.inria.fr/sparql-template/>
29    TEMPLATE ug:printContractionRule {
30     "<rule><body><![CDATA[ \n"
31     kg:prolog()
32     "CONSTRUCT { \n"
33       st:call-template(ug:printPrint,?t,?def)
34     "} WHERE { \n"
35       st:call-template(ug:printExpansion,?t,?def)
36     " FILTER NOT EXISTS { \n"
37        st:call-template(ug:printPrint,?t,?def)
38     " } \n"
39     "}]" "]></body></rule> \n"
40    }
41    WHERE { ?t ug:definition ?def . }
42  ]]></body></rule>
43  <rule><body><![CDATA[
44    # generates the expansion rule
45    # called for every formal definition unit type.
46    Prefix ug: <http://ns.inria.org/ug/v1#>
47    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
48    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
49    Prefix owl: <http://www.w3.org/2002/07/owl#>
50    Prefix st: <http://ns.inria.fr/sparql-template/>
51    TEMPLATE ug:printExpansionRule(?t,?def) {
52     "<rule><body><![CDATA[ \n"
53     kg:prolog()
54     "CONSTRUCT { \n"
55       st:call-template(ug:printExpansion,?t,?def)
56     "} WHERE { \n"
57       st:call-template(ug:printPrint,?t,?def)
58     " FILTER NOT EXISTS { \n"
59        st:call-template(ug:printExpansion,?t,?def)
60     " } \n"
61     "}]" "]></body></rule> \n"
```

```
62    }
63    WHERE { ?t ug:definition ?def . }
64  ]]></body></rule>
65  <rule><body><![CDATA[
66    # generates the print pattern
67    Prefix ug: <http://ns.inria.org/ug/v1#>
68    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
69    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
70    Prefix owl: <http://www.w3.org/2002/07/owl#>
71    Prefix st: <http://ns.inria.fr/sparql-template/>
72    TEMPLATE ug:printPrint(?t,?def) {
73     " ?"xsd:string(?centralname) " a " st:turtle(?t) " . \n"
74     st:call-template(ug:printActantSlot,?def,?centralname) "\n"
75    }
76    WHERE {
77     GRAPH ?def {
78       ?def rdfs:label ?centralname .
79     }
80    }
81  ]]></body></rule>
82  <rule><body><![CDATA[
83    # generates the description of actantial positions of the defined unit type
84    Prefix ug: <http://ns.inria.org/ug/v1#>
85    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
86    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
87    Prefix owl: <http://www.w3.org/2002/07/owl#>
88    Prefix st: <http://ns.inria.fr/sparql-template/>
89    TEMPLATE ug:printActantSlot(?def,?centralname) {
90     " ?"xsd:string(?centralname) " " st:turtle(?marker) " " "?"xsd:string(?name) " . "
91    }
92    WHERE {
93     GRAPH ?def {
94      ?u ug:marker ?marker .
95       ?u rdfs:label ?name .
96     }
97    }
98  ]]></body></rule>
99  <rule><body><![CDATA[
100   # generates the expansion pattern
101   Prefix ug: <http://ns.inria.org/ug/v1#>
102   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
103   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
104   Prefix owl: <http://www.w3.org/2002/07/owl#>
105   Prefix st: <http://ns.inria.fr/sparql-template/>
106   TEMPLATE ug:printExpansion(?t,?def) {
107    st:call-template(ug:printExpansionNode,?def) "\n"
108    st:call-template(ug:printExpansionArc,?def) "\n"
109   }
110   WHERE { }
111 ]]></body></rule>
112 <rule><body><![CDATA[
113   # generates the description of each participant node
114   Prefix ug: <http://ns.inria.org/ug/v1#>
115   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
116   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
117   Prefix owl: <http://www.w3.org/2002/07/owl#>
118   Prefix st: <http://ns.inria.fr/sparql-template/>
119   TEMPLATE ug:printExpansionNode(?def) {
```

```
    " ?"xsd:string(?name) " a " st:turtle(?t) " ."
   }
   WHERE {
    GRAPH ?def {
      ?u a ?t .
      ?u rdfs:label ?name .
    }
    FILTER NOT EXISTS { ?t rdfs:subClassOf ug:Participant }
   }
 ]]></body></rule>
 <rule><body><![CDATA[
   # generates the description of each actantial triple
   Prefix ug: <http://ns.inria.org/ug/v1#>
   Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
   Prefix owl: <http://www.w3.org/2002/07/owl#>
   Prefix st: <http://ns.inria.fr/sparql-template/>
   TEMPLATE ug:printExpansionArc(?def) {
   " ?"xsd:string(?uname) " " st:turtle(?p) " ?"xsd:string(?vname) " ."
   }
   WHERE {
    ?p rdfs:subPropertyOf ug:actantialRelation .
    GRAPH ?def {
      ?u ?p ?v .
      ?u rdfs:label ?uname .
      ?v rdfs:label ?vname .
    }
}
  ]]></body></rule>
</rdf:RDF>
```

D.2 Instanciation pour la théorie Sens-Texte

D.2.1 Ontologie pour le modèle Sens-Texte

L'ontologie suivante décrit la partie du modèle Sens-Texte qui est indépendante des langues. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt#>.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> .
@prefix synsem: <http://www.w3.org/ns/lemon/synsem#> .
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .


<http://ns.inria.org/ug/v1/mtt/> a owl:Ontology ;
    dc:title "The Meaning-Text theory ontology (MTT)" ;
    owl:imports <http://ns.inria.org/ug/v1/> .

mtt:LexicalUnitType a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
```

```
    rdfs:label "LexicalUnitType" ;
    rdfs:comment "The class of all lexical unit types."@en .

mtt:GrammaticalUnitType a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
    rdfs:label "GrammaticalUnitType" ;
    rdfs:comment "The class of all grammatical unit types."@en .

mtt:SurfaceSemanticUnitType a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
    rdfs:label "SurfaceSemanticUnitType" ;
    rdfs:comment "The class of all surface semantic unit types."@en .

mtt:DeepSemanticUnitType a owl:Class ;
    rdfs:subClassOf ug:UnitType ;
    rdfs:label "DeepSemanticUnitType" ;
    rdfs:comment "The class of all deep semantic unit types."@en .

mtt:LexicalUnit a mtt:LexicalUnitType , owl:Class ;
    rdfs:subClassOf ug:Unit ;
    rdfs:label "LexicalUnit" ;
    rdfs:comment "The class of all lexical units."@en .

mtt:GrammaticalUnit a mtt:GrammaticalUnitType , owl:Class ;
    rdfs:subClassOf ug:Unit ;
    rdfs:label "GrammaticalUnit" ;
    rdfs:comment "The class of all grammatical units."@en .

mtt:SurfaceSemanticUnit a mtt:SurfaceSemanticUnitType , owl:Class ;
    rdfs:subClassOf ug:Unit ;
    rdfs:label "SurfaceSemanticUnit" ;
    rdfs:comment "The class of all surface semantic units."@en .

mtt:DeepSemanticUnit a mtt:DeepSemanticUnitType , owl:Class ;
    rdfs:subClassOf ug:Unit ;
    rdfs:label "DeepSemanticUnit" ;
    rdfs:comment "The class of all deep semantic units."@en .

mtt:LexicalUnitType rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ug:specializationOf ;
       owl:hasValue mtt:LexicalUnit ] .

mtt:GrammaticalUnitType rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ug:specializationOf ;
       owl:hasValue mtt:GrammaticalUnit ] .

mtt:SurfaceSemanticUnitType rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ug:specializationOf ;
       owl:hasValue mtt:SurfaceSemanticUnit ] .

mtt:DeepSemanticUnitType rdfs:subClassOf [ a owl:Restriction ;
       owl:onProperty ug:specializationOf ;
       owl:hasValue mtt:DeepSemanticUnit ] .

[] rdf:type owl:AllDisjointClasses ;
   owl:members (
     mtt:LexicalUnitType
     mtt:GrammaticalUnitType
```

```
      mtt:SurfaceSemanticUnitType
      mtt:DeepSemanticUnitType ) .

mtt:LexicalUnit owl:disjointWith mtt:SurfaceSemanticUnit .
mtt:LexicalUnit owl:disjointWith mtt:DeepSemanticUnit .
mtt:GrammaticalUnit owl:disjointWith mtt:SurfaceSemanticUnit .
mtt:GrammaticalUnit owl:disjointWith mtt:DeepSemanticUnit .
mtt:SurfaceSemanticUnit owl:disjointWith mtt:DeepSemanticUnit .


###################################
# Deep syntactic relations symbols

mtt:i a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "i" ;
    rdfs:comment "The deep syntactic actantial relation symbol i."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:ii a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "ii" ;
    rdfs:comment "The deep syntactic actantial relation symbol ii."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:iii a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "iii" ;
    rdfs:comment "The deep syntactic actantial relation symbol iii."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:iv a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "iv" ;
    rdfs:comment "The deep syntactic actantial relation symbol iv."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:v a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "v" ;
    rdfs:comment "The deep syntactic actantial relation symbol v."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:vi a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:label "vi" ;
    rdfs:comment "The deep syntactic actantial relation symbol vi."@en ;
    rdfs:domain mtt:LexicalUnit ;
    rdfs:range mtt:LexicalUnit .

mtt:attr a owl:ObjectProperty ;
    rdfs:subPropertyOf ug:circumstantialRelation ;
    rdfs:label "attr" ;
```

```
135     rdfs:comment "The deep syntactic circumstantial relation symbol attr."@en ;
136     rdfs:domain mtt:LexicalUnit ;
137     rdfs:range mtt:LexicalUnit .
138
139 mtt:coord a owl:ObjectProperty ;
140     rdfs:subPropertyOf ug:circumstantialRelation ;
141     rdfs:label "coord" ;
142     rdfs:comment "The deep syntactic circumstantial relation symbol coord."@en ;
143     rdfs:domain mtt:LexicalUnit ;
144     rdfs:range mtt:LexicalUnit .
145
146 mtt:append a owl:ObjectProperty ;
147     rdfs:subPropertyOf ug:circumstantialRelation ;
148     rdfs:label "append" ;
149     rdfs:comment "The deep syntactic circumstantial relation symbol append."@en ;
150     rdfs:domain mtt:LexicalUnit ;
151     rdfs:range mtt:LexicalUnit .
152
153 ###################################
154 # Surface semantics relations symbols
155
156 mtt:1 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
157     rdfs:subPropertyOf ug:actantialRelation ;
158     rdfs:label "1" ;
159     rdfs:comment "The surface semantics actantial relation symbol 1."@en ;
160     rdfs:domain mtt:SurfaceSemanticUnit ;
161     rdfs:range mtt:SurfaceSemanticUnit .
162
163 mtt:2 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
164     rdfs:subPropertyOf ug:actantialRelation ;
165     rdfs:label "2" ;
166     rdfs:comment "The surface semantics actantial relation symbol 2."@en ;
167     rdfs:domain mtt:SurfaceSemanticUnit ;
168     rdfs:range mtt:SurfaceSemanticUnit .
169
170 mtt:3 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
171     rdfs:subPropertyOf ug:actantialRelation ;
172     rdfs:label "3" ;
173     rdfs:comment "The surface semantics actantial relation symbol 3."@en ;
174     rdfs:domain mtt:SurfaceSemanticUnit ;
175     rdfs:range mtt:SurfaceSemanticUnit .
176
177 mtt:4 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
178     rdfs:subPropertyOf ug:actantialRelation ;
179     rdfs:label "4" ;
180     rdfs:comment "The surface semantics actantial relation symbol 4."@en ;
181     rdfs:domain mtt:SurfaceSemanticUnit ;
182     rdfs:range mtt:SurfaceSemanticUnit .
183
184 mtt:5 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
185     rdfs:subPropertyOf ug:actantialRelation ;
186     rdfs:subPropertyOf ug:actantialRelation ;
187     rdfs:label "5" ;
188     rdfs:comment "The surface semantics actantial relation symbol 5."@en ;
189     rdfs:domain mtt:SurfaceSemanticUnit ;
190     rdfs:range mtt:SurfaceSemanticUnit .
191
192 mtt:6 a ug:ActantialRelationSymbol , owl:ObjectProperty ;
```

```
193     rdfs:subPropertyOf ug:actantialRelation ;
194     rdfs:subPropertyOf ug:actantialRelation ;
195     rdfs:label "6" ;
196     rdfs:comment "The surface semantics actantial relation symbol 6."@en ;
197     rdfs:domain mtt:SurfaceSemanticUnit ;
198     rdfs:range mtt:SurfaceSemanticUnit .
199
200 ###################################
201 # Link between unit types of different levels
202
203 mtt:deepSense a owl:ObjectProperty ;
204     rdfs:label "deepSense" ;
205     rdfs:comment "The subject, a surface semantic unit type, has the object as deep semantic
            unit type."@en ;
206     rdfs:domain mtt:SurfaceSemanticUnitType ;
207     rdfs:range mtt:DeepSemanticUnitType .
208
209 mtt:surfaceSense a owl:ObjectProperty ;
210     rdfs:label "surfaceSense" ;
211     rdfs:comment "The subject, a lexical unit type, is meaningful and has the object as suface
            semantic unit type."@en ;
212     rdfs:domain mtt:LexicalUnitType ;
213     rdfs:range mtt:SurfaceSemanticUnitType .
214
215 ###################################
216 # Link between units of different levels
217
218 mtt:dSense a owl:ObjectProperty ;
219     rdfs:subPropertyOf ug:circumstantialRelation ;
220     rdfs:label "dSense" ;
221     rdfs:comment "The subject, a surface semantic unit, has the object as deep semantic
            unit."@en ;
222     rdfs:domain mtt:SurfaceSemanticUnit ;
223     rdfs:range mtt:DeepSemanticUnit .
224
225 mtt:sSense a owl:ObjectProperty ;
226     rdfs:subPropertyOf ug:circumstantialRelation ;
227     rdfs:label "sSense" ;
228     rdfs:comment "The subject, a lexical unit, is meaningful and has the object as suface
            semantic unit."@en ;
229     rdfs:domain mtt:LexicalSemanticUnit ;
230     rdfs:range mtt:SurfaceSemanticUnit .
231
232 ###################################
233 # Alignment with ontolex
234
235 ug:LexicalUnitType owl:equivalentClass ontolex:LexicalEntry .
236
237 ug:DeepSemanticUnitType owl:equivalentClass ontolex:LexicalSense .
238
239 synsem:semArg rdfs:subPropertyOf ug:actantSlot .
```

D.2.2 Base de règles d'alignement avec Ontolex

La base de règles suivante complémente la base de règles OWL 2 RL/RDF. Elle permet d'assurer l'interopérabilité entre les modèles **gu-langue** et **ontolex**. Elle est accessible à l'URL

<http://ns.inria.org/ug/v1/mtt/ug-ontolex.rul>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://ns.inria.fr/edelweiss/2011/rule#"
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rule><body><![CDATA[
    # language-usage transfer rule: unit types
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    Prefix: ontolex: <http://www.w3.org/ns/ontolex#>
    Prefix: ontolex-synsem: <http://www.w3.org/ns/ontolex-synsem#>
    CONSTRUCT {
      ?t ontolex-synsem:semArg ?s .
    }
    WHERE {
      ?t a mtt:DeepSemanticUnitType .
      ?t ug:actantSlot ?s .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # language-usage transfer rule: unit types
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    Prefix: ontolex: <http://www.w3.org/ns/ontolex#>
    Prefix: ontolex-synsem: <http://www.w3.org/ns/ontolex-synsem#>
    CONSTRUCT {
      ?t1 ug:specializationOf [
        a ug:Signature ;
        ug:onActantSlot ?s ;
        ug:allUnitsFrom ?t2 ] .
    }
    WHERE {
      ?t1 a ontolex:LexicalSense ;
        ontolex-synsem:semArg ?s .
      ?t2 a ontolex:LexicalSense ;
        ontolex-synsem:isA ?s .
    }
  ]]></body></rule>
  <rule><body><![CDATA[
    # language-usage transfer rule: unit types
    Prefix ug: <http://ns.inria.org/ug/v1#>
    Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
    Prefix owl: <http://www.w3.org/2002/07/owl#>
    Prefix: ontolex: <http://www.w3.org/ns/ontolex#>
    Prefix: ontolex-synsem: <http://www.w3.org/ns/ontolex-synsem#>
    CONSTRUCT {
      ?t1 a ontolex:LexicalSense ;
        ontolex-synsem:semArg ?s .
      ?t2 a ontolex:LexicalSense ;
        ontolex-synsem:isA ?s .
    }
    WHERE {
      ?t1 ug:specializationOf [
```

```
57        a ug:Signature ;
58        ug:onActantSlot ?s ;
59        ug:allUnitsFrom ?t2 ] .
60     }
61   ]]></body></rule>
62 </rdf:RDF>
```

D.2.3 Ontologie pour le modèle Sens-Texte spécifique à une langue

D.2.3.1 Ontologie MTT-fr

L'ontologie suivante décrit un fragment du modèle Sens-Texte spécifique au français. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt/fr#>.

```
1  @Prefix dc: <http://purl.org/dc/elements/1.1/> .
2  @Prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3  @Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4  @Prefix xml: <http://www.w3.org/XML/1998/namespace> .
5  @Prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6  @Prefix owl: <http://www.w3.org/2002/07/owl#> .
7
8  @Prefix ug: <http://ns.inria.org/ug/v1#> .
9  @Prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
10 @Prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .
11
12 <http://ns.inria.org/ug/v1/mtt/fr/> a owl:Ontology ;
13    dc:title "The french Meaning-Text theory vocabulary (MTT-fr)" ;
14    owl:imports <http://ns.inria.org/ug/v1/mtt/> .
15
16 ####################################
17 # Data categories - example of section 1.2.2.2
18
19   ### cette exhaustivité n'est pas nécessaire
20 mtt-fr:Indicatif a mtt:GrammaticalUnitType , ug:UnitType , owl:Class ;
21    rdfs:subClassOf mtt:GrammaticalUnit , ug:Unit .
22
23 mtt-fr:Present a mtt:GrammaticalUnitType , owl:Class .
24 mtt-fr:Singulier a mtt:GrammaticalUnitType , owl:Class .
25 mtt-fr:Pluriel a mtt:GrammaticalUnitType , owl:Class .
26 mtt-fr:Determine a mtt:GrammaticalUnitType , owl:Class .
27 mtt-fr:Indetermine a mtt:GrammaticalUnitType , owl:Class .
28 mtt-fr:Defini a mtt:GrammaticalUnitType , owl:Class .
29 mtt-fr:Indefini a mtt:GrammaticalUnitType , owl:Class .
30
31 ####################################
32 # Data categories hierarchical organisation - example of section 3.2.1
33
34 mtt-fr:Nombre a mtt:GrammaticalUnitType , owl:Class .
35
36 mtt-fr:Pluralisable a mtt:GrammaticalUnitType , owl:Class ;
37    rdfs:subClassOf mtt-fr:Nombre .
38
39 mtt-fr:Singulier a mtt:GrammaticalUnitType , owl:Class ;
40    rdfs:subClassOf mtt-fr:Pluralisable .
41
42 mtt-fr:Pluriel a mtt:GrammaticalUnitType , owl:Class ;
43    rdfs:subClassOf mtt-fr:Pluralisable .
```

```
44
45 [] ug:conjunctionOf ( mtt-fr:Singulier mtt-fr:Pluriel ) ;
46    ug:specialization ug:AbsurdUnit .
47
48 ####################################
49 # Surface syntactic relations symbols - example of section 1.2.2.3
50 # le choix relation actancielle ou relation circonstancielle peut etre faux.
51
52 mtt-fr:subjectale a ug:ActantialRelationSymbol , owl:ObjectProperty ;
53    rdfs:subPropertyOf ug:actantialRelation ;
54    rdfs:label "subjectale" ;
55    rdfs:domain mtt:LexicalUnit ;
56    rdfs:range mtt:LexicalUnit .
57
58 mtt-fr:objectaleDirecte a ug:ActantialRelationSymbol , owl:ObjectProperty ;
59    rdfs:subPropertyOf ug:actantialRelation ;
60    rdfs:label "objectaleDirecte" ;
61    rdfs:domain mtt:LexicalUnit ;
62    rdfs:range mtt:LexicalUnit .
63
64 mtt-fr:restrictive rdfs:subClassOf ug:circumstantialRelation ;
65    rdfs:label "restrictive" ;
66    rdfs:domain mtt:LexicalUnit ;
67    rdfs:range mtt:LexicalUnit .
68
69 mtt-fr:prepositionnelle a ug:ActantialRelationSymbol , owl:ObjectProperty ;
70    rdfs:subPropertyOf ug:actantialRelation ;
71    rdfs:label "prepositionnelle" ;
72    rdfs:domain mtt:LexicalUnit ;
73    rdfs:range mtt:LexicalUnit .
74
75 mtt-fr:completive rdfs:subClassOf ug:circumstantialRelation ;
76    rdfs:label "completive" ;
77    rdfs:domain mtt:LexicalUnit ;
78    rdfs:range mtt:LexicalUnit .
79
80 mtt-fr:phraseologique a ug:ActantialRelationSymbol , owl:ObjectProperty ;
81    rdfs:subPropertyOf ug:actantialRelation ;
82    rdfs:label "phraseologique" ;
83    rdfs:domain mtt:LexicalUnit ;
84    rdfs:range mtt:LexicalUnit .
85
86 mtt-fr:modificative rdfs:subClassOf ug:circumstantialRelation ;
87    rdfs:label "modificative" ;
88    rdfs:domain mtt:LexicalUnit ;
89    rdfs:range mtt:LexicalUnit .
90
91 mtt-fr:determinative a ug:ActantialRelationSymbol , owl:ObjectProperty ;
92    rdfs:subPropertyOf ug:actantialRelation ;
93    rdfs:label "determinative" ;
94    rdfs:domain mtt:LexicalUnit ;
95    rdfs:range mtt:LexicalUnit .
```

D.2.3.2 Ontologie MTT-en

L'ontologie suivante décrit un fragment du modèle Sens-Texte spécifique à l'anglais. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt/en#>.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-en: <http://ns.inria.org/ug/v1/mtt/en/> .

<http://ns.inria.org/ug/v1/mtt/en/> a owl:Ontology ;
    dc:title "The english Meaning-Text theory vocabulary (MTT-en)" ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/> .

####################################
# Data categories - example of section 8.2.3

mtt-en:Active a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .

mtt-en:Indicative a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .

mtt-en:Present a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .

mtt-en:Singular a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .

mtt-en:Plural a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .

mtt-en:Definite a mtt:GrammaticalUnitType , owl:Class ;
  rdfs:subClassOf mtt:GrammaticalUnit .
```

D.2.4 Fragments de dictionnaires explicatifs et combinatoires

D.2.4.1 Ontologie ecd-fr

L'ontologie suivante décrit un fragment du Dictionnaire Explicatif et Combinatoire français. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt/fr/ecd#>.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-fr: <http://ns.inria.org/ug/v1/mtt/fr#> .
@prefix ecd-fr: <http://ns.inria.org/ug/v1/mtt/fr/ecd#> .

<http://ns.inria.org/ug/v1/mtt/fr/ecd/> a owl:Ontology ;
    dc:title "The french explanatory and combinatorial dictionary (MTT-fr-ecd)" ;
```

```
15    owl:imports <http://ns.inria.org/ug/v1/mtt/fr/> .
16
17 ####################################
18 # deep semantic unit types - example of section 1.2.2.1
19
20 ### cette exhaustivité n'est pas nécessaire
21 ecd-fr:dsem-TempsB1 a mtt:DeepSemanticUnitType , ug:UnitType , owl:Class ;
22    rdfs:subClassOf mtt:DeepSemanticUnit , ug:Unit .
23
24 ecd-fr:dsem-Certain1A3 a mtt:DeepSemanticUnitType , owl:Class .
25 ecd-fr:dsem-Maintenant1 a mtt:DeepSemanticUnitType , owl:Class .
26 ecd-fr:dsem-Orwell a mtt:DeepSemanticUnitType , owl:Class .
27 ecd-fr:dsem-Causer1 a mtt:DeepSemanticUnitType , owl:Class .
28 ecd-fr:dsem-Engage3 a mtt:DeepSemanticUnitType , owl:Class .
29 ecd-fr:dsem-Devenir1 a mtt:DeepSemanticUnitType , owl:Class .
30 ecd-fr:dsem-Politique3 a mtt:DeepSemanticUnitType , owl:Class .
31 ecd-fr:dsem-Meilleur2 a mtt:DeepSemanticUnitType , owl:Class .
32 ecd-fr:dsem-Oeuvre5 a mtt:DeepSemanticUnitType , owl:Class .
33 ecd-fr:dsem-TousB1 a mtt:DeepSemanticUnitType , owl:Class .
34
35
36 ####################################
37 # surface semantic unit types - example of section 1.2.2.2
38
39 ### cette exhaustivité n'est pas nécessaire
40 ecd-fr-ssem:Oper1 a mtt:SurfaceSemanticUnitType , ug:UnitType , owl:Class ;
41    rdfs:subClassOf mtt:SurfaceSemanticUnit , ug:Unit .
42
43 ecd-fr-ssem:NePas a mtt:SurfaceSemanticUnitType , owl:Class .
44 ecd-fr-ssem:Orwell a mtt:SurfaceSemanticUnitType , owl:Class .
45 ecd-fr-ssem:Doute a mtt:SurfaceSemanticUnitType , owl:Class .
46 ecd-fr-ssem:Effet a mtt:SurfaceSemanticUnitType , owl:Class .
47 ecd-fr-ssem:Pos2 a mtt:SurfaceSemanticUnitType , owl:Class .
48 ecd-fr-ssem:Engagement a mtt:SurfaceSemanticUnitType , owl:Class .
49 ecd-fr-ssem:Qualite a mtt:SurfaceSemanticUnitType , owl:Class .
50 ecd-fr-ssem:Oeuvre a mtt:SurfaceSemanticUnitType , owl:Class .
51 ecd-fr-ssem:Politique a mtt:SurfaceSemanticUnitType , owl:Class .
52
53
54 ####################################
55 # lexical unit types - example of section 1.2.2.3
56
57 ### cette exhaustivité n'est pas nécessaire
58 ecd-fr:Orwell a mtt:LexicalUnitType , ug:UnitType , owl:Class ;
59    rdfs:subClassOf mtt:LexicalUnit , ug:Unit .
60
61 ecd-fr:Ne a mtt:LexicalUnitType , owl:Class .
62 ecd-fr:Pas a mtt:LexicalUnitType , owl:Class .
63 ecd-fr:Avoir a mtt:LexicalUnitType , owl:Class .
64 ecd-fr:De a mtt:LexicalUnitType , owl:Class .
65 ecd-fr:Doute a mtt:LexicalUnitType , owl:Class .
66 ecd-fr:Quant a mtt:LexicalUnitType , owl:Class .
67 ecd-fr:A a mtt:LexicalUnitType , owl:Class .
68 ecd-fr:Le a mtt:LexicalUnitType , owl:Class .
69 ecd-fr:Effet a mtt:LexicalUnitType , owl:Class .
70 ecd-fr:Positif a mtt:LexicalUnitType , owl:Class .
71 ecd-fr:Son a mtt:LexicalUnitType , owl:Class .
72 ecd-fr:Engagement a mtt:LexicalUnitType , owl:Class .
```

```
ecd-fr:Politique a mtt:LexicalUnitType , owl:Class .
ecd-fr:Sur a mtt:LexicalUnitType , owl:Class .
ecd-fr:Qualite a mtt:LexicalUnitType , owl:Class .
ecd-fr:Oeuvre a mtt:LexicalUnitType , owl:Class .


####################################
# lexical unit types - example of section 3.2.1

ecd-fr:Football a mtt:LexicalUnitType , owl:Class ;
    rdfs:subClassOf mtt-fr:Singulier .

ecd-fr:Lunettes a mtt:LexicalUnitType , owl:Class ;
    rdfs:subClassOf mtt-fr:Pluriel .

ecd-fr:Cheval a mtt:LexicalUnitType , owl:Class ;
    rdfs:subClassOf mtt-fr:Pluralisable .
```

D.2.4.2 Ontologie ecd-en

L'ontologie suivante décrit un fragment du Dictionnaire Explicatif et Combinatoire français. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt/en/ecd#>.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-en: <http://ns.inria.org/ug/v1/mtt/en#> .
@prefix ecd-en: <http://ns.inria.org/ug/v1/mtt/en/ecd#> .

<http://ns.inria.org/ug/v1/mtt/en/ecd/> a owl:Ontology ;
    dc:title "The english explanatory and combinatorial dictionary (MTT-en-ecd)" ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/en/> .

####################################
# deep semantic unit types - running example: to eat - to graze


ecd-en:dsem-eater a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:domain mtt:DeepSemanticUnit ;
    rdfs:range mtt:DeepSemanticUnit .

ecd-en:dsem-eaten a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:domain mtt:DeepSemanticUnit ;
    rdfs:range mtt:DeepSemanticUnit .

ecd-en:dsem-container a ug:ActantialRelationSymbol , owl:ObjectProperty ;
    rdfs:subPropertyOf ug:actantialRelation ;
    rdfs:domain mtt:DeepSemanticUnit ;
    rdfs:range mtt:DeepSemanticUnit .
```

```
35
36 ecd-en:dsem-ToEat a mtt:DeepSemanticUnitType , ug:UnitType , owl:Class ;
37    rdfs:subClassOf mtt:DeepSemanticUnit , ug:Unit ;
38    rdfs:subClassOf [ ug:obligatoryActantSlotRootOf ecd-en:dsem-eater ] ;
39    rdfs:subClassOf [ a ug:Signature ;
40       ug:onActantSlot ecd-en:dsem-eater ;
41       ug:allUnitsFrom ecd-en:dsem-Animal ] ;
42    rdfs:subClassOf [ ug:obligatoryActantSlotRootOf ecd-en:dsem-eaten ] ;
43    rdfs:subClassOf [ a ug:Signature ;
44       ug:onActantSlot ecd-en:dsem-eater ;
45       ug:allUnitsFrom ecd-en:dsem-Thing ] ;
46    rdfs:subClassOf [ ug:actantSlotRootOf ecd-en:dsem-container ] .
47
48 ecd-en:dsem-ToGraze a mtt:DeepSemanticUnitType , ug:UnitType , owl:Class ;
49    rdfs:subClassOf mtt:DeepSemanticUnit , ug:Unit ;
50    rdfs:subClassOf [ ug:prohibitedActantSlotRootOf ecd-en:dsem-container ] ;
51    rdfs:subClassOf [ a ug:Signature ;
52       ug:onActantSlot ecd-en:dsem-eaten ;
53       ug:allUnitsFrom ecd-en:dsem-Vegetal ] .
54
55 ###############################
56 # Representation of the Unit Graph of figure 8.1
57
58
59 ecd-en:dsem-negated a ug:ActantialRelationSymbol , owl:ObjectProperty ;
60    rdfs:domain mtt:DeepSemanticUnit ;
61    rdfs:range mtt:DeepSemanticUnit .
62
63 ecd-en:dsem-revulsed a ug:ActantialRelationSymbol , owl:ObjectProperty ;
64    rdfs:domain mtt:DeepSemanticUnit ;
65    rdfs:range mtt:DeepSemanticUnit .
66
67 ecd-en:dsem-revulser a ug:ActantialRelationSymbol , owl:ObjectProperty ;
68    rdfs:domain mtt:DeepSemanticUnit ;
69    rdfs:range mtt:DeepSemanticUnit .
70
71 ecd-en:dsem-seeer a ug:ActantialRelationSymbol , owl:ObjectProperty ;
72    rdfs:domain mtt:DeepSemanticUnit ;
73    rdfs:range mtt:DeepSemanticUnit .
74
75 ecd-en:dsem-seeen a ug:ActantialRelationSymbol , owl:ObjectProperty ;
76    rdfs:domain mtt:DeepSemanticUnit ;
77    rdfs:range mtt:DeepSemanticUnit .
78
79 ecd-en:dsem-dead a ug:ActantialRelationSymbol , owl:ObjectProperty ;
80    rdfs:domain mtt:DeepSemanticUnit ;
81    rdfs:range mtt:DeepSemanticUnit .
82
83 ecd-en:dsem-John a mtt:DeepSemanticUnitType , owl:Class .
84 ecd-en:dsem-No a mtt:DeepSemanticUnitType , owl:Class .
85 ecd-en:dsem-Revulsion a mtt:DeepSemanticUnitType , owl:Class .
86 ecd-en:dsem-See a mtt:DeepSemanticUnitType , owl:Class .
87 ecd-en:dsem-Dead a mtt:DeepSemanticUnitType , owl:Class .
88 ecd-en:dsem-Animal a mtt:DeepSemanticUnitType , owl:Class .
89
90 ecd-en:ssem-John a mtt:SurfaceSemanticUnitType , owl:Class .
91 ecd-en:ssem-Not a mtt:SurfaceSemanticUnitType , owl:Class .
92 ecd-en:ssem-Revulsion a mtt:SurfaceSemanticUnitType , owl:Class .
```

```
ecd-en:ssem-See a mtt:SurfaceSemanticUnitType , owl:Class .
ecd-en:ssem-Dead a mtt:SurfaceSemanticUnitType , owl:Class .
ecd-en:ssem-Animal a mtt:SurfaceSemanticUnitTypet , owl:Class .

ecd-en:Oper1 a mtt:LexicalUnitType , owl:Class .
ecd-en:John a mtt:LexicalUnitType , owl:Class .
ecd-en:Revulsion a mtt:LexicalUnitType , owl:Class .
ecd-en:No a mtt:LexicalUnitType , owl:Class .
ecd-en:Sight a mtt:LexicalUnitType , owl:Class .
ecd-en:Dead a mtt:LexicalUnitType , owl:Class .
ecd-en:Animal a mtt:LexicalUnitType , owl:Class .
```

D.2.5 Représentations linguistiques

L'ontologie suivante décrit quelques exemples de représentations lingusitiques en anglais. Elle est accessible à l'URL <http://ns.inria.org/ug/v1/mtt/en/example#>.

```
@prefix ex: <http://ns.inria.org/ug/v1/mtt/en/example#> .
@prefix ug: <http://ns.inria.org/ug/v1#> .
@prefix mtt: <http://ns.inria.org/ug/v1/mtt#> .
@prefix mtt-en: <http://ns.inria.org/ug/v1/mtt/en/> .
@prefix ecd-en: <http://ns.inria.org/ug/v1/mtt/fr/ecd#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


################################
# Representation of the Unit Graph of figure 8.1

<http://ns.inria.org/ug/v1/mtt/en/example#> a owl:Ontology ;
    owl:imports <http://ns.inria.org/ug/v1/mtt/en/ecd#> ;
    rdfs:comment "John feels no revulsion at the sight of a dead animal" .

_:djohn a ecd-en:dsem-John .
_:dnot a ecd-en:dsem-Not .
_:drevulsion a ecd-en:dsem-Revulsion .
_:dsee a ecd-en:dsem-See .
_:ddead a ecd-en:dsem-Dead .
_:danimal a ecd-en:dsem-Animal .

_:dnot ecd-en:dsem-negated _:drevulsion .
_:drevulsion ecd-en:dsem-revulsed _:djohn .
_:drevulsion ecd-en:dsem-revulser _:dsee .
_:dsee ecd-en:dsem-seeer _:djohn .
_:dsee ecd-en:dsem-seeen _:danimal .
_:ddead ecd-en:dsem-dead _:ddead .

_:sjohn a ecd-en:ssem-John .
_:snot a ecd-en:ssem-Not .
_:srevulsion a ecd-en:ssem-Revulsion .
_:ssee a ecd-en:ssem-See .
_:sdead a ecd-en:ssem-Dead .
_:sanimal a ecd-en:ssem-Animal .

_:snot mtt-ssem:1 _:srevulsion .
_:srevulsion mtt-ssem:1 _:sjohn .
_:srevulsion mtt-ssem:2 _:ssee .
_:ssee mtt-ssem:1 _:sjohn .
```

```
_:ssee mtt-ssem:2 _:sanimal .
_:sdead mtt-ssem:1 _:sanimal .

_:sjohn mtt:deepSense _:djohn .
_:snot mtt:deepSense _:dnot .
_:srevulsion mtt:deepSense _:drevulsion .
_:ssee mtt:deepSense _:dsee .
_:sdead mtt:deepSense _:ddead .
_:sanimal mtt:deepSense _:danimal .

_:oper a ecd-en:Oper1 , mtt-en:Active , mtt-en:Indicative , mtt-en:Present .
ex:i01 a ecd-en:John , mtt-en:Singular , mtt-en:Definite .
_:revulsion a ecd-en:Revulsion , mtt-en:Singular , mtt-en:Indefinite .
_:no a ecd-en:No .
_:sight a ecd-en:Sight , mtt-en:Singular , mtt-en:Definite .
_:animal a ecd-en:Animal , mtt-en:Singular , mtt-en:Indefinite .
_:dead a ecd-en:Dead .

_:oper mtt-dsyn:i ex:i01 .
_:oper mtt-dsyn:ii _:revulsion .
_:revulsion mtt-dsyn:attr _:no .
_:revulsion mtt-dsyn:ii _:sight .
_:sight mtt-dsyn:i ex:01 .
_:sight mtt-dsyn:ii _:animal .
_:animal mtt-ssem:attr _:dead .

ex:01 mtt:surfaceSense _:sjohn .
_:no mtt:surfaceSense _:snot .
_:revulsion mtt:surfaceSense _:srevulsion .
_:sight mtt:surfaceSense _:ssee .
_:dead mtt:surfaceSense _:sdead .
_:animal mtt:surfaceSense _:sanimal .
```

Représentation des connaissances sémantiques lexicales de la Théorie Sens-Texte : Conceptualisation, représentation, et opérationnalisation des définitions lexicographiques

Résumé : Nous présentons notre recherche en ingénierie des connaissances appliquée à la linguistique. Plus particulièrement, aux prédicats linguistiques, aux représentations linguistiques, et aux définitions lexicographiques de la théorie linguistique Sens-Texte (TST). Nous adoptons une méthodologie en trois étapes.

Nous étudions dans un premier temps la conceptualisation de la TST, et montrons en quoi elle devrait être étendue pour faciliter une formalisation ultérieure. Nous justifions en particulier la nécessité de définir un nouveau niveau de représentation sémantique profond, basé sur des graphes. Nous y définissons la notion de type d'unité sémantique profonde et sa structure actancielle : un ensemble de positions actancielles signées, qui peuvent être obligatoires, optionnelles, ou interdites, et étiquetées par des rôles sémantiques lexicalisés. Nous montrons que l'organisation hiérarchique des types d'unité sémantique profonde peut correspondre à une hiérarchie de sens au sein de laquelle les structures actancielles sont héritées et spécialisées. Nous reconceptualisons les définitions lexicographiques au niveau sémantique profond, et au niveau du dictionnaire. Finalement, nous présentons un prototype d'éditeur de définitions basé sur la manipulation directe de graphes, qui permettera une intégration future de nos travaux dans des projets de lexicographie explicative et combinatoire.

Ensuite, nous proposons un formalisme de représentation des connaissances adapté à cette conceptualisation. Nous démontrons que les logiques de description et le formalisme des Graphes Conceptuels ne sont pas adaptés pour représenter les connaissances de la TST. Nous construisons alors un nouveau formalisme de représentation des connaissances adapté, dit des Graphes d'Unités.

Enfin nous étudions l'opérationnalisation du formalisme des Graphes d'Unités. Nous lui associons une sémantique formelle basée sur la théorie des modèles et l'algèbre relationnelle, et montrons que les conditions de décidabilité du raisonnement logique correspondent aux intuitions des lexicographes. Nous proposons également une implémentation du formalisme avec les standards du web sémantique, ce qui permet de profiter des architectures existantes pour le partage, l'interopérationnalisation, et l'interrogation des connaissances sur le web des données lexicales liées.

Mots-clés : Représentation de Connaissances, Connaissances Linguistiques, Théorie Sens-Texte, Prédicats Linguistiques, Représentations linguistiques, Définitions Lexicographiques, Sémantique Décompositionnelle, Web des données liées

---

Meaning-Text Theory Lexical Semantic Knowledge Representation: Conceptualization, Representation, and Operationalization of Lexicographic Definitions

Abstract: We present our research in applying knowledge engineering to linguistics. In particular, to linguistic predicates, linguistic representations, and lexicographic definitions of the Meaning-Text Theory (MTT). We adopt a three-step methodology.

We first study the MTT conceptualization, and show how it should be extended to ease its formalization. We therefore justify the need of defining a new deep semantic, graph-based, representation level for the Meaning-Text model. We define the notion of deep semantic unit types and its actantial structure : a set of signed obligatory, optional or forbidden actant slots with lexicalized semantic roles as labels. We show that their hierarchical organization may correspond to a hierarchy of meanings, inside which actantial structures are inherited and specialized. We reconceptualize lexicographic definitions at the deep semantic level, and at the level of dictionaries. Finally, we present a definition editor prototype based on graph direct manipulation, which will allow us, in future work, to integrate our formal model into explanatory combinatorial lexicographic projects.

We then propose a knowledge representation formalism adapted for this conceptualization. We demonstrate that Description Logics and the Conceptual Graphs formalism do not fit our needs. This leads us to construct a new knowledge representation formalism : the Unit Graphs formalism.

Finally, we operationalize the Unit Graphs formalism. We assign it a formal semantic model, which we create based on model theory and relational algebra. We then show that the reasoning decidability conditions match the intuitions that lexicographers have. We also provide an implementation using semantic web standards, which enable us to use existing architectures for sharing, interoperability, and knowledge querying over the web of lexical linked data.

Keywords: Knowledge Representation, Linguistic Knowledge, Meaning-Text Theory, Linguistic Predicates, Linguistic Representations, Lexicographic Definitions, Decompositional Semantics, Web of linked data