# Modélisation, indexation et recherche de documents structurés

Franck Fourel

▶ **To cite this version:**

Franck Fourel. Modélisation, indexation et recherche de documents structurés. Autre [cs.OH]. Université Joseph-Fourier - Grenoble I, 1998. Français. NNT : . tel-00004888

**HAL Id: tel-00004888**
**https://theses.hal.science/tel-00004888**

Submitted on 19 Feb 2004

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

présentée par

**Franck Fourel**

pour obtenir le grade de Docteur
de l'Université Joseph Fourier - Grenoble 1
*(arrêtés ministériels du 5 juillet 1984 et du 30 Mars 1992)*
Spécialité : **Informatique**

---

# Modélisation, indexation et recherche de documents structurés

---

Date de soutenance : 5 Février 1998

Composition du jury :

| | |
|---|---|
| Président : | M. Jean-Pierre Giraudin |
| Rapporteurs : | M. Patrick Bosc |
| | M. Claude Chrisment |
| Examinateurs : | M. Edmond Lassalle |
| | M. Farid Ouabdesselam |
| | Mme. Marie-France Bruandet |
| | M. Philippe Mulhem |

Thèse préparée au sein du laboratoire
Communication Langagière et Interaction Personne-Système – IMAG
Université Joseph Fourier - Grenoble 1

*Je tiens à remercier :*

*Jean-Pierre Giraudin, professeur à l'Université Pierre Mendès-France, pour avoir accepté de présider ce jury de thèse.*

*Patrick Bosc, professeur à l'École Nationale Supérieure de Sciences Appliquées et de Technologie de Lannion, et Claude Chrisment, professeur à l'Université Paul Sabatier de Toulouse, pour avoir bien voulu être rapporteurs de ce travail. Je les remercie très sincèrement pour leur intérêt et pour les commentaires et critiques constructives qui ont contribué à la forme finale de la thèse.*

*Farid Ouabdesselam, professeur à l'Université Joseph Fourier, et Edmond Lassalle, ingénieur de recherche et responsable d'équipe au CNET de Lannion, pour leur aimable participation au jury et pour l'intérêt qu'ils ont manifesté pour mon travail.*

*Marie-France Bruandet, professeur à l'Université Joseph Fourier, et Philippe Mulhem, maître de conférence à l'Université Pierre Mendès-France, pour le temps qu'ils ont su me consacrer, le soutien qu'ils m'ont apporté, leurs remarques pertinentes ainsi que leurs multiples lectures du manuscrit. Je leur dois l'aboutissement de ce travail.*

*Yves Chiaramella, professeur à l'Université Joseph Fourier et directeur du CLIPS, pour m'avoir accueilli au sein de ce laboratoire.*

*Catherine Berrut, membre éminente de l'équipe MRIM, pour son amitié, sa vision générale de la recherche d'informations ainsi que pour ses conseils permanents : "tu ne vas quand même pas mettre CES chaussures pour ta soutenance !!"*

*Les membres de l'équipe MRIM, Jean-Pierre, Nathalie et Iadh qui contribuent activement à l'ambiance qui règne dans cette équipe.*

*Les résidants de notre bâtiment B, et en particulier les membres de l'équipe IHM (Laurence, Joëlle, Daniel, François, Eric, Frédéric, David) qui rendent le mot "travail" beaucoup plus sympathique.*

*Tous les autres pour le plaisir que j'ai eu à les cotoyer, Laurent et sa logique quasi industrielle, Arnaud et Sébastien et leurs chasses aux marmottes, Hervé, Jean-Luc et Stella, Tes et Nath, Mathieu et Laure, Céline et enfin Françoise.*

*J'attends maintenant avec impatience que les deux autres "freu" (François et Frédéric) ainsi que quelques autres (Céline, Laure, Arnaud, Seb et Anne) aient eux aussi la joie de rédiger, à leur tour, leurs propres remerciements.*

*Enfin, je tiens plus particulièrement à remercier mes parents et ma soeur, Nathalie, pour leur présence et l'affection qu'ils m'ont toujours témoigné.*

# Table des matières

# Table des figures

# Chapitre 1

# Introduction

> "As we embark on the information age, the use of electronic information is spreading through all sectors of society, both nationally and internationally"
>
> Nabil R. Adam - "International Journal on Digital Libraries"

La technologie informatique propose depuis plusieurs années des logiciels grand public (Hypercard, Macromedia Director,..) ou spécialisés (librairies digitales, applications médicales, ...) capables de gérer simultanément des textes, des images, des vidéos ou encore des sons. Il est donc de plus en plus aisé et de plus en plus courant de créer des documents dits *multimédia* à partir de son ordinateur personnel ou dans un environnement professionnel.

D'un autre côté, l'essor constant des réseaux de communication, permettant l'échange et le transfert des informations, ne cesse d'accroître l'intérêt pour ces documents multimédia, ou plus généralement pour les documents dits *électroniques*.

Dans le cadre d'une utilisation personnelle, la création de documents à l'aide de la technologie multimédia a des allures ludiques. Toutefois, parmi les possesseurs d'ordinateur personnel, qui peut avouer n'avoir jamais été confronté à la recherche du bon document contenant la bonne information. La technologie multimédia n'a pas pour l'instant répondu à ce problème. Elle aurait même tendance à l'amplifier de par la nature même des documents issus de cette technologie. Pour une utilisation personnelle, il est toujours possible de fouiller un disque dur de quelques giga-octets, mais cela est long et fastidieux. Dans le cadre d'une utilisation professionnelle, où la taille des disques est multipliée par 10 ou 100, où les données sont réparties sur des machines inter-connectées, et où le temps est précieux cette fouille devient irréaliste et la nécessité d'avoir des outils qui effectuent cette recherche à votre place est alors évidente.

Offrir des outils de création n'est donc pas suffisant, il faut aussi proposer des solutions pour le stockage, la gestion et la recherche de ces documents. Actuellement, les solutions pour la création des documents multimédias sont nombreuses alors que les alternatives pour leur gestion sont peu développées. Les documents sont gérés sous la forme de fichiers par des systèmes tels que des éditeurs ou des navigateurs qui ne favorisent pas leur réutilisation.

Nous pouvons citer l'exemple caractéristique des documents HTML (HyperText Markup

Language) du World Wide Web, et plus globalement l'exemple des sites Web. Un site Web contient l'ensemble des documents HTML visibles ainsi que les données qui constituent ces documents: texte, images, vidéos, séquences audios, etc. Il existe de nombreux éditeurs WYSIWYG ("What You See Is What You Get") pour ces documents (Netscape Communicator, Microsoft Internet Explorer, AOLpress, Amaya, etc) mais très peu sont capables de gérer un site complet, c'est-à-dire un ensemble de documents, ou de proposer des fonctionnalités de stockage et/ou de recherches qui soient adaptées à ces documents.

Pourtant, il existe des solutions tant pour le stockage que pour la recherche. Pour stocker ces documents, la solution qui semble la plus évidente et la plus naturelle est l'utilisation d'un système de gestion de bases de données (SGBD). Ces systèmes autorisent la manipulation de grandes quantités de données en assurant leur persistance, leur sécurité mais aussi leur partage entre de multiples utilisateurs et/ou de multiples machines. Pour rechercher, et surtout retrouver, les documents répondant à un besoin précis, les systèmes de recherche d'informations (SRI) proposent des techniques pour accéder aux documents en utilisant leur contenu sémantique.

Nous estimons donc aujourd'hui que la combinaison des SGBD et des SRI assurera une meilleure gestion d'une base de documents HTML, ou plus généralement une meilleure gestion d'une base de documents structurés. Cependant, les caractéristiques de ces documents introduisent des problématiques nouvelles tant pour leur stockage que pour leur recherche. Il faut d'une part définir un modèle pour le stockage de ces documents conservant l'indépendance entre la structure du document et les données qui le composent, c'est-à-dire une séparation entre leurs aspects logique et physique. D'autre part, il faut proposer un modèle de recherche capable d'intégrer les caractéristiques structurelles des documents et les spécificités provenant des différents médias.

Si l'utilisateur exprime un besoin particulier, document à propos du surf des neiges, image contenant un surfeur, élément contenant une image à propos du surf, etc, il souhaite que le système lui permette d'accéder directement à la partie *pertinente* du document répondant à son besoin. Il ne souhaite ni une référence à cette partie, ni le document complet qui contient cette partie mais il veut le fragment pertinent qui peut être indifféremment un paragraphe de texte, une image ou basé sur tout autre média des documents considérés.

La gestion des documents électroniques représente donc un nouveau challenge pour tout système informatique. Pour notre part, nous allons particulièrement nous intéresser à l'usage de la structure des documents dans les fonctionnalités de recherche d'informations. Nous donnons un bref aperçu du document structuré en délimitant clairement les aspects sur lesquels nous portons notre attention. Puis nous présentons de manière générale les fonctionnalités des systèmes de recherche d'informations avant de décrire le déroulement de ce document.

## 1.1 Les documents structurés

Les premiers éditeurs de documents étaient des éditeurs de documents textuels n'intégrant que très peu de fonctionnalités. Aujourd'hui des éditeurs tels que Adobe FrameMaker, Microsoft Word ou Applix Applix*ware* proposent des environnements complets dans lesquels apparaissent des fonctionnalités de structuration des documents. Ces fonctionnalités permettent d'attacher un type particulier à chaque partie, de décrire leur contenu, etc. Avec l'apparition

d'éditeurs collaboratifs, ces fonctionnalités vont s'avérer indispensables.

L'évolution des éditeurs de documents est la conséquence du développement des besoins de chacun en terme d'informations. Le besoin de toujours avoir à sa disposition les informations s'accroît avec le développement des moyens de communications. Aujourd'hui, les documents électronique représentent un moyen d'échanger des informations. Cependant, la compréhension des documents sur des environnements hétérogènes a requis le développement de normes de représentation des documents: ODA [Ass91], SGML [Bur94] et plus récemment HyTime [Erf94]. Les modèles de représentation sous-jacents à ces normes reposent sur la structure du document, mais surtout sur la séparation explicite des structures du document. En effet, pour un document donné, il est possible de décrire au moins deux structures différentes: la *structure logique* et la *structure physique*.

*La structure logique* définit une organisation hiérarchique des données du document c'est-à-dire une organisation de l'information qui sous-tend le discours de l'auteur du document. Cette organisation s'établit autour d'abstractions représentant des parties du document: un document est composé d'un titre, d'une ou plusieurs sections, elles même sont composées d'un titre, d'une ou plusieurs sous-sections, etc.

*La structure physique* définit une organisation externe des données qui composent le document c'est-à-dire une présentation du document. Elle est dépendante de l'environnement. S'il s'agit d'une présentation "papier", c'est le format de la feuille (A4, A3, etc) ou son mode d'utilisation (portrait ou paysage) qui peuvent caractériser l'environnement. Dans le cadre d'une présentation informatique, ce sont les fonctionnalités de du système qui déterminent cet environnement. De manière générale, on considère qu'un document est composé de pages qui sont elles-même composées d'un en-tête, d'un corps et de notes de bas de pages, etc.

La distinction entre ces deux types de structure peut se faire selon deux notions: *l'indépendance vis à vis de l'environnement de présentation* et *le contenu* des documents.

La structure logique est totalement indépendante de l'environnement alors que la structure physique en dépend entièrement. Par exemple, si un document est présenté en format A4 puis en format A3, les pages de ces deux présentations ne pointeront pas sur les même données. De manière générale, il n'est pas intéressant de rechercher l'information en fonction de cette structure puisqu'elle dépend de l'environnement et elle est modifiée si l'environnement change. Il est toujours possible de faire référence à la structure physique dans l'interrogation en indiquant explicitement dans la requête le mode de présentation qui est considéré.

La notion de contenu, quant à elle, est à rapprocher de l'information véhiculée par un document et par ses parties. Nous avons défini la structure logique par sa capacité à organiser l'information. Par exemple, dans notre document, nous avons défini cette partie, la section 1.1, pour présenter les documents structurés puis la partie suivante, la section 1.2, pour présenter la recherche d'information. Nous avons choisi ce découpage car nous traitons des concepts différents dans ces deux parties. Il y a donc une dépendance entre la structure logique et les informations thématiques du document, c'est-à-dire son *contenu sémantique*. Cette dépendance n'est pas aussi explicite pour la structure physique.

Il existe d'autres types de structures sur lesquels nous revenons plus tard dans ce document. Toutefois, nous pouvons d'ores et déjà affirmer que la structure logique est la structure

la plus utilisée et la plus utilisable pour tout système cherchant à gérer des documents structurés.

## 1.2 Les systèmes de recherche d'informations

Dans les systèmes de recherche d'information (SRI), le besoin de l'utilisateur est exprimé par une requête portant sur le contenu sémantique des documents recherchés.

Les systèmes de recherche d'information (SRI) diffèrent donc des systèmes de gestion de base de données (SGBD) de par le fait qu'ils analysent le contenu des documents qu'ils gèrent et qu'ils ne se contentent pas de comparaisons strictes entre les termes des requêtes et les éléments de la base.

Un SRI est classiquement composé de deux parties : la première a trait à la définition d'un modèle de représentation des documents, et la seconde porte sur l'interrogation de ces documents. La première partie utilise un modèle qui permet d'exprimer le contenu sémantique de chaque document. La seconde partie contient une phase de compréhension des requêtes et une phase de correspondance entre le contenu sémantique des documents et de la requête.

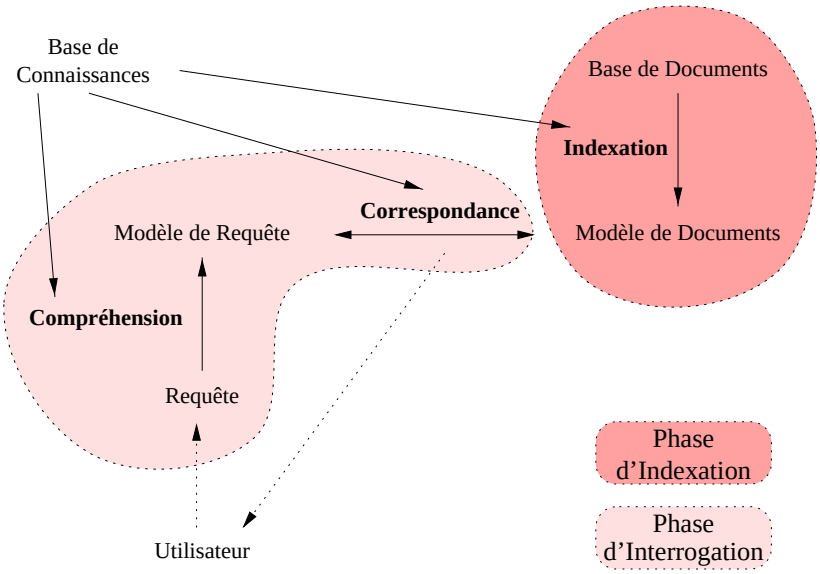



**Figure 1.1.** Schéma des fonctionnalités d'un système de recherche d'information

La figure 1.1 décrit les fonctionnalités d'un SRI. Nous revenons plus largement dans les sections suivantes sur les phases d'indexation et d'interrogation. Cependant, nous allons voir à partir de ce schéma les points communs et surtout les différences entre les SGBD et les SRI.

Tout d'abord ces deux types de systèmes ont la particularité de gérer des ensembles de données. Ils intègrent donc des fonctions pour modéliser ces données (Langage de Description de Données), mais aussi pour manipuler et pour interroger ces données (Langage de Manipulation de Données). Dans chacun des systèmes l'utilisateur a accès aux données via des

langages d'interrogation. Toutefois, les processus respectifs, mis en jeu lors d'une interrogation sont différents. Ainsi les SGBD appliquent un processus de correspondance *strict* alors que les SRI basent ce processus sur la notion de *pertinence* d'un document à la requête. Nous revenons sur cette différence dans la section 1.2.2.

Des travaux récents [VAB96] ont montré qu'une intégration des SGBD et des SRI dans un même système présente des avantages indéniables. La figure 1.2 présente un type d'intégration: le SGBD gère les données (stockage, méthodes d'accès, ...) et des extensions SRI (méthodes d'indexation et méthodes d'interrogation) sont intégrées au niveau de l'application et construites au dessus du langage de manipulation des données (LMD) du SGBD. Il est possible, comme cela est par exemple proposé dans SQL-3 [Bla95], d'adjoindre au LMD des fonctionnalités de recherche provenant des SRI. Il s'agit alors d'une intégration plus forte permettant à des applications différentes de partager les mêmes fonctionnalités du SGBD.

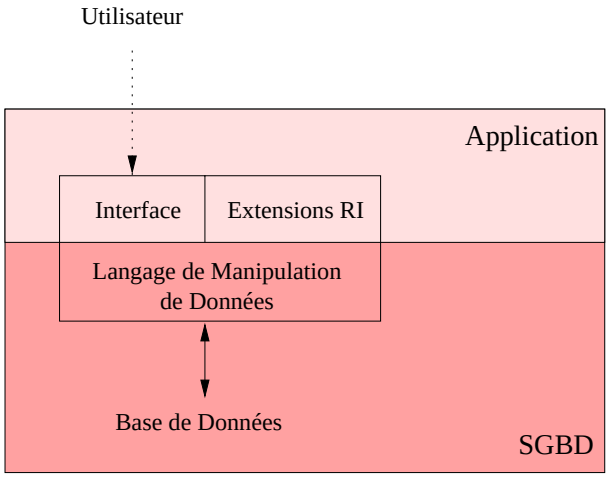



**Figure 1.2.** Intégration SRI-SGBD

### 1.2.1 La phase d'indexation

Cette phase, spécifique aux SRI, permet d'extraire les concepts des documents de la base. Dans un SRI un document est considéré comme un support qui véhicule de l'information. La phase d'indexation permet de capturer cette information et de la représenter selon un modèle: le *modèle de documents*. L'information capturée correspond au contenu sémantique du document et la représentation de ce contenu sémantique est appelée un *index* de document.

Le contenu sémantique d'un document s'exprime à travers un langage d'indexation. Ce langage dépend du modèle de recherche. Il peut reprendre un formalisme de représentation de connaissances (graphes conceptuels, logiques descriptives, etc) ou bien être dépendant du modèle de recherche : ensemble de mots-clés, vecteur de termes d'indexation, etc.

Cependant ces derniers types de langage d'indexation, instanciés par des mots-clés, sont ambigus. Un terme (ou mot-clé) a plusieurs sens (polysémie), il a également des synonymes.

D'autre part, si les langages d'indexation basés sur des formalismes de représentation de connaissance sont plus précis, leurs traitements requièrent des calculs beaucoup plus complexes et peu adéquats à la recherche. Ces types de représentation restent donc difficiles à manipuler sur de grandes quantités de données.

Le dilemme rencontré dans ces systèmes est donc assez similaire a celui d'autres domaines: *modéliser de la manière la plus précise et la plus efficace possible les documents.* Ces deux critères sont antagonistes puisque l'efficacité nous rapproche des modèles de base de données (modèle relationnel ou à objets) alors que la précision demande des modèles de représentation élaborés capables de rendre chaque concept du document.

Pour reprendre la comparaison entre SRI et SGBD, ces derniers systèmes n'analysent pas le contenu des données qu'ils gèrent. Ils utilisent des modèles pour stocker et rendre accessibles les données qui ne posent pas ce type de problème tant qu'ils gèrent des données traditionnelles: attributs formatés, champs textuels. Par contre l'introduction des données complexes dans les SGBD, données structurées et/ou multimédias, inscrit les SRI et les SGBD dans des challenges comparables. A savoir représenter ces données et y accéder efficacement.

### 1.2.2 La phase d'interrogation

La phase d'interrogation est la phase d'interaction entre le système et l'utilisateur. Ce dernier exprime son besoin d'information via un langage de requête (langue naturelle, ensemble de mots-clés, ...) que le système va se charger de traduire. Cette traduction se fait selon le *modèle de requête* et a pour but de comprendre les besoins de l'utilisateur et de les exprimer dans un formalisme similaire à celui mis en jeu lors de l'indexation des documents. Ce processus fournit une requête interne.

Suite à cette phase de compréhension de la requête, le système calcule la correspondance entre la requête interne et chaque index des documents. Ce calcul, établi par la *fonction de correspondance*, a classiquement pour résultat une liste ordonnée des documents de la base. Le premier document de cette liste est celui qui est considéré par le système comme le plus pertinent, c'est-à-dire celui qui répond le mieux à la requête, toujours d'après le système. Le dernier document est celui qui est considéré par le système comme le moins pertinent. Cette notion de pertinence, spécifique aux SRI, repose sur la proximité entre les besoins exprimés par l'utilisateur et les résultats fournis par le système. Classiquement on différencie la *pertinence utilisateur* de la *pertinence système.*

*pertinence système* : le document est jugé pertinent par le système pour une requête.

*pertinence utilisateur* : le document est jugé pertinent par l'utilisateur en fonction de ses besoins.

On juge la qualité d'un SRI en fonction de son aptitude à donner tous les documents jugés pertinents par l'utilisateur et uniquement ces documents. Deux critères sont utilisés pour mesurer les performances qualitatives d'un SRI: le rappel et la précision [SG83]. Un système *idéal* aurait un taux de rappel et un taux de précision maximal égaux à 1.

Nous donnons les définitions de ces deux critères en sachant que les travaux actuels sur la notion de pertinence dans les SRI tendent à montrer que ces critères sont dorénavant

insuffisants pour juger de la qualité d'un SRI. Il faut donc les considérer comme des indicateurs de qualité pris dans un contexte particulier, le domaine d'application du système.

*Rappel* : capacité du système à retrouver tous les documents pertinents pour l'utilisateur.

*Précision* : capacité du système à ne fournir que les documents pertinents pour l'utilisateur.

Dans un SGBD, la notion de pertinence n'existe pas, le document (ou les données) répond complètement à une requête ou ne répond pas. Le SRI cherche pour sa part à classer les documents en fonction de leur pertinence, c'est-à-dire qu'il évalue, selon un modèle interne de représentation, la distance sémantique entre la requête et les documents. Les documents les plus proches de la requête, c'est-à-dire les documents qui ont une distance sémantique à la requête la plus petite, correspondent alors aux meilleurs réponses. L'identité sémantique entre un document et une requête, c'est-à-dire une distance nulle, se traduit par une valeur élevée des deux critères suivants:

*spécificité du document pour la requête* : quantité d'informations du document qui permet de déduire la requête.

*exhaustivité du document pour la requête* : quantité d'informations du document déduite de la requête.

Un critère d'exhaustivité du document pour la requête *élevé* signifie que la plus grande partie du contenu du document est comprise dans la requête. Un critère de spécificité du document pour la requête *élevé* signifie que la plus grande partie du contenu de la requête est comprise dans le document.

Appliqué aux documents structurés, ces deux derniers critères indiquent clairement qu'il est possible de trouver un composant meilleur que les autres parmi les agrégations de composants. En effet, une agrégation (ou composition) de composants va modifier la quantité d'informations et les valeurs des deux critères d'exhaustivité et de spécificité vont évoluer. Nous allons donc chercher à modéliser le comportement des informations décrivant les composants en fonction de relations telles que l'agrégation avec pour objectif final retrouver le meilleur composant pour une requête parmi l'ensemble des composants d'un document structuré.

## 1.3 Contenu de la Thèse

Nous avons vu que les documents électroniques sont désormais créés dans un cadre proposant une structure du document, proche des normes de représentation des documents structurés (SGML, ODA, etc). Des efforts doivent maintenant se porter sur les moyens mis en oeuvre pour rechercher dans des fonds documentaires constitués de documents électroniques. Les systèmes de recherche d'informations proposent des techniques et des modèles permettant de représenter le contenu sémantique des documents et de les rechercher d'après la représentation de ce contenu. Par ailleurs, les systèmes de gestion de bases de données permettent dorénavant de stocker et de manipuler les documents électroniques.

La problématique générale à laquelle nous sommes confrontés réside dans l'application de techniques de recherches d'informations sur des documents structurés. Cette application soulève des problèmes complexes que les modèles actuels ne résolvent pas complètement.

Ainsi, s'il est dorénavant possible sur des extensions du langage OQL (Object Query Language) de spécifier une contrainte sur la structure que doit avoir le document recherché, les critères sur le contenu du document recherché sont trop pauvres pour être exploitables. De plus, ils présentent souvent la particularité d'être trop incomplets pour couvrir assez précisément le document global, c'est-à-dire qu'ils ne renseignent qu'une partie de l'ensemble des composants structurels qui forment le document global.

Nos objectifs sont multiples. Nous souhaitons proposer un mode d'interrogation du document structuré qui soit moins fortement contraint que les modes actuels, par exemple nous voulons donner la possibilité d'accéder à une partie de document en indiquant son auteur même si cette information n'était pas donnée initialement de manière explicite. Pour cela nous cherchons à rendre explicite des informations qui restent trop souvent implicites dans la structure du document. Nous avons aussi pour objectif de proposer un modèle capable de gérer des données issues de médias différents. Dans notre travail, nous nous limitons aux données textuelles et aux images fixes, mais notre modèle reste ouvert à l'accueil d'autres médias.

Nous proposons dans un premier temps un modèle de représentation du document structuré. Ce modèle repose sur les caractéristiques générales des relations de structure syntaxique, à savoir relation de composition, relation de séquence et relation de référence. Il permet d'accueillir des données textuelles et des images fixes conformément à des spécifications structurelles. En outre, il conserve une séparation explicite entre les composants structurels, issus de la structure syntaxique, et les descripteurs de ces composants, les attributs.

Nous nous intéressons ensuite à la définition d'un processus d'indexation, appelé *indexation structurelle*, explicitant la portée des informations contenues dans les attributs des composants structurels. Ce processus utilise les propriétés des relations de structure définies dans le modèle pour agrémenter la description du document. Nous définissons alors les stratégies de recherche en utilisant le résultat du processus d'indexation structurelle, et particulièrement les propriétés sur les valeurs des attributs générées lors de ce processus.

## 1.4 Organisation de la Thèse

Le plan que nous allons suivre dans ce document est le suivant :

*Chapitre 2* - Représentation et recherche de documents dans les SGBD et dans les SRI

Nous commençons par l'étude des modèles de représentations pour des données textuelles, des images fixes et des documents structurés. Nous nous intéressons plus spécifiquement à l'interrogation des documents structurés en orientant notre étude selon l'usage de la structure et l'usage du contenu sémantique dans la recherche. Nous donnons les principaux pointeurs sur les langages d'interrogation des documents structurés ainsi que les stratégies d'indexation et de recherche des SRI appliqués à des fonds composés de documents structurés.

*Chapitre 3 et 4* - Les relations de structure du document textuel et Modèle de représentation des

images

Nous proposons une étude sur les relations de structure qui organisent le document textuel. Nous donnons une représentation de ces relations pour chaque type de structure rencontré puis nous cherchons à les unifier dans un modèle homogène. Nous séparons ainsi la structure syntaxique du document textuel de sa structure sémantique.

De manière similaire, nous proposons un modèle pour les images fixes. Ce modèle de représentation repose également sur des relations, dites de structure, liant des entités d'informations.

*Chapitre 5* - Modèle de représentation du document structuré

A la suite de la description des modèles pour les documents textuels et pour les images fixes, nous proposons un modèle générique de représentation du document structuré capable d'accueillir indifféremment du texte et des images fixes. Ce modèle pour les documents structurés reprend les caractéristiques énoncées dans les chapitres 3 et 4. Par ailleurs, il intègre des descripteurs de composants structurels sous la forme d'attributs.

*Chapitre 6 et 7* - Indexation Structurelle et Interrogation

Nous abordons dans ces deux chapitres les problèmes relatifs à l'indexation et à l'interrogation des documents structurés. Nous introduisons dans un premier temps les notions de *couverture* et de *dépendance* dans le document structuré. Les systèmes traditionnels, ainsi que les modèles sous-jacents à ces systèmes, ne remplissent pas les propriétés requises pour garantir une bonne couverture du document par les informations. Nous attachons alors à la notion d'attributs une propriété dépendante des relations de structure du document: la *portée de l'attribut*. Il s'agit, à travers la portée, de définir et de contrôler la propagation des informations via les relations de structure dans le document structuré. Cette propriété augmente singulièrement la couverture du document par les attributs et les informations contenues dans ceux-ci. De plus, elle présente la particularité de rendre explicite des informations trop souvent implicites dans le document structuré, et donc d'augmenter la cohérence des valeurs des attributs.

A partir d'une classification des attributs et de leurs portées, nous établissons un protocole d'indexation incluant la portée: l'*indexation structurelle*. Nous utilisons, lors de l'interrogation, les propriétés engendrées par l'application des portées des attributs pour mettre en correspondance une requête avec les éléments d'une base de documents structurés. Dans la phase d'interrogation, nous cherchons à restituer non pas un ou plusieurs documents complets, mais des composants structurels de ces documents. Ceux qui, selon notre modèle, répondent le plus précisément.

*Chapitre 8* - Mise en oeuvre du modèle, le prototype *my Personal Daily News*

Dans ce chapitre, nous présentons notre travail dans le contexte du développement d'une application sur le système à objets $O_2$. Nous expérimentons notre approche sur un corpus constitué de quotidiens d'informations que nous représentons à l'aide de notre modèle. Ces documents, fortement structurés, présentent l'avantage de mêler indifféremment dans leur contenu textes et images.

Nous décrivons les choix d'implantation que nous avons faits ainsi que les fonctionnalités principales de l'application *my Personal Daily News*.

*Chapitre 9* - Conclusion

La conclusion nous permet de situer nos travaux par rapport à l'existant, en fixant les apports que nous faisons. Nous allons de plus exprimer les suites à donner à ce travail en particulier au niveau de l'interrogation des documents structurés et de la forme des réponses fournies par le système.

# Première partie

# Représentation et recherche de documents dans les SGBD et dans les SRI

---

*Obtenir les informations pertinentes répondant à un besoin précis, organisées, triées et exploitables, signifie que nous maitrisons l'accès à l'information. Il n'existe pas de méthode idéal pour acquérir cette maîtrise mais il existe différentes approches que nous allons chercher à identifier.*
*Les systèmes de gestion de bases de données (SGBD) propose des méthodes basées sur une structuration de l'information et des chemins d'accès prédéfinis.*
*Les systèmes de recherche d'informations (SRI) exploitent le contenu des documents pour identifier les sujets qu'ils recouvrent afin de proposer des accès par le "contenu sémantique".*
*Enfin, les systèmes hypertextuels cherchent eux-aussi à organiser les documents en les inter-reliant et en proposant des méthodes pour consulter les documents en suivant ces relations.*
*Dans ce chapitre, nous donnons un aperçu des différentes méthodes d'accès que nous avons étudiées. Nous revenons tout d'abord plus complètement sur les modes d'accès qui sont actuellement proposées puis nous les détaillons en différenciant les types de documents traités (documents non structurés, documents textuels structurés, images fixes et documents multimédias). Nous nous intéressons principalement à la manière dont les documents sont représentés et aux caractéristiques des fonctionnalités de recherche qui fournissent des accès aux documents.*

---

# Chapitre 2

# Représentation et recherche de documents

Nous abordons les différents aspects de la problématique des documents structurés dans des systèmes de recherche de données et les systèmes de recherche d'informations. Nous identifions et comparons les différentes approches rencontrées.

## 2.1 Introduction à la problématique

Le document est un objet intelligible qui véhicule de l'information. Sa présentation, sa forme et son contenu, qui dépendent du support physique, évoluent. Le document, qui a pour support physique premier le *papier*, utilise aujourd'hui des supports électroniques. Cette évolution offre de nouvelles possibilités tant pour le contenu du document que pour sa forme. Il est aujourd'hui aisé de produire un document comportant du texte, des images, des graphiques, des données audio ou encore de la vidéo. Le processus de production du document dit *électronique* profite donc pleinement des nouvelles technologies informatiques.

Toutefois, si la technologie évolue et les informations contenues dans le document prennent des formes aussi diverses et variées, le processus de production du document électronique reste similaire au processus de production du document écrit. Il est décomposable en deux phases [Dup95]: un *processus rédactionnel* afin de recueillir et de valider l'information et un *processus éditorial* pour mettre en forme le document. C'est lors de cette mise en forme que les informations qui vont constituer les données du document vont être organisées, ou autrement dit *structurées*. C'est donc cette organisation qui va rendre le document intelligible.

L'intérêt de cette mise en forme dans le processus de production du document est aujourd'hui amplifié par les besoins d'échanges d'informations. Pour établir une communication entre deux personnes, il est nécessaire d'avoir un langage commun. Il en va de même pour les documents et ce sont alors les normes de représentation des documents qui fournissent ce

langage commun. Ces normes, dont SGML est la principale [Bur94, Ass91], fournissent une structure d'accueil pour les documents qui deviennent alors reconnaissables par des systèmes hétérogènes. L'organisation des données qui constituent le document se fait alors selon des règles établies via la norme et le document peut aisément être transmis entre des systèmes reconnaissant la norme. Les normes autorisent des échanges et garantissent une organisation homogène des données.

SGML est devenu un standard d'échange par sa capacité à fournir des éléments pour organiser les documents et à fournir un langage de marqueur désormais compréhensible par la plupart des systèmes. SGML part du postulat que tout document électronique admet une *structure logique* qui reflète l'organisation des données du document. Les documents sont donc classés selon le type de leur structure logique: lettre, livre, facture, journal, etc, et ce type est décrit dans une DTD (Document Type Definition) à laquelle le document doit se conformer. Dans cette DTD, les types d'éléments de structure d'un document sont spécifiés: dans un journal il y a un titre, des articles, etc, ainsi que les compositions et enchaînements possibles entre ces types d'éléments: un journal est composé d'un titre suivi d'un article. La norme SGML s'occupe de l'organisation logique des parties de document et donne au système la responsabilité de présenter le document, c'est-à-dire de lui donner une *structure physique*. Il s'agit là d'une séparation explicite des données et de leur mode de présentation. Cette séparation facilite les échanges puisqu'il suffit de transmettre un document avec sa DTD que le système va alors pouvoir analyser et il va construire une présentation selon ses propres capacités. L'élaboration de DSSSL (Document Style Semantics and Specification Language) [Pre96], défini pour fournir une sémantique à la présentation des entités logiques, vient de la nécessité de bien délimiter les deux structures. DSSSL peut être présenté comme un complément de SGML pour établir des règles de présentation communes à des documents.

Comme nous l'avons souligné dans l'introduction, l'évolution dans la production du document est sensible, toutefois un aspect complémentaire à cette évolution reste trop délaissé: *les modalités d'accès au document électronique*. En effet, il est "facile" de construire un document électronique en suivant des règles pour organiser les données mais est-il aussi facile de retrouver de l'information dans ce type de document ?

C'est ce point crucial, tant pour la réutilisation de l'information que pour son obtention, que nous allons étudier ici à travers différentes mises en oeuvre informatiques reposant sur des modélisations de documents complexes et des stratégies de recherche adaptées à ces documents.

### 2.1.1 Modes de recherche

Les modes de recherche et donc d'accès aux informations stockées dans des bases se classifient habituellement en fonction des types des systèmes. Ainsi, nous pouvons distinguer les systèmes de gestion de bases de données (SGBD), les systèmes de recherche d'information (SRI) et les systèmes hypertextuels par les modes d'accès qu'ils proposent. De manière naïve, il est facile de considérer que chacun de ces systèmes propose un mode d'accès à l'information qui lui est particulier. Nous nous en tenons ici à des systèmes traditionnels et nous reviendrons par la suite sur la tendance actuelle qui tend à combiner et à intégrer les fonctionnalités de chacun [ABV95, BMN94, Bru93].

*Les systèmes de gestion de bases de données*

A partir d'un SGBD traditionnel, un utilisateur a la possibilité de consulter les données stockées par ce système en utilisant un langage de requête déclaratif (SQL [ISO89, Bla95] pour le modèle relationnel, ou OQL [ODM93] pour le modèle à objet).

Le principal inconvénient lié à ce mode de recherche provient de la connaissance nécessaire de la base et du schéma sous-jacent. En effet, les requêtes de type SQL sont dirigées par la structure du schéma de la base de données. Elles présentent l'avantage d'être simples et rapides à écrire mais ceci peut se transformer en inconvénient lorsque le schéma de la base n'est pas connu par l'utilisateur. Les requêtes requièrent donc une connaissance exacte de la structure de la base.

L'autre avantage connu et reconnu de ce type de requête repose sur la notion de correspondance exacte entre les valeurs des attributs spécifiées dans la requête et celles stockées dans la base. Il est donc toujours possible de déterminer, pour une base donnée, si une requête admet ou non des réponses.

Le SGBD traditionnel offre donc une consultation que nous pouvons qualifier de *rigide* puisque reposant sur une correspondance exacte et une connaissance de la structure du schéma.

*Les systèmes de recherche d'informations*

Un SRI, ou système de recherche d'informations, est un système qui stocke, gère et manipule un ensemble de documents, de façon à permettre de retrouver ceux dont le contenu correspond le mieux à leur besoin d'information [Nie90].

La notion de documents englobe toute forme d'entité à laquelle il est possible d'attacher une description obtenue par un processus d'indexation. Cette description correspond à la représentation du document selon un modèle qui, contrairement au schéma d'une base de donnée, doit rester indépendant du langage de requête. Ainsi les SRI proposent un langage de requête tendant à se rapprocher de la langue naturelle et utilisant un mode de correspondance basé sur une évaluation de la similarité entre un document et la requête.

Cette correspondance se veut donc graduelle et offre en résultat un ensemble de documents qui sont considérés par le SRI comme pertinents pour la requête puisque contenant dans leur description les éléments de la requête. Il s'agit ici de la *pertinence système*. C'est ensuite à l'utilisateur d'évaluer dans quelle mesure les documents qu'il obtient en réponse correspondent à ses besoins, c'est-à-dire dans quelle mesure la pertinence du système se rapproche de sa propre pertinence, la *pertinence utilisateur*. Le SRI traditionnel offre donc une consultation *souple* qui autorise une connaissance inexacte du fonds documentaire et qui évalue la distance entre la requête et les documents.

*Les systèmes hypertextuels*

Enfin, les systèmes hypertextuels proposent une consultation qui n'est plus basée sur la formulation d'une requête qui retourne un ensemble de données ou de documents, mais ils proposent de parcourir les informations contenues dans la base par *navigation*. La navigation requiert la définition de relations au sein des documents pour consulter les documents. Ces relations peuvent reposer sur des choix ad-hoc du créateur du document ou bien sur fondements sémantiques [ACG91, AMC95]. Ainsi l'introduction de la notion "d'hyperindex" permet de

diriger les relations hypertextuelles par des fondements sémantiques : des relations sont créées entre noeuds de l'hypertexte lorsque la distance sémantique établie entre ces noeuds révèle une forte similarité. Il s'agit d'aider à la construction d'un hypertexte [WvdBH96] mais aussi de fournir une sémantique, trop souvent délaissée, aux relations hypertextuelles.

Le concept de documents hypertextuels et de lecture hypertextuelle est une matérialisation du raisonnement humain qui fonctionne par association. Cette idée de représenter des données liées entre elles similairement à des associations d'idées du raisonnement humain a été introduite par Vannevar Bush en 1945 [Bus45]. Les systèmes hypertextuels sont aujourd'hui bien connus et fréquemment utilisés, la consultation du réseau internet s'effectue à l'aide d'applications hypertextuelles (Internet Explorer, Netscape, ...) et de nombreux cd-rom proposent une consultation de leurs données sur ce mode. Toutefois, ces systèmes comportent encore des failles et doivent affronter plusieurs problèmes relatifs à leur organisation ou a leur mode de consultation. Ammar Kheirbek [Khe95] décrit trois grands problèmes : les problèmes de navigation, de l'organisation des informations et de la modélisation. Ces problèmes soulignent que malgré l'attractivité de ce mode de consultation est attractif, il n'est pas sans défaut.

### 2.1.2 Fonctionnalités BD et RI

La recherche de *données* par leur structure est la technique la plus courante dans les systèmes de gestion de bases de données. En effet, l'interrogation des SGBD relationnels et objets reposent chacun sur des langages de requêtes déclaratifs SQL [ISO89] et OQL [ODM93] respectivement qui nécessitent une bonne connaissance de la structuration interne des données, c'est-à-dire une bonne connaissance du schéma de la base.

En effet pour formuler une requête dans ces langages, il faut connaître le nom de chacun des attributs concernés et le nom des tables dans le modèle relationnel et des classes dans le modèle à objets. Il faut aussi connaître le domaine de ces attributs pour pouvoir consulter les données gérées par ces systèmes. L'utilisation de ces langages nécessitent donc une expertise de la part de l'utilisateur, et cela même si le schéma définissant le nom des attributs et des tables est accessible.

Par la suite, nous revenons sur les fonctionnalités de recherche que nous estimons nécessaires dans un système documentaire. Nous établissons la correspondance avec les systèmes traitant exclusivement des données en utilisant les travaux sur ce sujet établis par Van Rijsbergen [Rij79] et Blair [Bla90]. Nous nous intéressons ensuite aux différentes formes de recherche qui peuvent permettre de fournir des accès à des documents structurés. Nous en tirons alors des conclusions sur l'orientation de nos travaux.

Nous revenons un instant sur ce qui va caractériser la recherche documentaire vis-à-vis de la recherche de données. La distinction entre ces deux types de recherche n'est pas évidente au premier abord. Considérons donc qu'une base contienne les informations suivantes : une personne de nom *Eric* qui a *25 ans*. Cette donnée, l'âge, se caractérise par le fait qu'elle présente un et un seul point d'accès. En effet, pour retrouver l'âge de cette personne, il faut connaître le nom de cette donnée, *âge de Eric*, car il s'agit de l'unique moyen pour accéder à cette information. Les documents présentent habituellement de nombreux points d'accès qui permettent de les retrouver par des combinaisons de descripteurs tels que l'auteur, la date, le titre, le contenu, etc.

Van Rijsbergen [Rij79] et Blair [Bla90] différencient la recherche de données de la recherche documentaire par les propriétés que nous reprenons dans la figure 2.1.

| | Recherche de Données | Recherche documentaire |
|---|---|---|
| Correspondance | Exacte | Partielle |
| Modèle | Déterministe | Non Déterministe |
| Langage de requête | Déclaratif | Naturel |
| Exactitude des réponses | Correspondance | Pertinence |

**Figure 2.1.** Propriétés de la recherche de données et de la recherche documentaire

Examinons chacun de ces points, avant de voir les différentes approches proposant des accès à des documents structurés. En recherche de données, les éléments retrouvés doivent correspondre exactement à ce qui est recherché. En recherche documentaire, le résultat est composé d'un ensemble de documents qui répondent et les "meilleurs" documents sont restitués en premier lieu. Le modèle sur lequel repose la recherche de données est déterministe, c'est-à-dire qu'il est nécessaire d'avoir une relation entre une requête bien construite et la réponse correcte à cette requête. En recherche documentaire, cette relation existe avec un certain degré de certitude (ou d'incertitude). Les langages de requête des systèmes de recherche de données utilisent généralement un vocabulaire limité et une syntaxe restreinte qui donne un aspect artificiel aux requêtes. La recherche documentaire s'appuie quant à elle plus généralement sur une formulation plus naturelle des requêtes et plus proche de la langue naturelle. De plus les requêtes de ces systèmes sont incomplètes alors que les requêtes des systèmes de recherche de données doivent préciser complètement ce qui est demandé.

Le dernier point qui différencie les deux types de systèmes concerne l'exactitude des réponses fournies. La recherche de données connaît deux types de réponses : la base interrogée contient une donnée correspondant exactement à la requête ou bien il n'existe pas de réponse à la requête. L'évaluation est donc objective. Les systèmes de recherche documentaire proposent une évaluation subjective se rapportant à l'utilité de la réponse : "est-ce que le système a satisfait votre besoin d'information ?". Il s'agit ici d'évaluer le critère de pertinence des réponses.

L'évolution des systèmes ainsi que l'introduction de nouveaux types de données dans ces systèmes a considérablement rapproché SGBD et SRI. Ainsi les SGBD "multimédia", c'est-à-dire capable de stocker et de manipuler des données non factuelles intègrent des requêtes sur ces nouvelles données avec une correspondance reposant sur la notion de pertinence [WMG+93, Ozk95, KSP+95, WNM+95, Nar95]. L'interrogation des données textuelles [Sav90, Loe94], des images fixes [CDDA88, NF93, PPS94, OS95] ou des séquences vidéo [ADHC94, ABL95, WDG95] requiert en effet un mode de correspondance plus souple que ceux proposés dans les systèmes de recherche de données. Le rapprochement entre les deux types de systèmes s'établit par la nécessité de faire face à de nouvelles données et à de nouveaux types de documents.

Aujourd'hui les systèmes ne cherchent donc plus seulement à manipuler des données mais ils s'orientent vers la gestion des *documents complexes*, que nous appellerons indifféremment documents structurés ou documents électroniques. Le document reste un assemblage de don-

nées liées par des relations d'agrégation et d'enchaînement. Les données prennent des formes aussi diverses que du texte, des images fixes, des images animées, des graphiques, des séquences sonores ou vidéo [AK92, BGMS93, BR94, AC95, Gha95, RNL95].

Les systèmes gérant des documents structurés peuvent être couverts par le terme très général de librairies digitales (digital library) dont l'intérêt actuel se mesure à la vue de l'émergence dans la communauté scientifique de nouvelles conférences spécifiques ou de nouvelles publications (International Journal on Digital Libraries édité par Springer). Ces systèmes abordent nécessairement tous les problèmes liés à l'acquisition, la gestion, l'indexation, la classification, la recherche, la sécurité et la présentation des informations digitales. Nous nous intéressons pour notre part principalement à une part non négligeable de ces systèmes : la représentation et la recherche des documents.

### 2.1.3 Synthèse

Nous cherchons à établir un mode d'accès à des documents complexes en utilisant les relations de structure qui lient les composants de ces documents. Pour cela, nous allons dans un premier temps considérer comment sont traités les documents non structurés, c'est-à-dire les approches traditionnellement rencontrées. Puis nous étudions les modèles de représentation qui ont été proposés pour des documents textuels structurés ainsi que les fonctionnalités introduites dans les langages de requêtes de ces systèmes. Nous considérons ensuite l'image fixe comme un document à multiples facettes et nous nous intéressons aux systèmes et modèles qui les représentent ainsi. Cette étude nous permet de mettre en évidence les caractéristiques propres aux données textuelles et aux images fixes. Enfin, nous voyons comment les documents combinant ces deux types de données sont traités.

Nous allons ainsi dégager les grandes lignes de notre travail, c'est-à-dire que nous cherchons à répondre aux questions suivantes : "quels sont les éléments indispensables dans la représentation des documents structurés ?", mais aussi, "quelles sont les lacunes des systèmes actuels auxquelles nous sommes confrontés ?".

## 2.2 Les documents non structurés

L'approche consistant à considérer le document comme une entité indivisible est la plus courante dans les systèmes traditionnels car moins complexe et donc plus facile à mettre en oeuvre. C'est la nature du document qui fournit le type de l'élément de base: chaînes de caractères pour le document textuel et pixels pour les images fixes. Par nature, ces documents sont monomédias car indivisibles.

Nous allons donner une rapide description de ces approches dorénavant bien connues en séparant les deux médias qui nous intéressent: textes et images. Nous cherchons aussi à différencier dans cette description les approches provenant des bases de données et celles issues des systèmes de recherche d'informations. Dans cette description, nous nous intéressons particulièrement aux modes de représentation et aux modes d'interrogation qui sont proposées sur ces types de documents.

### 2.2.1 Les documents textuels

Partons de l'exemple du SGBD à objets $O_2$. Dans ce système, le document textuel est vu sous la forme d'une liste de chaînes de caractères dont chaque chaîne correspond à une ligne du document (classe Text dans le $O_2$Kit [tec95]). Il s'agit donc d'une représentation "simpliste" et peu exploitable. La plupart des autres SGBD relationnels (Oracle, Informix, etc) se contentent de ce type d'approche en permettant de retrouver une chaîne de caractères incluse dans les documents textuels qu'ils stockent. L'interrogation des documents textuels sur les SGBD se caractérise principalement par l'emploi d'opérateurs de recherche "plein texte".

Ces opérateurs permettent de spécifier une chaîne de caractères en autorisant le plus souvent l'emploi de "jokers" tels que les symbole "%", (LIKE "canto%": toutes les chaînes commençant par "canto"), ou "?", (LIKE "canto?a": toutes les chaînes de la forme "canto?a" où "?" est une lettre quelconque). L'emploi de ces opérateurs se révèle difficile car leur puissance d'expression reste limitée, surtout sur de longs textes.

Afin de fournir des langages de requêtes contenant des opérateurs plus expressifs, il est nécessaire de modéliser le texte et d'exprimer au moins la structure des symboles qui le composent: les symboles de base sont les caractères, ils s'agrègent pour former des chaînes et sont mis en séquence, etc. A partir de ces modèles [Loe94], il est alors possible d'établir des algèbres pour manipuler et interroger les documents textuels en proposant des opérateurs de recherche "plein texte" plus évolués: prise en compte de la distance lexicale, spécification des positions de chaînes, etc. Cependant, il s'agit là de travaux tenant compte de la structure du document.

Les systèmes de recherche d'informations cherchent en premier lieu à donner une représentation synthétique des documents textuels. Cette représentation, appelée *index*, peut prendre différentes formes, du simple mot-clé [Rij79] à des structures sémantiques complexes [Ber88]. L'obtention de cette représentation passe par une phase d'analyse des documents appelée *indexation*.

L'indexation classique d'un document textuel consiste à sélectionner les termes du document qui n'apparaissent pas dans un anti-dictionnaire ("le", "la", etc). Par la suite un processus de lemmatisation est appliqué sur les termes sélectionnés afin de ne conserver que les racines de ces termes. Il s'agit le plus souvent d'enlever le suffixe et/ou le préfixe. Enfin, une mesure de représentativité des racines des termes est calculée à partir de leur nombre d'apparition dans le document. Salton, dans [Sal71], note $FreLoc(t, d)$ le nombre d'occurrences du terme $t$ dans le document $d$ et il note $FreCor(t, \mathcal{D})$ fréquence du terme $t$ dans le corpus de documents $\mathcal{D}$. A partir de ces deux fréquences d'apparition, il est classique de calculer une représentativité du terme dans le document (ou pondération), notée $w(t, d)$, à partir de la formule *"tf.idf"*, où *tf* signifie "term frequency" et *idf* signifie "inverse document frequency".

$$w(t,d) = FreLoc(t,d).log_2\left(\frac{|\mathcal{D}|}{FreCor(t,\mathcal{D})}\right)$$

Le poids d'un terme d'indexation est proportionnel à la fréquence d'occurrence interne dans le document (*tf*), autrement dit à sa représentativité dans le document, et il est inversement proportionnel à sa fréquence globale dans le corpus (*idf*). Ceci permet de déterminer

son aptitude à discriminer le document dans le corpus.

Ce processus d'indexation connaît de nombreuses variantes dans les différents SRI. Ces variations dépendent du modèle de recherche mais aussi du domaine d'application. L'approche que nous avons présentée est toutefois assez générale et présente l'avantage d'être simple et efficace. Son inconvénient majeur réside dans le fait qu'il n'existe pas de relations entre les termes d'indexation et que le contexte de chaque terme est donc perdu.

La phase d'interrogation pour ce type d'indexation, et plus généralement pour les SRI appliqués à des documents textuels est proche du langage naturel. Un processus de lemmatisation est appliqué à la requête dont le résultat est mis en correspondance avec les index des documents selon le modèle de recherche (booléen, vectoriel, etc).

Cette approche est actuellement plus fréquente dans la communauté de recherche d'informations. La qualité des réponses fournies est souvent suffisante dans des systèmes généralistes mais lorsque les documents sont plus spécifiques ou bien lorsque chaque document aborde de nombreux thèmes différents, cette approche connaît ses limites. Nous pensons qu'il est alors préférable de modéliser plus finement le document et son contenu. Ceci passe par la prise en compte de la structure et par l'usage de vocabulaires contrôlés et/ou de formalismes de représentation de connaissances en lieu et place des mots-clés.

### 2.2.2 Les images fixes

Les images fixes sont aujourd'hui traitées comme des données classiques dans les SGBD modernes. Il est ainsi permis de les stocker et de les manipuler aisément dans un SGBD. Dans les systèmes généralistes, Oracle, Informix, $O_2$, etc, leur interrogation reste très limitée puisqu'il n'existe pas de méthodes permettant d'extraire automatiquement des informations sur ces données ni de méthodes pour les comparer efficacement entre elles. Des attributs externes classiques peuvent donc décrire l'image et ce sont ces attributs qui sont consultés pour rechercher une image.

Nous voyons, au travers du système Chabot, une première étape vers un processus d'indexation automatique des images fixes. Nous pouvons aussi citer le récapitulatif sur ce type de travaux faits par Faloutsos [Fal95].

*Le système Chabot*

Chabot est un système de recherche d'images pour une base de données développée en collaboration avec le DWR (Department of Water Resources) qui a fourni une collection d'images composées de plus de 500 000 unités. L'objectif de ce projet est de proposer des solutions de stockage et de recherche en ligne des images en offrant une recherche simple, flexible et basée sur le contenu des images.

Le SGBD relationnel utilisé est POSTGRES qui autorise des types complexes et l'intégration de nouvelles fonctions pour le traitement des images. Les images stockées dans cette base sont d'une part décrites par des attributs formatés (date, lieu, etc) et d'autre part par des attributs plus complexes basés sur l'histogramme des couleurs ou des concepts. Nous donnons des exemples de requêtes possibles avec ce système.

**Requête 1**
**retrieve** *(q.all)* **from** *q* **in** *PHOTOCD_BIB*

**where** *q.shoot_date > 1/1/94 and MeetsCriteria("MostlyRed",q.histogram)*

A travers cette requête, nous recherchons une image prise après le 1/1/94 et ayant comme couleur dominante le rouge. C'est par l'appel de la fonction *MeetsCriteria* que ce dernier critère est spécifié. L'utilisateur n'est pas libre de choisir la couleur qu'il veut, il doit sélectionner une couleur parmi celles qui sont pré-définies ("Some Red", "Red", "MostlyRed", etc). Lors de la recherche, les images dont le nombre de pixels sont de cette couleur ou d'une couleur proche dépassent un seuil fixé sont restituées à l'utilisateur.

Par ailleurs, l'utilisateur peut intégrer de nouveaux concepts sur les ensembles d'images qui ont été retrouvées. Ces concepts sont par la suite réutilisables lors de requêtes suivantes. Il s'agit en quelque sorte de compléter l'indexation par des concepts élémentaires, par exemple la requête suivante spécifie que la description de l'image doit contenir le concept "sunset" :

**Requête 2**
**retrieve** *(q.all)* **from** *q* **in** *PHOTOCD_BIB*
**where** *q.shoot_date > 1/1/94* **and** *MeetsCriteria("MostlyRed",q.histogram)*
**and** *q.description* $\backsim$ *"sunset"*

Les tests menés sur une partie de la collection montrent que l'introduction des concepts dans la description des images est profitable. L'analyse des résultats de ce système montre que pour des requêtes utilisant le seul contenu physique des images (ici l'histogramme des couleurs) les images pertinentes sont toutes restituées mais le bruit est trop important (précision proche de 6%). Les meilleurs résultats sont obtenus avec la combinaison des concepts et des critères sur la couleur des images.

Il apparaît donc que l'usage des concepts dans la description des images agrémente singulièrement l'utilisabilité du système de recherche. Parmi les systèmes de recherche d'images, nous pouvons citer le travail de Harmandas & al. [HSD97] qui cherche à exploiter l'environnement textuel des images pour indexer puis interroger les images. Dans ce travail, des techniques classiques d'indexation des données textuelles, telles que celles présentées en section 2.2.1, sont appliquées. Puis, les données textuelles qui constituent l'environnement de chaque image sont déterminées et fournissent l'indexation de chacune de ces images. Différents choix d'environnement sont comparés afin de déterminer le meilleur possible dans le cadre de l'utilisation de cette approche pour indexer les images d'un site Web.

Nous pouvons noter que la plupart des approches actuelles d'indexation et de recherche d'images s'articulent autour de modèles de représentation des images décrivant l'image selon plusieurs facettes: facette physique, facette perceptive, facette symbolique, etc. Ceci démontre la complexité de ces données mais aussi leur forte structuration. Nous voyons par la suite des systèmes et modèles plus complètement (2.4).

## 2.3 Les documents textuels structurés

La structure des documents textuels peut être abordée selon différentes approches: la structure inhérente aux données textuelles et la structure fournie par la structure logique des documents. Les données textuelles peuvent être vues comme une arborescence dont les symboles de base sont les caractères qui en s'agrégeant forment des mots, puis des groupes,

etc, pour finalement constituer le document textuel complet. Nous allons principalement nous intéresser à la seconde approche qui définit la structure du document textuel à partir de sa structure logique.

Nous présentons dans un premier temps des approches reposant sur des SGBD ou s'apparentant à des SGBD textuels. Nous présentons ensuite l'approche originale de Navarro [NBY95b] qui fait cohabiter au sein d'un même modèle des vues multiples du document, c'est-à-dire qu'il représente plusieurs types de structure pour un même document. Ce travail est à la lisière des travaux sur les SGBD et des travaux provenant de systèmes de recherche d'informations. C'est par ces derniers que nous concluons cette section en décrivant l'usage qui est fait de la structure dans des SRI.

### 2.3.1 Le SGBD : structure d'accueil de documents textuels structurés

Les SGBD traditionnels, qu'ils soient relationnels (Oracle, Informix, etc) ou à objets ($O_2$, ObjectStore, etc), offrent la possibilité de stocker et manipuler des données textuelles. Cependant, ils ne proposent pas de mécanisme pour intégrer automatiquement la structure de ces données.

Nous considérons ici les travaux qui proposent d'une part une structure d'accueil pour des documents textuels structurés et d'autre part des extensions des langages de requêtes traditionnels (SQL, OQL) pour interroger ce type de documents.

#### a) Représentation des documents textuels structurés sur un SGBD

Nous présentons deux types de SGBD accueillant des documents structurés : la première structure d'accueil est établie sur un SGBD relationnel [BCK+94] alors que la seconde est établie sur un SGBD à objets [Chr96]. Nous distinguons donc les deux types traditionnels de SGBD, ceux basés sur le modèle relationnel [Mac81, Cra81, SSL+83, Bla98, BCK+94] et ceux basés sur le modèle à objets [YA94, CPC94, Zha95, VBA95, FFE96, VAB96, Chr96, ACC+97].

*Accueil sur un SGBD relationnel*

La principale lacune des SGBD relationnels pour la gestion et la recherche de documents structurés réside dans leur manque de souplesse pour définir des types abstraits. Il n'est pas facile de rendre fidèlement la structure d'un document à travers des relations, et lorsque cela est rendu possible [BCK+94], les requêtes SQL sont rendues complexes de même que le schéma de la base. En effet le passage d'une hiérarchie décrivant la structure du document à un ensemble de relations la représentant est un processus assez lourd dont le résultat d'une des implantations possibles est la suivante :

> TEXT_NODES(nodeid, genid, content) : un triplet correspond à un noeud de l'arborescence structurelle et il est décrit par un identifiant unique (nodeid), un type de noeud (genid) et le contenu de ce noeud.
> TEXT_ATTRIBUTES(nodeid,attr,value) : un triplet correspond à la valuation d'un attribut SGML d'un élément de structure. Pour un noeud de l'arborescence structurelle (nodeid), l'attribut identifié par attr prend la valeur value.
> TEXT_STRUCTURE(a_nodeid,d_nodeid) : un doublet correspond à la relation de composition. Le noeud identifié par a_nodeid de l'arborescence structurelle est le père du noeud identifié par d_nodeid.

Cette représentation des documents structurés, c'est-à-dire des éléments structurels, de la relation de composition et des attributs des éléments structurels, présente l'avantage d'autoriser la consultation de tels documents avec le langage SQL associé à des opérateurs de comparaison de chaînes de caractères. Toutefois la lourdeur de la modélisation ainsi que la complexité des relations qui en résultent dégradent singulièrement les performances des systèmes.

D'autres exemples de systèmes documentaires implantés sur des SGBD relationnels [Mac81, Cra81, SSL+83, Bla98] montrent la difficulté de cette tâche en raison de l'inadéquation du modèle relationnel pour représenter des documents structurés. La plupart de ces travaux s'orientent donc vers les modèles objets [YA94, Zha95, VBA95, FFE96, VAB96, Chr96, ACC+97]. Ces modèles présentent l'avantage de s'apparenter fortement aux modèles de documents structurés par certaines de leurs caractéristiques :

- les modèles objets fournissent des constructeurs de type que ne possèdent pas les modèles relationnels.
- les modèles objets intègrent des propriétés telles que l'agrégation qui existent aussi dans les documents structurés.
- les modèles objets proposent la définition de types atomiques d'une plus grande diversité que les modèles relationnels.

Il est donc préférable d'accueillir des documents structurés sur un système à objet [CPC94]. Si les avantages de ces système sont indéniables vis à vis des systèmes relationnels, les fonctionnalités de recherche documentaire restent à définir et à implanter. S'il reste encore beaucoup à faire pour accueillir de manière générique des documents structurés sur un SGBD, le couplage SGBD-SRI que nous présentions en introduction prend ici toute sa valeur quand nous évoquons les fonctionnalités de recherche.

En effet, actuellement ces fonctionnalités se limitent à des recherches de données qui proviennent des SGBD-OO traditionnels et donc des langages de requêtes déclaratifs qui leurs sont associés tels que OQL (Object Query Language) [ODM93]. Pour des systèmes documentaires, il est par exemple nécessaire de pouvoir "naviguer" dans les documents, de pouvoir consulter de manière synthétique leur contenu de même qu'il est souhaitable de fournir des accès directs aux documents répondant à un besoin particulier de l'utilisateur. La mise en place de ce dernier type d'accès nécessite l'implantation de fonctionnalités de recherche que les SGBD-OO traditionnels ne possèdent pas et qui relèvent de la recherche d'informations. Nous tenons à préciser que nous ne tenons pas à opposer deux types de systèmes, SGBD et SRI, ayant leurs propriétés propres mais nous voulons montrer leurs aspects complémentaires dans le cadre de fonds documentaires constitués de documents structurés.

*Accueil sur un SGBD à objets*

Le langage POQL [Chr96] proposé en aval d'un protocole de chargement de documents SGML dans une base de données à objet, en l'occurrence le système $O_2$, introduit quelques-unes de ces nouvelles fonctionnalités. Nous présentons cette approche et particulièrement la traduction d'un document SGML vers la base d'objets. Dans cette application, les documents se conforment à une DTD et le protocole de traduction a nécessité l'implantation de nouveaux

types permettant par exemple de représenter les alternatives dans la structure ("une illustration est composée d'une image ou d'un texte"). Pour représenter des documents SGML sur le SGBD $O_2$, il a fallu répondre à deux questions élémentaires : *"comment représenter une définition de type de document (DTD) sous la forme d'un schéma $O_2$?"* et *"comment traduire les balises SGML vers des objets et des valeurs dans une base du SGBD?"*.

La traduction d'une DTD vers un schéma $O_2$ s'établit à l'aide d'une grammaire non contextuelle annotée de programmes qui spécifient l'association entre les symboles de la grammaire et leur représentation dans la base. D'autre part, la traduction des balises SGML vers des éléments de la base peut se faire selon deux approches : une représentation *faiblement typée* dans laquelle chaque noeud de l'arbre syntaxique SGML est un objet dont un attribut indique le type du noeud et une représentation *fortement typée* dans laquelle une correspondance est établie entre la DTD et un schéma d'une base: à chaque type de noeud correspond une classe d'objets. Cette seconde approche, plus riche sémantiquement, a été adoptée par Christophides et elle nécessite une extension des primitives de modélisation : ajout des *unions de types* pour représenter l'alternative dans la structure et des *n-uplets ordonnés* pour représenter la notion d'ordre sur les éléments SGML.

En fournissant ces extensions sur les primitives de modélisation du SGBD $O_2$, il devient possible d'avoir un passage automatique d'un document SGML conforme à une DTD vers un ensemble d'objets appartenant à un schéma fidèle à la DTD. Chaque connecteur présent dans une DTD pour définir les types d'éléments SGML correspond à un constructeur de type du SGBD $O_2$ (voir figure 2.2). Par ailleurs, les attributs présents dans les documents SGML et attachés aux types d'éléments dans la DTD sont représentés par des attributs classiques dans les classes d'objets. Enfin, le contenu des éléments SGML prend la forme d'un attribut dont le type dépend du contenu SGML. Là encore une table de correspondance est établie.

| Connecteur SGML | Sens du connecteur | Constructeur de type $O_2$ | |
|---|---|---|---|
| a , b | a suivi de b | tuple ordonné | tuple(a : ClasseA, b : ClasseB)<br>constraint: a!= nil, b !=nil |
| a \| b | a ou b | union marquée | union(a : ClasseA, b : ClasseB)<br>constraint: a!= nil \| b !=nil |
| a* | occurrences de a | liste | tuple(a : list(ClasseA)) |
| a+ | occurrences de a, au moins une | liste | tuple(a : list(ClasseA))<br>constraint: a != list() |

**Figure 2.2.** Correspondance entre les connecteurs SGML et les constructeurs de type O2

Nous avons dit que les systèmes du monde objet semblent plus aptes à accueillir des documents structurés que les systèmes relationnels, toutefois nous constatons que ce travail de traduction de documents SGML vers un SGBD à objets n'est pas élémentaire. Si les approches diffèrent pour définir ce que doit être la structure d'accueil d'un document SGML sur un SGBD à objets, l'objectif est toujours de faciliter à terme l'accès à ces documents et donc leur interrogation. Pour cela, l'approche adoptée par Christophides basée sur une représentation fortement typée permet de conserver au niveau du schéma de la base la sémantique de la structure et facilite ainsi l'interrogation. De plus, il devient possible via cette approche de

séparer totalement la structure du document du contenu de ce document puisque seule la structure peut être chargée dans la base et le contenu peut être conservé sous la forme de fichiers.

Ce dernier point nous permet d'aborder le difficile problème du mode de stockage du document dans la base: faut-il conserver une entité document indivisible ou l'éclater en autant de composants structurels? Une discussion sur ce problème nécessite la prise en compte de l'usage qui est fait des documents. Comparot-Poussier et Chrisment discutent de ce problème dans [CPC94] en prenant en compte la réutilisation des parties des documents mais aussi de la gestion de ses versions. Dans un environnement complet de gestion de documents électroniques tout ses aspects doivent être pris en compte pour établir un choix. Dans notre cadre plus particulier, la recherche sur de tels documents, il s'agit surtout de savoir si nous autorisons ou non des vues multiples d'un même document.

La définition de vues multiples d'un même document signifie qu'il devient possible de voir le document selon les différentes sortes de structures qui peuvent lui être attachées : structure logique, structure physique, structure de présentation, etc. Nous voyons un travail mené par Navarro sur le thème particulier de l'interrogation des documents structurés textuels admettant différentes structures.

### b) Les fonctionnalités des langages de requêtes

Nous cherchons ici à donner un aperçu des fonctionnalités proposées dans les langages de requêtes qui s'adressent spécifiquement aux documents structurés. Dans cette revue des fonctionnalités, nous décrivons des fonctionnalités provenant principalement de systèmes gérant des documents textuels, mais aussi de systèmes capables d'accueillir des images fixes, par exemple le système MULTOS [Tha90], l'un des précurseurs en la matière. Cependant, les fonctionnalités présentées prennent leurs origines dans la définition du document textuel structuré même si elles peuvent s'appliquer plus généralement à tout type de document structuré.

Nous organisons cette description selon trois types de requêtes: les requêtes purement structurelles, les requêtes sur le contenu et enfin les requêtes sur les descripteurs structurels, autrement dit les requêtes sur les attributs décrivant les composants structurels du document. Les requêtes structurelles et les requêtes sur les descripteurs peuvent être généralisées à des documents multimédias.

*Les requêtes purement structurelles*

Le système MAESTRO [Mac90, Mac91] conçu pour être un environnement complet de gestion et de manipulation de documents structurés propose un langage de requête qui intègre des fonctionnalités de navigation dans la structure logique et d'extraction de composants logiques. Le résultat d'une requête prend la forme d'un *chemin d'accès* à un composant, laquelle est typée par la catégorie du composant auquel aboutit le chemin d'accès. Les résultats sont des pointeurs sur les fragments de documents retrouvés, ils sont nommés *références*. Il est ensuite possible d'effectuer des opérations ensemblistes sur ces résultats lorsqu'ils aboutissent à un composant de la même catégorie.

Le langage de requête de MAESTRO propose d'extraire n'importe quel composant d'après la catégorie du composant, par exemple un titre ou un chapitre. Le résultat est un ensemble de *chemins d'accès* de la forme suivante :

Groupe Document Elément [Indice] [Elément [Indice]]

Le Groupe correspond à une classe de document et le Document est un identificateur de document dans la classe de document. Pour chaque classe de document, un type de structure est associé conformément à une grammaire adoptant une syntaxe proche de SGML. Un Elément est un type de composant et l'indice indique la position dans une liste de composants répétés, 0 s'il n'y a pas de composants répétés. Considérons la requête suivante :

**Requête 1**
*List* **gets** *Section* **of** *Chapter*

Pour cette requête, le résultat est un ensemble de pointeurs sur les sections qui sont des composantes d'un chapitre. Cet exemple illustre la possibilité d'accéder à un composant par son nom. Il est aussi possible d'extraire les types de composant apparaissant à plusieurs niveaux dans la hiérarchie structurelle, par exemple en formulant la requête suivante :

**Requête 2**
*List2* **gets** *Title* **of** *Chapter*

Le résultat List2 est alors composé de tous les titres qui apparaissent dans un chapitre, c'est-à-dire les titres de chapitre, les titres de section, les titres de sous-section, etc. Puisque le chemin exact d'accès n'a pas été déclaré explicitement, par exemple en exprimant qu'on ne souhaite que des titres de section, le moteur de recherche consulte tout les descendants des chapitres pour retrouver tout les composants qui sont des titres.

Ce langage de requête propose ainsi un assortiment d'opérateurs permettant de manipuler tant la composition des documents structurés que la position des composants (opérateurs last, first). Des conditions peuvent aussi être spécifiées par des quantificateurs tels que all, any ou notany. Dans l'exemple qui suit, nous cherchons les sections des chapitres composées d'au moins deux sous-sections.

**Requête 3**
*List2* **gets** *Section (***having any** *2 SubSection)* **of** *Chapter*

Ce langage de requête propose un éventail assez complet d'opérateurs structurels permettant d'extraire des composants structurels avec une connaissance plus ou moins précise de la structure et de spécifier des conditions sur les caractéristiques structurelles de ces composants. Ces types de requêtes s'identifient par la notion de chemin d'accès.

Il faut toutefois noter que le système MAESTRO n'est pas le premier à avoir proposé la notion de chemin d'accès. Le système MULTOS (MULTimedia Office Server) et son langage d'interrogation [BRG88, BR90] intègre lui aussi cette notion en autorisant la navigation au sein de la hiérarchie structurelle des documents.

Le langage de requête de MULTOS propose ainsi deux types de composition : la composition directe notée "." et la composition indirecte notée "*". Réécrivons la requête 1 avec le langage de MULTOS :

**Requête 4**
**FIND** *Document*
**SCOPE** *Book*
**WHERE** *.Chapter*Title*

Dans cette requête, le ".Chapter" signifie que le chapitre est un descendant direct du document de type Book alors que le "*Title" signifie que le titre doit être un descendant du chapitre, sans être obligatoirement un descendant direct. Le système MULTOS offre donc des requêtes aussi puissantes que celles de MAESTRO sauf sur un point : le résultat des requêtes est toujours un document. Ceci signifie que nous ne récupérons pas uniquement les titres comme le propose le système MAESTRO, mais les documents qui contiennent ces titres. MULTOS ne restitue pas des parties de document mais le document complet alors que MAESTRO fournit des pointeurs sur les parties de document qui répondent correctement. Nous revenons par la suite sur la nécessité d'accéder directement à la partie "pertinente" dans le cadre de documents complexes.

Le langage de requête POQL (Path Object Query Language) [Chr96] reprend la notion de chemins d'accès et la généralise comme une extension du langage de requête du SGBD $O_2$. Il introduit aussi la notion de variable de chemins. Considérons toujours le même exemple de requête mais maintenant avec le langage POQL :

**Requête 5**
**select** *t*
**from** *Books PATH_p.title(t)*

Les valuations de la variable "PATH_p" sont tous les chemins dans un livre qui aboutissent à un attribut de nom title, autrement dit les chemins qui aboutissent à un élément composé d'un titre. La variable t qui sera donnée en résultat sera donc instanciée par le contenu des attributs title correspondant et cela quelle que soit la profondeur. En fait, le traitement de cette requête retourne un ensemble de n-uplets contenant un attribut par variable. Ici, nous avons donc un ensemble de couples contenant deux attributs : t et PATH_p. La valeur de l'attribut PATH_p correspond au chemin d'accès complet qui doit être traversé pour atteindre un attribut de nom title et la valeur de l'attribut t contient la valeur de l'attribut title atteint par le PATH_p correspondant.

Le fait de considérer une variable chemin d'accès autorise de nouveaux types de requêtes structurelles permettant de comparer le contenu de ces variables. Cela n'était pas possible avec les systèmes MAESTRO et MULTOS. Considérons par exemple que nous cherchons les différences structurelles entre deux livres. Cette recherche de différence va alors porter sur la comparaison des deux variables de chemin PATH_p1 et PATH_p2 des deux livres Book1 et Book2 et s'exprime simplement de la manière suivante :

**Requête 6**
*(***select** *PATH_p1*
**from** *Book1 PATH_p1)*
-
*(***select** *PATH_p2*
**from** *Book2 PATH_p2)*

Les variables de chemin PATH_p1 et PATH_p2 vont être valuées par l'ensemble des chemins dans les structures respectives des livres Book1 et Book2. Ainsi le résultat de cette différence ensembliste est l'ensemble des chemins du livre Book1 qui ne sont pas définis dans Book2.

L'introduction des chemins d'accès favorise l'interrogation structurelle des documents en autorisant une connaissance incomplète de la structure du document. Pour l'exemple que nous

avons pris, il n'est pas utile de connaître la structure exacte du document pour atteindre l'ensemble des composants de nom title. Cette approche présente donc des avantages indéniables pour l'interrogation des documents structurés et plus généralement pour l'interrogation des objets complexes. En effet ce type de langage démontre qu'il peut apporter des solutions pour l'interrogation des documents hypertextuels [CR94]. Cependant, Christophides avoue que la puissance d'expression et la complexité des langages avec des chemins d'expressions généralisés sont encore trop peu étudiées pour pouvoir conclure sur leur réelle utilisation. L'évaluation d'une requête explose de manière exponentielle en fonction du nombre d'attributs et de chemins par type impliqué dans la base ainsi que du nombre de variables de chemins de la requête. Ces critiques peuvent se formuler dans des applications particulières. La complexité augmente pour des requêtes complexes or l'usage actuel des applications de recherche montre que les requêtes ne comportent que très peu de termes, de un à trois dans la majorité des cas (cette constatation a été faite lors du SIGIR 1997, à Philadelphie, durant une table ronde sur l'usage des moteurs de recherche commerciaux). L'usage d'un langage tel que POQL doit donc s'avérer utile dans la plupart des cas.

Cette première caractéristique des langages de requête destinés aux documents structurés nous amène à parler de la forme prise par les résultats. Nous avons vu que les réponses du système MULTOS sont uniquement des documents complets et non des fragments de document. Le système MAESTRO fournit en résultat un ensemble de pointeurs sur les fragments du document qui répondent. Là encore, POQL présente l'avantage de retourner les objets qui répondent à la requête, c'est-à-dire les fragments du document, ainsi que la possibilité de présenter le chemin d'accès à cet objet au travers d'une variable de chemin. Les langages de type OQL présentent l'avantage de pouvoir construire des tuples en résultat d'une requête, ce qui permet de construire des vues du document structuré contenant uniquement les fragments qui intéressent l'utilisateur.

Nous allons maintenant aborder un point crucial auxquels sont confrontés les systèmes cherchant à interroger les documents structurés: la combinaison au sein des requêtes des aspects structurels et du contenu du document. Les trois langages de requêtes que nous avons vus jusqu'alors (MAESTRO, MULTOS et POQL) cherchent à adjoindre aux requêtes structurelles des fonctionnalités de recherche par le contenu afin de rendre leurs langages plus exploitables. Ces fonctionnalités restent le plus souvent des opérateurs de recherche textuelle qui s'apparentent aux recherches traditionnelles dans des champs textuels.

*Les requêtes sur le contenu*

Le système MAESTRO introduit l'opérateur in qui permet de vérifier qu'un terme se trouve dans un document ou dans un fragment de document. Il faut rappeler que le contenu des documents dans le système MAESTRO n'est pas indexé et qu'il est représenté par une liste de mots. Il s'agit donc d'un opérateur de "pattern matching". Dans l'exemple qui suit nous reprenons la requête 1 en ajoutant une condition sur l'apparition du terme "soft" dans le titre de la section. Cette requête restitue donc l'ensemble des sections d'un chapitre dont le titre de la section contient le mot "soft".

**Requête 7**
*List* **gets** *Section* **where** *("soft"* **in** *title)* **of** *Chapter*

Le système MULTOS propose une recherche sur le contenu assez similaire à celle de MAESTRO, du moins sur les documents textuels. Les prédicats sur les composants textuels

prennent la forme suivante :

component CONTAINS string_expr_list [distance]

Dans cette formulation, le composant noté component doit être un composant textuel et la liste d'expressions notée string_expr_list correspond à une liste de chaînes de caractères pouvant utiliser les notation du LIKE de SQL. L'expression de la distance spécifie la distance minimum permise entre les chaînes de caractères données dans la liste. Par rapport à MAESTRO, le système MULTOS introduit donc une plus grande flexibilité dans l'interrogation des portions textuelles.

Ce type d'interrogation sur les portions textuelles est assez classique dans les bases de données textuelles. En effet, des systèmes comme Textriever [Bur91], le modèle hybride de Navarro & Baeza-Yates [NBY95b, NBY96] ou les systèmes basés sur les PAT expressions [ST94] procurent ce type d'opérateurs. D'autres opérateurs sur les documents textuels existent et peuvent donc être incorporés dans les langages de requêtes. Pour une description plus complète de ces opérateurs, le lecteur pourra se reporter au travail de Loeffen qui compare plusieurs modèles de texte [Loe94] et présente les opérateurs de différents modèles dont les P-string [GT87], les PAT expressions [ST94] et le "containment model" [Bur92].

*Les requêtes sur les descripteurs*

Les systèmes MAESTRO et MULTOS autorisent la définition et l'interrogation sur les attributs rattachés à des composants structurels avec toutefois certaines limites pour chacun des deux systèmes. Ainsi le système MAESTRO permet la définition d'attributs uniquement sur les documents globaux ou bien sur les groupes de documents. De plus ces attributs sont purement textuels. Dans MULTOS, une confusion s'établit entre la notion de composant et la notion d'attribut. Ainsi, certains composants structurels des documents prennent la forme d'attributs et peuvent être interrogés avec des opérateurs spécifiques : $=$, $\neq$, $<$, $>$, $\leq$, $\geq$ ou Between sous la forme suivante

component rel_op constant

La requête suivante retournera l'ensemble des documents ayant un titre dont la numérotation est 5 en considérant que le composant Num existe en tant que descendant direct du composant Title.

**Requête 8**
**FIND** *Document*
**SCOPE** *Book*
**WHERE** *.Chapter*Title.Num = 5*

Le langage de requête POQL admet un comportement similaire à MULTOS puisque les attributs provenant de SGML sont transformés en attribut de l'objet représentant l'élément structurel de SGML. Toutefois ce langage introduit la notion de variable d'attribut qui permet d'assouplir l'interrogation en ne nécessitant pas une connaissance parfaite de la structure des documents.

Avec l'introduction des variables d'attribut, il n'est pas nécessaire de connaître le nom exact de l'attribut. Il faut déclarer une variable d'attribut et spécifier dans la requête la valeur qu'elle doit prendre. Nous donnons en exemple la requête 9 dans laquelle nous cherchons des

livres comportant un attribut instancié par la valeur “final”.

**Requête 9**
**select** *t*
**from** *Books{t}.#A(x)*
**where** *x= "final"*

Lors de la recherche, chaque attribut des livres de la racine Books sont alors des candidats potentiels. Cependant les attributs des livres ont des types différents. Pour résoudre ce problème, une union des types correspondants aux attributs d'un livre est inférée pour la variable x. Puis le domaine de cette variable est restreint conformément à la clause where de la requête. Cette restriction aboutit à une union dont les marqueurs sont des chaînes de caractères. Ainsi, la variable #A sera valuée uniquement avec les attributs de type correspondant, par exemple l'attribut status qui est de type chaîne de caractères.

Mis à part la souplesse introduite par cette notion de variable d'attribut, l'autre intérêt concerne la possibilité de comparer des attributs de même type sans avoir une connaissance explicite de ces attributs. Il est ainsi possible de vérifier si deux attributs de deux livres (Book1 et Book2) différents ont deux attributs n'ayant pas le même nom dans chaque livre mais admettant la même valeur :

**Requête 10**
**select** *struct(att:#A, att:#B, x)*
**from** *Book1.#A(x), Book2.#B(y)*
**where** *x= y*

Dans cette requête, les valeurs x et y doivent être identiques sans que les noms des attributs soient obligatoirement identiques.

La rigidité des langages descriptifs de type SQL ou OQL est ainsi remise en cause. Les recherches peuvent devenir plus génériques en éludant partiellement les noms des attributs puisque la seule spécification d'une valeur suffit pour écrire une requête. Ce type de requête nous rapproche considérablement des requêtes des systèmes de recherche d'informations qui autorisent la seule spécification d'un ensemble de mots clés en tant que requête.

### 2.3.2 Le document textuel structuré: une ou plusieurs structures?

Nous avons vu jusqu'à présent des systèmes qui intègrent la structure dans la représentation du document. Nous allons voir maintenant un modèle autorisant la définition de plusieurs structures pour un document ainsi que l'interrogation simultanée de ces différentes structures.

Nous allons nous intéresser ici aux caractéristiques du modèle de représentation des documents textuels structurés de Navarro & Baeza-Yates [NBY95b, Nav95, NBY95a, NBY96] autorisant des vues multiples du document, puis nous donnons les caractéristiques principales du langage de requêtes.

### a) Vues multiples d'un document textuel structuré

L'idée est de pouvoir proposer un texte sous différentes vues pour pouvoir interroger en combinant les critères de ces différentes vues. Il s'agit par exemple de retrouver *un texte en italique proche d'une figure et contenant le mot terre*. Cette requête intègre plusieurs aspects de la structure : la forme du texte avec l'italique, le composant logique avec la figure et le contenu du texte avec le terme terre. La difficulté réside donc dans la définition d'un modèle permettant d'avoir ces différentes vues du texte et de les rendre utilisables lors de l'interrogation. Une base de texte est alors modélisée par le septuplet suivant :

$$(\mathcal{T}, \mathcal{V}, C, N, R, Constr, Segm)$$

avec :

- $\mathcal{T} : [1..T] \rightarrow \Sigma$ est la table de texte où $T$ est la taille de la base en terme de nombre de symboles et $\Sigma$ est l'alphabet.
- $\mathcal{V}$ est l'ensemble fini de vues du texte.
- $C : \mathcal{V} \rightarrow 2^{\mathcal{C}}$ est l'ensemble des constructeurs de chaque vue du texte. $\mathcal{C}$ est l'ensemble fini des constructeurs.
- $N : \mathcal{V} \rightarrow 2^{N}$ est l'ensemble des noeuds de chaque vue du texte. $\mathcal{N}$ est l'ensemble fini des noeuds.
- $R : \mathcal{V} \rightarrow 2^{\mathcal{N} \times \mathcal{N}}$ est la relation binaire qui définit l'arborescence de chaque vue du texte.
- $Constr : \mathcal{N} \rightarrow \mathcal{C}$ donne le constructeur de chaque noeud.
- $Segm : \mathcal{N} \rightarrow [1..T] \times [1..T]$ détermine le segment associé à chaque noeud.

La notion de constructeur s'apparente à la notion de type de composant, dans une vue représentant la structure logique d'un livre les constructeurs sont Book, Chapter, Section, etc. Chaque constructeur est associé à un noeud de la structure qui réfère lui-même une partie du document, appelée ici segment de texte.

Le texte est vu comme une séquence de symboles appartenant à l'alphabet $\Sigma$. La granularité des symboles n'est pas précisée dans le modèle et ces symboles peuvent être les caractères, les mots, etc. La structure correspond à un ensemble de hiérarchies indépendantes modélisées par l'intermédiaire des différentes vues décrites dans l'ensemble $\mathcal{V}$. Avoir plusieurs vues signifie que les segments de texte associés aux noeuds des différentes hiérarchies peuvent se recouper alors que ce recoupement entre segments n'est pas autorisé au sein d'une même vue.

Une vue spéciale est décrite pour le "pattern matching", c'est-à-dire pour les opérateurs de comparaison textuelle. Cette vue, appelée *text view*, a une structure plate dont les noeuds réfèrent un ensemble de segments disjoints. Les constructeurs des noeuds de cette vue sont tous des constructeurs *text*. Cette vue se caractérise par le fait qu'elle comporte un noeud pour chaque segment possible du texte.

Le modèle défini par Navarro & Baeza-Yates [NBY95b, NBY96] est un des rares modèles qui s'attaque au problème des représentations multiples d'un même document à travers des

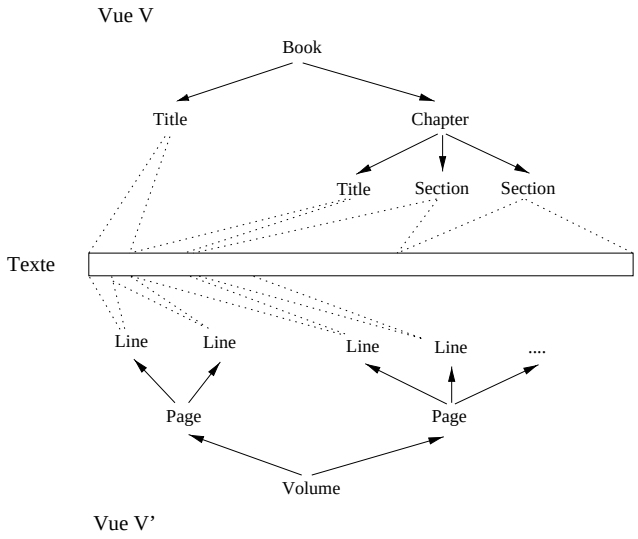



**Figure 2.3.** Modèle de texte à vues multiples défini dans [NBY95b]

vues référençant les mêmes segments de texte. La définition d'un tel modèle et la prise en compte lors de l'interrogation de noeuds provenant de vues différentes implique un traitement des segments inclus et des segments qui se recoupent pour résoudre les requêtes. Dans la figure 2.3, nous donnons un exemple de représentation d'un document avec deux vues: la vue V comporte les composants logiques (Book, Title, Chapter, Section, etc) et la vue V' comporte les composants physiques (Volume, Page, Line, etc).

Le problème des recoupements de segments qui n'avait pas de solution dans les PAT expressions a été résolu dans le modèle des "overlapped lists" [CCB94, CCB95]. Ce modèle cherche à unifier la recherche textuelle et la recherche des régions nommées (chapitre, titre, etc) en offrant une indexation au sein de laquelle les mots et les régions sont traités de manière similaire. Une région correspond alors à un composant logique de la structure.

La problématique liée à la représentation de plusieurs structures d'un même document ainsi qu'à l'interrogation combinée de ces différentes structures procure un vaste champ d'investigation. Les premiers résultats obtenus par Navarro fournissentë des requêtes mixant différentes structures: structure logique, structure physique, structure de présentation, etc. La mémoire visuelle des utilisateurs, qui s'attache principalement à la disposition des éléments les uns par rapport aux autres ainsi qu'à leur forme, peut alors être utilisée dans ces requêtes. Ceci diminue considérablement l'effort demandé à l'utilisateur puisqu'il n'a plus à rechercher la correspondance entre ce qu'il a vu et comment cela est représenté.

### b) Les apports des vues multiples dans les requêtes

Navarro fournit les opérateurs classiques de recherche sur les documents structurés textuels que nous retrouvons dans les modèles gérant une seule structure par document. Le principal apport de ce travail se situe au niveau de la combinaison de ces opérateurs sur de multiples

structures du document.

Considérons l'exemple pris dans [NBY96] pour lequel un livre est représenté par trois vues différentes $V$, $V'$ et $VP$. La vue $V$ est la vue classique représentant la structure logique du document et elle est donc composée des constructeurs représentant les composants logiques : Book, Introduction, Chapter, etc. La vue $V'$ est une représentation de la structure physique du document puisqu'elle comporte les constructeurs suivants : volume, page et line. Enfin, la troisième et dernière vue, $VP$, est la vue destinée à représenter les aspects liés à la présentation du document : italics, emphasize, etc.

Dans le modèle de Navarro que nous avons présenté en section 2.3.2, ces trois vues sont indépendantes tout en étant rattachées au même référent, à savoir les segments formant le texte. L'approche considérée pour l'interrogation n'est pas de rechercher dans les hiérarchies complètes mais de trouver rapidement un petit nombre de candidats pour la requête et d'éliminer ceux qui ne peuvent satisfaire les besoins exprimés dans la requête. Pour cela, le choix s'est porté sur la définition d'une algèbre sur les ensembles de noeuds. Ainsi chaque opérateur structurel prend en entrée un ensemble de noeuds et donne en résultat un autre ensemble de noeuds. Ces ensembles de noeuds sont des sous-ensembles de l'ensemble des noeuds des hiérarchies structurelles, noté $\mathcal{N}$ dans le modèle. Les opérations de recherche dans le texte sont quant à elles exécutées au niveau des feuilles de l'arbre syntaxique des requêtes. Et une technique d'index est utilisée afin de transformer le résultat de ces opérations en un ensemble de noeuds. Comme le montre la figure 2.4, les opérateurs structurels sont clairement séparés des opérateurs recherche textuelle. Le point de jonction entre ces deux types d'opérateurs s'effectue par au niveau des résultats qui sont dans les deux cas des ensembles de noeuds.

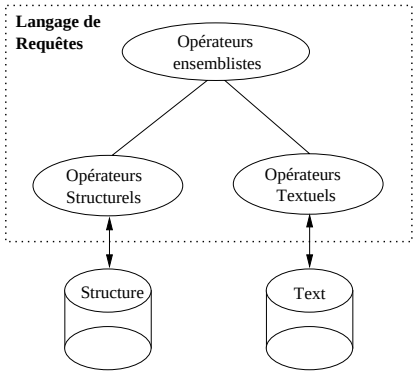



**Figure 2.4.** Structure du langage de requête de Navarro & al. [NBY96]

En fait, les opérateurs structurels permettent de spécifier un nom de constructeur, par exemple *chapter*, afin de retourner l'ensemble des noeuds qui sont des chapitres. Et les opérateurs ensemblistes agissent eux sur la composition des noeuds, leur position dans la séquence ou sur des opérateurs ensemblistes classique (intersection, union, différence).

Prenons comme premier exemple la requête suivante qui combine différentes vues :

**Requête 11**
*italics* **before***(20) (figure* **with** *"earth") (page)*

Dans la requête 11, l'utilisateur a eu recours à sa mémoire visuelle : il cherche un segment de texte en italique situé avant une figure à propos de la terre, ceci dans une même page. L'opérateur **with** effectue une recherche textuelle alors que la spécification a **before(n) (b)** exprime que l'on recherche un composant de nom a situé avant un composant b avec une distance maximale entre les deux de n symboles. Les noms a et b sont des noms de constructeur. Dans notre cas, un constructeur représentant un composant de présentation (italics) est mêlé à un composant physique, en l'occurrence la page.

Considérons maintenant les deux requêtes suivantes. La première est similaire à la requête 7 exprimée dans le langage du système MAESTRO. La seconde comporte une recomposition avec l'opérateur + qui réalise l'union des noeuds.

**Requête 12**
*title* **parent** *(Section* **parent** *Chapter)* **with** *"soft"*

La requête suivante permet de retourner pour chaque livre de la base son titre, le chapitre ayant pour titre "Introduction" et le chapitre ayant pour titre "Conclusion".

**Requête 13**
*title* **parent** *Book* + *(Chapter* **parent** *(title* **with** *"Introduction"))* + *(Chapter* **parent** *(title* **with** *"Conclusion"))*

La puissance d'expression de ce langage lié au modèle de représentation introduit réellement un apport dans le domaine. Cependant, il nous semble important d'insister sur deux points. Premièrement, nous pouvons regretter que le modèle n'autorise pas la définition d'attributs sur les composants structurels et que des techniques de recherche d'informations n'aient pas été intégrées. Enfin, l'usage qui est fait ici de la structure physique, c'est-à-dire la structure de présentation du document, est critiquable du fait que celle-ci est dépendante de l'environnement. Il aurait donc été souhaitable de voir apparaître dans la requête une référence explicite au type de structure physique que l'utilisateur souhaite interroger : présentation du document en mode A4, portrait, etc. Ceci aurait permis d'éviter toute confusion entre ce que fait le système et ce que l'utilisateur croit : il croit interroger des documents en mode A4 portrait alors que le système traite une vue A3 paysage des documents.

### 2.3.3 Les documents textuels structurés en recherche d'informations

Dans le domaine des systèmes de recherche d'informations, il existe différentes manières d'aborder les documents structurés. Soit on considère une structure dynamique déduite du contenu des documents, en appliquant par exemple des théories du discours fournissant une segmentation multi-paragraphe [HP93, Hea94]. Soit on considère une structure statique, provenant de la structure logique du document. Nous pensons comme Paradis [Par96] que la structure logique reflète suffisamment l'organisation des idées de l'auteur pour la considérer comme le support suffisamment fiable de la structure thématique du document. Nous privilégions donc la présentation de systèmes adoptant une structure statique.

### a) Représentation des documents textuels structurés

Nous allons dans cette partie nous intéresser particulièrement à deux modèles de représentation issus de deux systèmes de recherche d'informations : le système IOTA et le système PIF. D'autres modèles de représentation auraient pu avoir leur place ici tels que les approches cherchant à représenter le contenu des données textuelles de manière statistiques. Les deux systèmes que nous avons choisis donnent une vision des possibilités de représentation des documents textuels structurés : mesures statistiques, usage de thésaurus, usage des relations de structure, etc. Ces multiples possibilités de représentation démontrent la richesse des données textuelles et la variété des approches qui en découle.

Le système IOTA tire son originalité de l'adaptation aux documents structurés des mesures classiques de représentativité des termes d'indexation [Sal71]. Le système PIF propose un langage destiné à la représentation des documents textuels structurés en reprenant des propriétés provenant de théories du discours.

Nous présentons aussi, au travers du système HyperRIME [Khe95] et des travaux de Lee & al. [LYYB96], les stratégies d'indexation plus générales dans lesquelles l'information se propage selon les relations de structure. Ces travaux présentent la particularité de pouvoir s'appliquer à d'autres types de données.

*Le système IOTA*

Le système IOTA [Ker84, Def86, CDBK86] est destiné à l'interrogation de fonds documentaires composés de documents textuels structurés, en l'occurrence des normes d'exploitation et de fonctionnement du CNET. IOTA est un système de recherche d'informations intégrant un module d'indexation automatique des documents textuels et un module d'interrogation. L'approche classique d'indexation consiste à donner une représentation du contenu du document complet en ignorant la structure. Le corpus est alors constitué d'autant d'index qu'il y a de documents dans la base (un index par document). Le système IOTA n'adopte pas cette approche puisque le document n'est pas considéré comme une entité atomique mais comme un ensemble de composants issus de la structure logique. Chaque composant de la structure, appelé *unité d'indexation*, possède sa propre représentation et le corpus est alors composé d'autant d'index qu'il y a d'unités d'indexation, c'est-à-dire qu'il comporte un index par composant structurel.

Nous ne détaillons pas la phase d'extraction des connaissances du processus d'indexation automatique des unités puisque là n'est pas notre propos (le lecteur pourra se référer à [Bru87, Ker84]), mais nous allons éclairer la phase d'exploitation de la structure logique dans ce processus.

De manière simplifiée, le processus d'indexation correspond à une analyse automatique du texte reposant sur l'extraction de groupes nominaux. Les groupes nominaux extraits qui sont reconnus dans le thésaurus sont alors considérés comme des termes d'indexation. Ces termes sont pondérés selon des critères statistiques exprimant la représentativité mutuelle entre le terme et l'unité d'indexation.

Soit le corpus $D$ constitué de $N_D$ unités d'indexation $d_i : D = \{d_i\}$. Parmi ces unités, les $d_{fi}$ sont les unités d'indexation feuilles de la structure logique des documents. Soit $T = \{t_j\}$ l'ensemble des termes d'indexation du corpus correspondant à des groupes nominaux. Soit $TI(d_i)$ l'ensemble des termes d'indexation de l'unité d'indexation $d_i$ avec $TI(d_i) \subseteq T$. Soit les

fonctions suivantes :

- $FDOC(t_j)$ : nombre d'unités d'indexation dans le corpus $D$ où le terme d'indexation $t_j$ existe.
- $FTOT(t_j)$ : nombre total d'occurences du terme d'indexation $t_j$ dans le corpus $D$.
- $FLOC(t_j, d_i)$ : nombre total d'occurences du terme d'indexation $t_j$ dans l'unité d'indexation $d_i$. Il s'agit de la fréquence locale.
- $TAILLE(d_i)$ : taille de l'unité d'indexation $d_i$. Cette taille est définie en nombre total d'occurences des termes d'indexation de l'unité :

$$TAILLE(d_i) = \sum_{t_j \in TI(d_i)} FLOC(t_j, d_i)$$

Soit la représentativité d'un terme d'indexation $t_j$ par rapport à une unité d'indexation $d_i$ notée $REP(t_j/d_i)$ et la représentativité d'une unité d'indexation $d_i$ par rapport à un terme d'indexation $t_j$ notée $REP(d_i/t_j)$.

$$REP(t_j/d_i) = \frac{FLOC(t_j,d_i)}{TAILLE(d_i)} \quad et \quad REP(d_i/t_j) = \frac{FLOC(t_j,d_i)}{FTOT(t_j)}$$

Comme nous l'avons dit précédemment, une unité d'indexation correspond à une sous-arborescence de l'arborescence structurelle logique du document. Le processus d'indexation est appliqué sur les unités d'indexation constituant les feuilles de l'arborescence structurelle, les $d_{fi}$. A partir de cette indexation initiale, les unités d'indexation correspondant à des éléments non feuilles de l'arborescence structurelle sont calculées en utilisant les unités d'indexation des feuilles. Il y a donc remontée des informations extraites au niveau des feuilles vers la racine du document d'après le processus récursif suivant :

1. Les unités d'indexation feuilles $d_{fi}$ sont indexées automatiquement et de manière indépendantes. Elles admettent donc les deux représentativités décrites précédemment.
2. Les unités d'indexation de niveau supérieur sont alors indexées dynamiquement en fonction des représentativités de leurs unités filles. Considérons l'unité $d_k$ ayant pour filles les unités $d_i$, $i = 1, \ldots, n$.

$$REP(t_j/d_k) = \frac{\sum_{i=1,\ldots,n} FLOC(t_j/d_i)}{\sum_{i=1,\ldots,n} TAILLE(d_i)} \text{ et } REP(d_i, t_j) = \sum_{i=1,\ldots,n} REP(d_i, t_j)$$

Dans le système IOTA, les mesures de représentativité des termes d'indexation sont des mesures classiques mais qui ont été adaptées aux documents structurés et qui sont calculés dynamiquement à partir des mesures de représentativité calculées au niveau des unités d'indexation feuilles. Il s'agit ici d'une première étape vers un processus de propagation des informations en fonction des caractéristiques du document. Dans le système PIF et via son langage de représentation, François Paradis va aller plus loin encore sur cette voie.

*Le système PIF*

Paradis [PB96] définit un langage de représentation pour les documents textuels structurés en intégrant des caractéristiques issues des théories du discours. Ce langage permet la spécification des informations nécessaires à la recherche des documents textuels. La plupart de ces informations sont extraites de la norme TEI [TEI94] (Text Encoding and Interchange). Ce langage permet à la fois de décrire le contenu du document textuel, les caractéristiques de ce document mais aussi les relations structurelles du document. Ce langage est complété par un ensemble de règles de dérivation qui, à partir des informations décrites par le langage de représentation, permettent de déduire une nouvelle forme d'information, les thèmes du document. Ces règles de dérivation reposent sur les trois types de structure décrites dans le langage de représentation :

**Structure linguistique** : elle reflète l'organisation syntaxique des symboles du document textuel. Un symbole est un élément qui apparaît dans le texte, par exemple un mot, un groupe nominal ou une phrase. La granularité du symbole n'est pas définie et reste donc libre.

Cette organisation des symboles est décrite à partir de deux relations de base : la relation de composition notée *part* et la relation de séquence notée *seq*. La composition permet de regrouper des symboles pour former des symboles de granularité supérieure, par exemple la composition de mots forme un groupe nominal. La séquence détermine le sens de lecture naturel des symboles. Une troisième relation vient compléter la structure linguistique : la relation de *dépendance* qui permet d'exprimer qu'une relation est *subordonnée* à une autre. Si l'identificateur $s_2$ correspond à «d'» et $s_3$ correspond à «information», la relation de dépendance exprime que $s_2$ est dépendant de $s_3$ : dépendant$(s_2, s_3)$.

Les relation *part* et *seq* sont antisymétriques, non réflexives et injectives. La relation de *dépendance* est antisymétrique et non réflexive. Les relations de la structure linguistique sont des règles syntaxiques pour l'organisation des symboles qui dépendent principalement de la langue du document.

**Structure logique** : elle reflète l'organisation d'abstractions logiques qui représentent des parties du document. Il s'agit de parties du document d'une granularité plus grande que celle de la structure linguistique et donc d'une extension de la structure linguistique. La structure logique reprend les mêmes relations de composition et de séquence que celles de la structure linguistique mais cette fois-ci appliquées à des abstractions logiques. Une nouvelle relation liant des abstractions logiques est introduite. Il s'agit de la relation de référence, notée *ref*, qui permet d'introduire des renvois et des liens entre parties de document.

Les relations de la structure logique sont des règles syntaxiques pour l'organisation des abstractions logiques définies a priori par le type du document (chapitre, section, sous-section, etc).

Paradis définit une abstraction logique comme toute partie de document contenant suffisamment d'informations pour être compréhensible indépendamment des autres parties.

**Structure de discours** : elle reflète l'organisation des idées contenues dans le document et que l'auteur souhaite transmettre au lecteur du document. C'est l'organisation de ces idées qui rend un document compréhensible.

Deux relations apparaissent dans la structure de discours : la relation de dominance, *domine*, et la relation d'intention *intention*. La relation de dominance exprime l'agrégation des idées et elle repose complètement sur la relation *part* de la structure logique. Les informations contenues dans les abstractions logiques, liées entre elles par la relation *part*, s'agrégent en suivant la relation de dominance. Il y a donc à la fois la composition des abstractions logiques et composition des informations contenues dans ces abstractions. D'autre part, la relation d'intention désigne explicitement les idées présentes dans une partie de document et qui se réfèrent à une autre partie du document : les modèles de représentation des documents textuels structurés pour la RI sont décrits dans la section a).

A partir du langage comportant ces diverses relations de structure, des règles de dérivation sont introduites afin d'expliciter le contenu des documents textuels. Prenons quelques exemples de ces règles. Tout d'abord nous donnons la règle qui permet de dériver la relation de dominance à partir de la relation de composition de la structure logique :

**Règle 1 (Structure de discours)**
$(\text{part}(\sigma_1, \sigma_2) \wedge \text{division}(\sigma_1) \wedge \text{division}(\sigma_2)) \Rightarrow \text{domine}(\sigma_1, \sigma_2)$

Les $\sigma_1$ et $\sigma_2$ sont des abstractions logiques et leur composition par la relation *part* ($\sigma_1$ est composée de $\sigma_2$) implique l'existence de la relation de discours *domine* entre les deux abstractions. Maintenant nous considérons les thèmes qui apparaissent dans ces abstractions et qui sont indiqués à l'aide de la relation thème($\sigma_i$, $\epsilon$) qui signifie que le thème $\epsilon$ apparaît dans l'abstraction logique $\sigma_i$.

**Règle 2 (Héritage ascendant)**
$(\forall \sigma_i, \text{domine}(\sigma, \sigma_i) \supset \text{thème}(\sigma_i, \epsilon)\{\gamma\}) \Rightarrow \text{thème}(\sigma_i, \epsilon)\ \{H6\}$

Cette règle signifie qu'un thème $\epsilon$ est propagé vers les sections logiques mères uniquement s'il apparaît dans chacune des sections logiques filles. La notation $\{H6\}$ indique la provenance du thème, c'est-à-dire l'identificateur de la règle qui a permis la propagation de ce thème.

La définition de ces règles permet à Paradis de représenter différentes stratégies de propagation des thèmes au sein du document structuré selon la relation de composition. Il faut noter que la règle ayant permis la propagation d'une information est toujours conservée dans le résultat afin de pouvoir être utilisée par la suite lors de l'interrogation des documents. En effet, la confiance dans chacune de ces règles n'est pas égale et donc le classement des réponses sera basée sur l'origine de l'information, c'est-à-dire la règle qui a produit le thème au niveau de l'abstraction logique.

Nous reviendrons sur ce langage de représentation lorsque nous décrirons notre modèle de représentation des documents structurés puisque nous nous sommes inspirés de certaines de ces caractéristiques. Le système PIF comportant un module d'indexation automatique des documents démontre la faisabilité de cette approche.

*Le système HyperRIME*

L'approche proposée par IOTA a été reprise par la suite de manière plus générale dans les travaux de Kheirbek [Khe95] appliqués à un corpus constitué de documents hypertextuels. La stratégie d'indexation est assez similaire à celle de IOTA puisque ce sont les éléments feuilles de structure qui sont indexés et le résultat de cette indexation se propage vers les éléments parents

qui sont plus hauts dans la structure. Dans cette optique, chaque élément de structure admet une représentation de son contenu. Pour un élément feuille, la représentation est obtenue par un processus d'analyse du contenu. Pour un élément non feuille, la représentation est calculée en fonction de la représentation de ses fils dans la structure logique. Deux opérateurs de calcul définissant deux types de remontée sont proposées : le premier opérateur, noté $\otimes$, réalise l'intersection du contenu des représentations des éléments fils pour calculer celle de l'élément père, le second opérateur, noté $\oplus$, réalise l'union du contenu des représentations des éléments fils pour calculer celle de l'élément père.

Kheirbek privilégie le second opérateur (union) qui correspond mieux selon lui à la sémantique de la composition structurelle. Il explique ainsi que le fait qu'une partie $P$ soit composée d'une partie $P_1$ et d'une partie $P_2$ implique que l'indexation de $P$ doit tenir compte à la fois de l'indexation commune à $P_1$ et à $P_2$, mais aussi des parties de l'indexation spécifique à $P_1$ et spécifique à $P_2$.

Complémentairement à ce choix, Kheirbek décrit une stratégie de recherche générique reposant sur l'application du modèle logique de recherche d'informations. Cette stratégie utilise la propriété de l'opérateur $\oplus$ :

$$I(d) = I(d_1) \oplus I(d_2) \Rightarrow \begin{cases} I(d) \rightarrow I(d_1) \\ et \\ I(d) \rightarrow I(d_2) \end{cases}$$

Cette propriété signifie que l'index d'un composant implique logiquement l'index de chacun des composants qui sont des descendants structurels de ce composant.

Dans les expérimentations menées par Lee & al. [LYYB96] sur des techniques d'indexation de documents structurés afin de les comparer en terme de temps d'accès et d'espace requis par les index, la stratégie de remontée des informations vers la racine selon l'opérateur d'union est utilisée et représentée de différentes manières. Trois méthodes de représentation sont présentées. A l'origine seuls les composants (noeuds) feuilles de la structure comportent des index et les index des composants non feuilles sont l'union des index de leurs composants fils. Les index prennent la forme d'ensemble de termes.

**ANWR** (Inverted Index for All Nodes With Replication) : pour chaque composant, feuille et non feuille, les index contiennent tout les termes d'indexation qui les concernent.

**LNON** (Inverted Index for Leaf Nodes Only) : seuls les index des composants feuilles contiennent des termes d'indexation. Les index des composants non feuilles sont calculés à l'interrogation.

**ANOR** (Inverted Index for All Nodes Without Replication) : cette stratégie est basée sur le fait que des composants frères peuvent admettre des termes d'indexation commun. Dans ce cas ces termes sont remontés vers le composant père. Les index des composants sont calculés à partir de la formule suivante :

$$I(d_i) \cup I(ancestors) \cup I(descendents)$$

Nous pouvons constater que cette technique ainsi que cette formule ne sont valables que si l'ensemble des frères admettent un même terme d'indexation. Cela limite considérablement la portée de cette stratégie de représentation.

De ces trois stratégies, il ressort que la stratégie ANOR est la plus efficace en terme d'espace pour le stockage des index, mais aussi en terme de temps d'accès. Cependant cette efficacité est conditionnée par le taux de termes d'indexation remontés des fils vers les parents. Un taux égal à $b$ ($0 \leq b \leq 1$) signifie qu'un terme d'indexation d'un composant a une probabilité $b$ de remonter vers le composant père. En pratique, lorsque ce taux est inférieur à 0,1 il n'y a pas gain de la méthode ANOR vis-à-vis des autres méthodes. Il s'agit donc d'estimer l'apport réel d'une telle stratégie sur un corpus relativement représentatif. Par ailleurs, cette technique repose ici sur une représentation à base de termes d'indexation, nous pouvons nous interroger sur l'adaptation de cette technique à d'autres formalismes de représentation tels que les graphes conceptuels [OP97a] ou les logiques descriptives.

Ces travaux ont le mérite de montrer que le traitement des documents structurés est coûteux tant en terme d'espace requis par les index qu'en terme de temps d'accès. Le fait de ne plus considérer le document comme une entité indivisible mais comme une agrégation de $n$ composants multiplie d'autant le nombre d'index requis. C'est l'une des raisons pour lesquelles les travaux dans le domaine de l'interrogation des documents structurés n'adressent pas le problème de la représentation du contenu des documents et préfèrent adopter des techniques de "pattern matching" plutôt que des techniques de recherche d'informations.

### b) Usage de la structure lors de l'interrogation

Nous reprenons les caractéristiques d'interrogation des trois systèmes pour lesquels nous avons décrit les modèles de représentation: le système IOTA [Ker84], le système PIF [PB96] et le système HyperRIME [Khe95].

*Le système IOTA*

Le système IOTA [Ker84, Def86, CDBK86], dont nous avons décrit le processus d'indexation reposant sur la structure, présente un processus d'interrogation dans lequel la structure des documents est largement prise en compte. Dans ce processus, les éléments manipulés sont des arborescences de la hiérarchie structurelle liés par des opérateurs booléens (AND, OR et EXCEPT) qui ont une interprétation spécifique tenant compte des inclusions entre arborescences. Nous donnons les principales caractéristiques de ces interprétations avec les opérandes $L_1$ et $L_2$ qui sont des listes de références pondérées, c'est-à-dire des références à des arborescences. Soit $a_1$ et $a_2$ des éléments respectifs de $L_1$ et de $L_2$. Le système IOTA propose conjointement à ces stratégies une notion de pertinence dans les réponses par l'intermédiaire de fonctions associées aux opérateurs booléens : $F_{op}(p_1, p_2)$ où $op$ est un opérateur booléen et $p_1$ et $p_2$ sont les poids des références.

**AND** : $AND(L_1, L_2)$ fournit la liste des arborescences qui proviennent à la fois de $L_1$ et de $L_2$. La stratégie de sélection des arborescences est alors la suivante :

- $a_1 = a_2$ : $a_1$ est une réponse.
- $a_1 \subset a_2$ : $a_1$ est une réponse. C'est la plus petite arborescence qui est sélectionnée.

– $a_1$ et $a_2$ sont des arborescences disjointes : $a_1$ et $a_2$ ne sont pas des réponses.

La fonction de mesure de la pondération associée à l'opérateur de conjonction est :

$$F_{AND}(p_1, p_2) = \frac{(p_1 * p_2)}{(1 \Leftrightarrow p_1 \Leftrightarrow p_2) + 2 * p_1 * p_2}$$

Les opérateurs OR et EXCEPT ont été traités de manière similaire: $OR(L_1, L_2)$ fournit la liste des arborescences qui proviennent soit de $L_1$, soit de $L_2$ et $EXCEPT(L_1, L_2)$ fournit la liste des arborescences qui proviennent de $L_1$ et n'ayant pas de référence dans $L_2$.

Ce système de recherche d'information, même s'il n'a été testé que sur un corpus limité (un chapitre d'un manuel NEF qui est fortement structuré d'une taille d'environ 3000 mots), présente de nouvelles caractéristiques pour l'interrogation : la formulation d'une requête sans spécification structurelle contraignante, l'usage de la structure lors de l'interrogation qui permet de retrouver des parties de document et enfin la correspondance partielle introduisant une pertinence des réponses et donc un classement de celles-ci.

Bien entendu ce système est totalement différent des systèmes ou langages de requêtes que nous avons précédemment présentés. Il s'agit ici d'un véritable système de recherche d'information qui comporte donc une phase préliminaire d'analyse du contenu des documents donnant une représentation à ce contenu. Par ailleurs, le système propose un ensemble de réponses ordonnées alors que les autres systèmes n'offrent pour la plupart qu'un ensemble non ordonné.

*Le système PIF*

Nous avons vu que la modélisation des documents proposée par Paradis [PB96] autorise plusieurs stratégies de remontée d'informations dans la structure. Ces stratégies dépendent principalement de la provenance des informations et sont contrôlées par des règles générant la propagation des informations.

La fonction de correspondance de ce système présente la particularité de manipuler des couples *justification/contenu sémantique* donnant la représentativité. Une justification décrit la provenance de l'information alors que le contenu sémantique est une valeur numérique indiquant l'importance de l'information. Chaque élément de la structure est représenté par une liste de couples et la fonction de correspondance, basée sur une fonction booléenne, détermine les éléments de la structure qui sont pertinents.

Ces éléments de la structure sont par la suite triés par ordre décroissant de représentativité avant d'être mis en forme pour être visualisés par l'utilisateur. Puisque la fonction de correspondance manipule des couples, le classement se fait d'abord sur la justification puis sur le contenu sémantique. Il faut noter que l'ordonnancement des justifications dépend des types de requêtes et c'est donc la requête qui dirige la recherche.

Paradis a choisi de comparer son système à un système de référence, le systèmePAT, et cette comparaison montre un gain en terme de précision dans les réponses. En effet, si cette approche n'introduit pas de nouvelles informations, elle permet de caractériser l'information existante de manière plus précise. Ceci favorise la précision des réponses.

*Le système HyperRIME*

Le système HyperRIME est à l'origine un système qui cherche à intégrer des fonctionnalités RI et hypertextuelles sur une base de documents structurés. Nous décrivons ici la stratégie de recherche employée dans HyperRIME. Par ailleurs, il faut savoir que ce système intègre d'autres caractéristiques telles que la création de liens hypertextuels basée sur des mesures de distance sémantique entre composants structurels.

1. $(I(d) \rightarrow q)$ et $(q \rightarrow I(d))$, c'est-à-dire $I(d) \equiv q$ : la requête est une copie de l'index du composant $d$. Le composant $d$ répond à la requête.

2. $\neg(I(d) \rightarrow q)$ et $\neg(q \rightarrow I(d))$ : le composant $d$ et la requête ne sont pas comparables.

3. $(I(d) \rightarrow q)$ et $\neg(q \rightarrow I(d))$ : le composant $d$ répond à la requête selon le critère d'*exhaustivité* mais pas selon le critère de *spécificité*. Il faut donc rechercher parmi les composants qui sont des descendants de $d$, s'il en existe, qui répondent aux deux critères. Soit un composant $d_i$ descendant du composant $d$:

   (a) $(I(d_i) \rightarrow q)$ et $(q \rightarrow I(d_i))$ : nous retrouvons le cas idéal où la requête est une copie de l'index du composant $d_i$. Le composant $d_i$ répond à la requête.

   (b) $(I(d_i) \rightarrow q)$ et $\neg(q \rightarrow I(d_i))$ : le composant $d_i$ répond à la requête uniquement selon le critère d'*exhaustivité*. Il faut encore descendre dans la structure pour vérifier si nous pouvons trouver un composant plus précis.

   (c) $\neg(I(d_i) \rightarrow q)$ et $(q \rightarrow I(d_i))$ : le composant $d_i$ ne satisfait pas la requête, il est trop générique. Nous sommes allés trop bas dans la structure.

   Kheirbek indique que la meilleure réponse sera le dernier composant qui satisfait la situation 3(b), c'est-à-dire le plus petit composant dont l'index implique la requête et dont les descendants sont trop génériques.

4. $\neg(I(d) \rightarrow q)$ et $(q \rightarrow I(d))$ : le composant $d$ ne répond pas à la requête car il ne constitue pas une spécialisation de la requête.

Cette stratégie de recherche, nommée "Fetch and Browse", repose sur les propriétés engendrées lors de l'indexation des documents structurés. L'objectif de cette stratégie est de descendre le plus possible en profondeur dans la structure afin d'extraire de celle-ci l'élément le plus petit mais aussi le plus précis.

Ce type d'approche a depuis été repris dans divers systèmes cherchant à représenter le contenu des documents structurés. Ainsi Rölleke & al. [RF96] proposent un modèle reposant sur une logique à quatre valeurs afin de retrouver des objets complexes dont l'indexation adopte l'approche de Kheirbek.

Ces approches sont difficiles à évaluer en terme de performance et en terme de qualité. Tout d'abord, il n'existe pas de collections tests permettant de les comparer. Ensuite, ces approches n'ont pour l'instant été evaluées que sur des corpus restreints pour les valider et pour vérifier l'apport en terme de précision dans les réponses.

## 2.4 Les images, des documents à facettes multiples

Les images sont des données complexes qui, comme le texte, pour pouvoir être retrouvées, nécessitent une modélisation de leur contenu. La modélisation d'une image peut admettre autant de facettes que l'image comporte de caractéristiques : caractéristiques visuelles, physiques, symboliques, etc.

Nous présentons l'approche classique des bases de données à travers le système QBIC qui ne nécessite pas d'intervention humaine pour l'indexation des images. Puis nous présentons des modèles de représentation des images fixes pour la RI tenant compte des multiples facettes des images.

### 2.4.1 L'approche des bases de données

Le système QBIC, par rapport au système Chabot, intègre de nombreuses caractéristiques supplémentaires, notamment au niveau de la représentation physique de l'image, c'est pour cette raison que nous considérons que QBIC manipule des documents à facettes multiples.

*Le système QBIC*

QBIC est composé de deux modules : un module d'indexation et de stockage et un module d'interrogation. Le premier module intègre les fonctionnalités d'analyse de l'image permettant d'extraire, entre autres, des moyennes de couleur, l'histogramme des couleurs, la texture, des formes et des positions. Ces éléments d'information sont valables aussi bien pour l'image complète que pour des objets présents dans l'image. Nous résumons les propriétés caractérisant les images dans le système QBIC dans la table 2.5.

| Image Complète | Objets de l'image |
|---|---|
| Moyenne des couleurs | Moyenne des couleurs |
| Histogramme des couleurs | Histogramme des couleurs |
| Texture | Texture |
| Forme | Position des contours |
| Localisation dans l'image | Position des couleurs |

**Figure 2.5.** Eléments d'indexation et de recherche dans QBIC [NFF+93, FSN+95]

La capacité à indexer l'image complète ainsi que les objets présents dans l'image présente une avancée particulièrement intéressante par rapport à Chabot puisqu'il devient possible lors de l'interrogation de spécifier une région de l'image ainsi que des critères sur cette région. Toutefois ce type d'indexation reste limité à des images particulières dont le processus de production est contrôlé et vérifie de nombreuses propriétés : fond de l'image uniforme, éclairage et prise de vue contrôlés, etc. Il n'est donc pas possible d'indexer automatiquement n'importe quelle image par l'ensemble de ses propriétés.

Dans les conditions optimales, c'est-à-dire pour des images vérifiant les critères cités précédemment, il devient possible à l'aide de l'interface d'interrogation de spécifier chacune des

propriétés et de dessiner les objets requis par l'utilisateur dans les images. L'utilisateur peut donc formuler les requêtes suivantes :

**Requête 3**
*15% de jaune et 13% de bleu*
*une forme de poisson dont l'intérieur est rouge, sur un fond bleu*

Ce type de requête, comme pour le système Chabot, a montré ses limites et de nouvelles propriétés ont été ajoutées. Ces propriétés sont destinées à introduire un niveau symbolique dans QBIC. Ainsi l'indexation devient semi-automatique et un intervenant humain décrit les objets présents dans l'image à l'aide de concepts (voiture, homme, etc). Les requêtes évoluent ainsi avec la possibilité pour l'utilisateur de spécifier ce que représente les formes qu'il a dessiné. Pour rechercher les images composées d'une voiture rouge, il lui suffit de dessiner un objet rouge et de spécifier que cet objet représente une voiture.

Le système QBIC a par la suite été étendu de manière à pouvoir interroger les séquences vidéos [CHN+95]. Il est intéressant de constater que les résultats obtenus avec ce système, que ce soit pour des images fixes ou pour des vidéos, restent peu satisfaisants tant que les spécifications symboliques ne sont pas utilisées. Il est en effet difficile de décrire une image par ses seules caractéristiques physiques ou par les seules formes qui la composent.

Ces modèles présentent pour la plupart l'avantage indéniable de ne nécessiter qu'un minimum d'intervention humaine dans le processus d'indexation. Toutefois les techniques actuelles de *compréhension* du contenu d'une image ne permettent pas d'obtenir des résultats suffisamment convaincants pour ne conserver au sein d'un système de recherche d'image une indexation purement basée sur l'analyse automatique du contenu d'une image. Une indexation automatique aboutit à un système de recherche d'images dont les critères de recherche sont purement physiques, à savoir la couleur dominante d'une image ou bien la description de la forme d'un objet présent dans l'image. Comme l'explique Narasimhalu [Nar95], la recherche d'images nécessite la spécification de critères symboliques, c'est-à-dire que la description de la forme d'une image n'est pas suffisante, il faut pouvoir spécifier ce que représente cette image. Il préconise donc un processus semi-automatique comportant une interaction avec un *indexeur* afin d'obtenir une représentation comportant une symbolique de l'image. Tout comme la recherche plein texte a montré ses limites, la recherche d'images sur des critères physiques doit être complétée par des critères ayant trait à la symbolique. Cette prise en compte de la symbolique nécessite donc une nouvelle forme de modélisation que nous allons voir maintenant à travers les modèles développés pour des systèmes de recherche d'informations.

### 2.4.2 Les facettes d'une image en recherche d'informations

Nous allons décrire deux modèles de représentation des images de nature différente : le modèle général EMIR$^2$ proposé par Mechkour [Mec95c] et le modèle de Meghini [Meg95]. Nous en dégagerons les caractéristiques d'un modèle de représentation général des images comportant leur description symbolique.

*Le modèle EMIR$^2$*

Le modèle proposé par Mechkour [Mec95c] est un modèle général de représentation permettant une description complète des images fixes. Ce modèle se veut indépendant de tout

formalisme de représentation de données ou de connaissances. Il nous semble important de resituer le cadre de ce travail en expliquant clairement que, contrairement aux modèles ou systèmes précédents (Chabot ou QBIC), EMIR$^2$ n'aborde pas la problématique de l'indexation automatique des images fixes, mais qu'il fournit un espace dans lequel les caractéristiques de l'image fixe peuvent être représentées.

Dans ce modèle général, l'index d'une image est décrit à travers deux niveaux : le niveau physique et le niveau logique. Le niveau de description physique se limite à un ensemble d'attributs cherchant à caractériser le format de l'image et les caractéristiques des données qui composent l'image. De son côté, le niveau de description logique reflète le contenu sémantique de l'image et il est traité avec une attention particulière. Quatre vues sont nécessaires à la description du niveau logique. Nous allons reprendre chacune d'entre elles plus précisément :

**La vue structurelle** : l'image est vue comme une ensemble d'objets qui sont soit des objets simples, soit des objets complexes, c'est-à-dire des objets composés d'autres objets. La vue structurelle prend donc la forme d'un arbre ayant pour noeuds les objets de l'image et pour arcs les relations de composition structurelle. Cette vue permet de décomposer l'image en conservant la structure des éléments présents dans l'image. Par exemple, une image qui contient un arbre et une maison, elle-même composée d'un mur et d'un toit aura dans sa vue structurelle un objet $o_{image}$ composé d'un objet $o_1$ (l'arbre) et composé d'un objet $o_2$ (la maison) qui sera lui-même composé d'un objet $o_3$ (le mur) et d'un objet $o_4$ (le toit).

**La vue spatiale** : cette vue donne la description de la forme des objets de la vue structurelle dans l'image ainsi que les relations spatiales qui lient ces objets. Les relations spatiales permettent de positionner les objets les uns par rapport aux autres dans le plan de l'image. La vue spatiale d'une image est un graphe dans lequel les noeuds sont des objets spatiaux et les arcs sont les relations spatiales. Les objets spatiaux sont caractérisés par la forme de l'élément dans le plan de l'image : point, segment, polygone, etc. Les relations spatiales élémentaires décrites dans EMIR$^2$ sont au nombre de 12 (Nord, Sud, etc) en distinguant trois espaces de représentation : espace métrique, espace vectoriel et espace topologique.

**La vue perceptive** : elle regroupe les attributs indiquant la perception visuelle d'une image. Dans EMIR$^2$, la couleur, la brillance et la texture sont les trois attributs pris en compte.

**La vue symbolique** : elle fournit une description sémantique des objets structurels en associant à chacun d'entre eux un concept générique et en offrant la possibilité d'associer des attributs formatés à ces concepts. Cette description est aussi complétée par des relations symboliques entre les concepts génériques. Ainsi la vue symbolique est composée d'objets conceptuels (les concepts génériques) liés entre eux par des relations symboliques ou complétés par des attributs formatés : la vue symbolique de l'image que nous avons décrite dans la vue structurelle reprend alors les quatres concepts maison, arbre, toit et mur sous la forme d'objets symboliques. Si de plus nous savons que l'arbre fait de l'ombre à la maison, nous avons la relation symbolique *ombre*(arbre,maison).

Le modèle EMIR$^2$ fournit donc un ensemble de vues permettant de décrire assez complètement une image fixe. Ainsi les caractéristiques de perception, de contenu sémantique et de

positionnement des objets les uns relativement aux autres sont représentées. Par ailleurs le modèle intègre des notions d'incertitude et de pertinence dans la représentation de l'image.

Le modèle opérationnel de EMIR$^2$ repose sur le formalisme des graphes conceptuels [Mec95b, Mec95d]. Les graphes conceptuels [Sow84] présentent l'avantage de pouvoir unifier assez aisément les vues du modèle de Mechkour. La représentation globale de l'image selon le modèle correspond à un unique graphe conceptuel $g$. En effet chaque objet de EMIR$^2$, objet structurel, symbolique, perceptif et spatial, est représenté par un concept particulier dans un graphe conceptuel $g$. De la même manière les relations du modèle, relations de composition, relations spatiales et relations symboliques, sont représentées par des relations entre les concepts du graphe conceptuel $g$. Les attributs sont représentés par des concepts associés à leurs objets respectifs par une relation.

L'usage des graphes conceptuels pour représenter des images a été appliqué à plusieurs reprises avec de bons résultats sur différentes bases d'images : la collection-test d'images de Paris [CM97, OP97b] et les images médicales du système PRIME [MMFB97, BFMM97].

*Le modèle de Meghini*

Meghini propose également un modèle de représentation d'images fixes [Meg95] élaboré à partir de deux composantes élémentaires : un schéma conceptuel contenant la définition des entités pouvant se trouver dans une image et une population d'images. Tout comme Mechkour, les méthodes d'extraction automatique des connaissances ne sont pas abordées et seul l'aspect modélisation du contenu de l'image est décrit. Une image est alors vue comme un objet à trois niveaux :

**Forme** : l'image est vue comme une combinaison de régions colorées et sa représentation se définit par le triplet $(A, \pi, F)$. $A$ est une région de l'image, $\pi$ est une partition de $A$ c'est-à-dire un ensemble de points, $\pi = \{S_1, \ldots, S_n\}$ et enfin la fonction $F$ associe à chaque point de $A$ une couleur.

**Contenu** : ce niveau comprend un ensemble d'éléments pour la modélisation sémantique du contenu des images et son organisation. Cette modélisation se compose donc d'entités inter-reliées et elle repose sur trois mécanismes d'abstraction : la classification des entités (classes décrivant des collections d'objets similaires), leur agrégation (définition des propriétés des classes) et leur spécialisation (taxinomie des classes avec une relation "IsA").

**Liaison** : ce niveau établit la jonction entre les deux niveaux précédents. Il s'agit d'une fonction qui associe à des ensembles disjoints de spots des entités du "contenu". Il s'agit donc de la construction de la représentation de l'image puisque cette fonction lie les régions physiques de l'image à des entités conceptuelles.

Plusieurs différences notables apparaissent entre ces deux représentations et nous nous attardons plus particulièrement sur deux d'entre elles : la liaison entre les données physiques de l'image et sa représentation symbolique et les relations entre les objets de l'image.

Tout d'abord nous pouvons noter que la liaison entre les données physiques qui composent l'image (les pixels) et la représentation conceptuelle de ces données physiques est assez éloigné dans les deux modèles. Le modèle de Mechkour peut être critiqué sur ce point précis puisque

cette liaison n'apparaît pas clairement. Le niveau physique ne comporte pas de description précise des données ou des régions. Le niveau spatial donne uniquement la forme de l'élément sans spécifier la région exacte dans l'image (en termes de pixels par exemple) qu'il représente. Par contre, dans le modèle de Meghini, cette liaison est très claire puisque le modèle repose sur une fonction établissant le passage entre un niveau physique, *form level*, et un niveau conceptuel, *content level*.

Les relations entre les objets de l'image sont traitées par Mourad Mechkour au niveau du modèle alors que Carlo Meghini a choisi de les traiter lors de la correspondance. Ces relations sont indispensables à notre avis car elles peuvent intervenir fréquemment dans la recherche de données spatiales.

Cependant, le traitement des relations entre objets nécessitent de la part du moteur de recherche des capacités d'inférence et de déduction [OP97b]. Or les formalismes de représentation de Mechkour, les graphes conceptuels [Sow84], et de Meghini, la logique terminologique MIRLOG [SS96], ne supportent pas les mécanismes d'inférence sur les relations et ont donc nécessité des approches *ad-hoc*.

### 2.4.3 Synthèse

L'indexation des images pour leur recherche reste un problème ouvert. La nécessité de combiner des critères symboliques, des critères physiques et de lier ces critères par des relations est évidente. Cependant il n'existe pas à l'heure actuelle de technique permettant d'extraire d'une image l'ensemble de ces composants avec suffisamment de précision.

Les travaux actuels s'orientent vers la combinaison et la fusion de modèles provenant de l'analyse d'image et de la vision par ordinateur pour garantir que le procédé d'indexation puisse s'adapter à tout type d'images. Ces travaux menés au MIT par Minka et Picard [PM95, Pic96, MP96] vont dans ce sens.

Nous n'avons pas, pour notre part, l'objectif de décrire ce type de processus et nous préférons spécifier quels sont les éléments indispensables à la recherche des images en posant comme hypothèse qu'un processus automatique et/ou manuel permet d'obtenir ces éléments. Actuellement, ce processus est assisté par un intervenant humain.

Nous pouvons aussi noter que certains travaux cherchent à intégrer la recherche d'images dans un cadre plus général en prenant en considération l'environnement. Ces travaux n'abordent plus la recherche dans des bases constituées uniquement d'images mais considèrent que l'image est un élément parmi d'autres documents et qu'il est donc souhaitable d'utiliser cet environnement pour indexer les images. C'est le cas par exemple des travaux menés par [HSD97] qui considèrent des images dans un hypertexte et qui utilisent le contenu des destinations des liens hypertextuels proche des images pour indexer les images.

## 2.5 Les documents multimédias

Les documents multimédias présentent la particularité d'englober tout les types de documents que nous avons vus précédemment. Ces documents restent encore délicats à manipuler dans le cadre d'applications cherchant à les modéliser et à les rechercher. Avec ce type de

documents, les SGBD et les SRI tendent vers des travaux communs puisqu'il est apparu nécessaire de représenter le contenu sémantique de ces documents.

Comme pour les documents structurés avec la norme SGML, les documents multimédias sont en cours de normalisation. Ainsi, nous en sommes au stade où plusieurs normes sont en cours de développement, principalement MHEG [Pri93] et HyTime (Hypermedia/Time-based Structuring Language) [Erf94, DD94], et il devient maintenant impératif de fournir des applications reposant sur ces normes afin de démontrer leurs capacités. HyTime est construit au dessus de SGML et l'étend par des aspects temporels et hypermédias. HyTime hérite de SGML la possibilité de définir des types de documents et requiert, comme SGML, des mécanismes externes pour spécifier la présentation des documents.

Nous donnons dans la figure 2.6, reprise de Rutledge & al. dans [RvOHB97], une classification des principales normes de représentation et d'encodage des documents multimédias selon deux axes : leur indépendance vis à vis de la présentation et leurs fonctionnalités hypermédias. Il apparaît que l'indépendance entre la structure logique du document et ses données physiques permettent de conserver une indépendance vis à vis de la présentation. C'est le cas pour SGML et HyTime. MHEG ne fournit pas une telle indépendance puisqu'il propose principalement une norme pour l'organisation du contenu. Les normes PostScript et PDF sont quant à elles totalement dépendantes de leur présentation. Les fonctionnalités hypermédias qui englobent les aspects hypertextuels ainsi que l'intégration des données multimédias temporelles dans les document sont fortement présentes dans HyTime et MHEG. Elles sont limités à des fonctionnalités de navigation hypertextuelle dans HTML et PDF et peu ou pas présentes originellement dans SGML et le format PostScript.

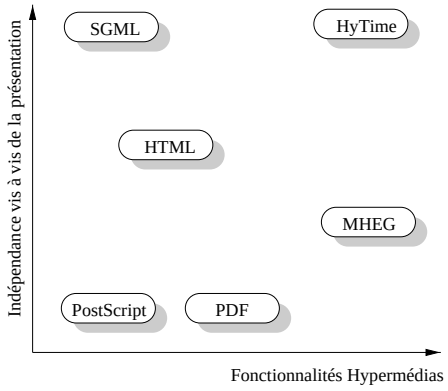



**Figure 2.6.** Les normes de représentation des documents multimédias

Pour notre part, nous nous intéressons uniquement à la recherche sur les documents multimédias. Nous présentons succinctement comment les SGBD actuels traitent ce point particulier, puis revenons sur le langage de requête de la norme HyTime, appelé HyQ [Kim93], avant d'aborder le traitement des documents multimédias en recherche d'informations.

### 2.5.1 Les documents multimédias dans les bases de données

Les SGBD actuels sont capables de gérer de nombreux médias, textes, images, séquences audios, vidéos, etc, cependant ils ne procurent que rarement l'ensemble des fonctionnalités nouvelles que nécessitent ces données. Nous en citerons deux parmi les plus importantes qui sont citées par Klas et Aberer dans [KA95]: l'intégration des aspects temporels et la représentation de ces données par des abstractions. C'est ce dernier point qui est le plus intéressant dans le cadre de la recherche sur ce type de documents.

En effet, l'introduction des données multimédias nécessitent, à la différence des données classiques, des modèles de représentation permettant de décrire le contenu de ces données. Elles nécessitent aussi des processus d'indexation afin d'extraire ces descriptions. Analyti et Christodoulakis propose un modèle de représentation des données multimédias dans [AC95] ainsi que des stratégies de recherche sur le contenu reposant sur la modélisation qui a été faite. La modélisation est basée sur une définition de classes d'objets correspondant à chaque média. De plus, des objets fournissant les descriptions "sémantiques" de ces objets sont décrits séparément des médias. Le processus d'interrogation intervient sur ces derniers objets. Toutefois, comme dans la plupart des SGBD dits "multimédias", les fonctionnalités d'indexation sur ces nouvelles données se doivent d'être contrôlées par un indexeur humain et restent incomplètes lorsqu'elles sont automatiques.

Le système MULTOS a été le premier à proposer une interrogation sur le contenu des documents multimédias. Nous avons présenté plusieurs caractéristiques de son langage de requêtes dans la section b). Des modèles pour les données textuelles [Sav90] ainsi que pour les images [CR90, CDY90] ont été proposées pour ce système. Nous pouvons toutefois regretter que, dans ce système, les images demeurent des *composants passifs*, à savoir qu'elles ne peuvent pas être interrogées sur leur contenu.

### 2.5.2 Un langage de requête pour les documents multimédias : HyQ

Nous rappelons que le langage HyTime [DD94] est une norme ISO 10744 qui utilise la syntaxe SGML pour offrir un modèle de données permettant de représenter des documents multimédias. Pour situer le langage HyTime par rapport à SGML, il nous faut dire que HyTime propose des extensions hypermédias non supportées par SGML (références externes, liens actifs, etc) ainsi qu'un cadre structuré, sous l'appellation "architectural forms", supportant les dépendances temporelles entre les éléments des documents.

L'objectif premier de cette norme n'est pas de réinventer ce qui existe tant au niveau des hypermédias que des données multimédias temporelles mais elle cherche à créer un consensus autour d'un modèle exprimant les dépendances entre objets. Nous nous intéressons plus particulièrement au langage HyQ qui permet l'interaction avec les systèmes utilisant le mécanisme d'adressage de HyTime.

*Les mécanismes d'adressage de HyTime*

Le langage HyTime permet de représenter des documents multimédias structurés et ses mécanismes d'adressage offrent la possibilité de localiser des objets dans des hiérarchies, c'est-à-dire retrouver des éléments dans une structure. Trois modes d'adressage doivent être

différenciés :

*Adressage par nom* : accès aux objets par leur nom formel qui se doit d'être unique dans un contexte donné (ensemble des documents d'une base).

*Adressage par coordonnées* : accès aux objets par leur position dans un espace donné. Le langage HyTime propose quatre mécanismes d'adressage par coordonnées :

*Tree Location Adress* : adressage dans une arborescence. L'expression '1 2 3 1' indique le premier fils du troisième fils du second fils de la racine.

*List Location Adress* : adressage dans une liste. L'expression '2 3' retourne les objets 2, 3 et 4 de la liste. Le premier chiffre localise l'objet de début alors que le second indique la dimension de la liste retournée.

*Path Location Adress* : adressage dans une arborescence par un chemin d'accès. Une arborescence est vue comme une matrice où les lignes sont les hauteurs de l'arbre et les colonnes sont les chemins d'accès aux feuilles de l'arbre. Un chemin s'exprime par '2 1 1 -1' où '2 1' sélectionne la seconde colonne de la matrice représentant l'arborescence ('2 2' aurait fourni la seconde et troisième colonne) et où '1 -1' adresse la totalité de la colonne.

*Relative Location Adress* accès aux objets d'une arborescence basé sur les relations des ces objets avec d'autres objets de l'arborescence.

*Adressage sémantique* : accès aux objets en fonction de leurs propriétés. Il peut s'agir de localiser des objets par la valeur de leurs identifiants ou de leurs attributs. Dans la requête suivante, nous sélectionnons toutes les valeurs de l'attribut STATUS dans DOMROOT.

**Requête 14**
*<HyQ>*
*proploc(DOMROOT ATTVAL[STATUS])*
*< \HyQ>*

Une description complète des modes d'adressage de HyTime peut être trouvée dans [Kim93] et [DD94].

*Exemple d'usage de HyQ*

Le Maitre, dans [MMR95], caractérise ce type de langage de requêtes par sa spécificité à ne traiter que des arborescences par opposition aux extensions des langages déclaratifs existants permettant de traiter les documents structurés (Blake pour SQL [BCK+94] et Christophides pour OQL [Chr96]).

HyQ manipule des listes d'objets en entrée comme en sortie. La sélection et la manipulation des données contenues dans les documents HyTime se font par des requêtes ou des fonctions de HyQ. Dans l'exemple de la requête 15, nous recherchons sur le domaine identifié par "mydocument" les objets de la liste DOMTREE qui sont de type chapitre.

**Requête 15**
*<HyQ qdomqin=mydocument>*
*Select (DOMTREE EQ(Proploc(CAND GI) "Chapitre"))*
*< \HyQ>*

Ce langage permet d'extraire des sous-arborescences d'une arborescence ce qui signifie qu'il est possible d'extraire une partie d'un texte ou encore des séquences d'une vidéo. Ce langage de requête semble particulièrement approprié pour l'élaboration de scripts décrivant des présentations d'objets multimédias puisqu'il permet de manipuler indifféremment tout type de médias. Des exemples de ce type d'usage sont donnés dans [Erf94].

En conclusion de ce bref aperçu du langage HyQ, il est intéressant situer le langage SgmlQL [MMR95] qui traite plus spécifiquement les documents SGML. Ce langage a été conçu pour manipuler des documents SGML sous tout leurs aspects: structure, attributs et références hypertextuelles. La manipulation structurelle est vue sous la forme d'une requête constituée d'une suite d'opérations d'extraction, de construction ou de transformation d'arbres. Une phase préliminaire à l'exécution de ces opérations est la localisation des arbres. Le langage propose alors des primitives reprenant les modes d'adressage de HyTime pour effectuer cette localisation.

Ces deux langages basés sur des modes d'adressage donnant un accès direct aux données d'une arborescence sont capables de restituer des parties d'un document structuré mais la lacune récurrente réside dans la non préparation des documents pour l'interrogation. Il s'agit de techniques d'accès dans des arborescences et de construction/manipulation de ces arborescences selon des critères syntaxiques. Cependant l'aspect sémantique est trop peu présent selon nous dans ces travaux. Nous allons maintenant voir comment cet aspect peut être pris en compte.

### 2.5.3 Les documents multimédias en recherche d'informations

L'interrogation par le contenu des documents structurés multimédia est difficile à mettre en place car elle nécessite l'instanciation de modèles pour chacun des médias comme l'avait proposé le système MULTOS. De plus, les travaux actuels partent souvent du postulat que les données sont indexées et n'aboutissent que rarement à un système complet comprenant les phase d'indexation et d'interrogation des documents.

Nous décrivons le travail que nous avons mené sur l'application PRIME-GC (pour Prototype de Recherche d'Informations MÉdicales - plate-forme Graphes Conceptuels) [FM96, Fou97, BFM$^+$97, BFMM97] qui manipule des données textuelles et des images. Dans cette application, nous définissons un processus d'indexation intégrant la structure puisque les informations se propagent des feuilles de la structure vers l'élément racine. Il s'agit ici d'une approche comparable à celles de Lee & al. [LYYB96], Kheirbek et Chiaramella [Khe95, CFM96] ou Fuhr et Rölleke[RF96].

*L'application PRIME-GC*

Dans notre approche, nous considérons une *structure d'indexation*, c'est-à-dire que nous définissons en amont du processus d'indexation les éléments de la structure qui doivent être indexés. Nous écartons ainsi des éléments qui n'apporteraient rien en tant que réponse ou bien qui nécessiteraient de la part de l'utilisateur une *fouille* dans ces éléments.

Les données de PRIME-GC sont modélisées par l'intermédiaire d'une grammaire BNF en respectant les spécificités du système d'informations en place au service du Centre Hospitalier

Universitaire de Grenoble d'où elles proviennent. Elles respectent la modélisation suivante :

<Dossier Patient> ::= <Dossier Administratif Patient> <Dossier Médical Patient>
<Dossier Médical Patient> ::= <Dossier Commun Patient> | *(<Dossier Médical de Spécialité>)
<Dossier Médical de Spécialité> ::= <Date d'Examen> <Examen>
<Examen> ::= 1*<Médecin Demandeur>
1*<Spécialiste>
<Compte-Rendu>
1*<Série d'Images>
0*1<Index Examen>
<Compte-Rendu> ::= <Texte>0*1<Index Compte-Rendu>
<Série d'Images> ::= 1* <IRM> <IRM> ::= <Image>0*1<Index Image>

Dans ce modèle de données apparaissent les éléments indexés, à savoir les Examens, les Compte-Rendus et les IRMs (Images par Résonance Magnétique). Les séries d'images ainsi que les données non médicales, les données factuelles et les éléments structurels qui sont des parents des examens ont été écartés de la structure d'indexation.

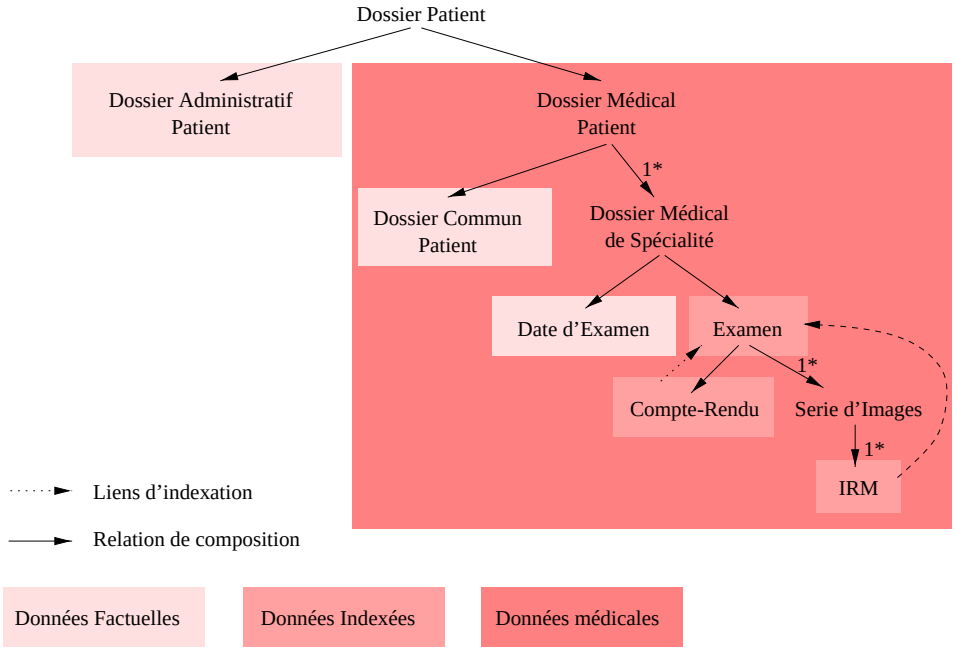



**Figure 2.7.** La structure des documents de PRIME-GC

A partir de la définition des éléments de la structure que nous indexons, nous avons utilisé le langage d'indexation défini par Berrut dans [Ber88]. Il s'agit d'un langage complexe, noté L, dont les termes prennent la forme suivante : TERME ∈ VOCABULAIRE ∪ (L × OPÉRATEUR × L). Le vocabulaire et les opérateurs de ce langage ont été établis en collaboration avec les médecins du CHU de Grenoble et comportent un vocabulaire spécialisé dont des règles

grammaticales garantissent les associations entre les éléments du vocabulaire par les opérateurs. Les opérateurs correspondent ici à des relations sémantiques telles que "en relation topologique avec", "a pour valeur", "a pour valeur locative", etc.

Ce langage d'indexation servait à l'origine pour l'indexation des textes médicaux du système RIME (les Compte-Rendus). Nous l'avons repris pour indexer le contenu sémantique des images médicales de PRIME-GC en utilisant le modèle développé par Mechkour pour représenter les images fixes [Mec95c]. Ainsi, les images indexées sont représentées par un sous-ensemble des vues décrites dans le modèle $EMIR^2$ (voir la description page 46) [Mec95c] : une vue structurelle, une vue symbolique et une vue spatiale. La vue spatiale comporte les informations géométriques décrivant les objets de l'image et les relations spatiales décrivant leurs positions relatives. La vue symbolique permet la représentation du contenu sémantique des objets de l'image. Enfin, la vue structurelle assure l'homogénéité et le dialogue entre les deux vues précédentes en décrivant la décomposition de l'image en objets.

Pour cette application particulière, nous avions donc des outils d'indexation tant pour les textes médicaux (les Compte-Rendus) [Ber88] que pour les images médicales (les IRMs) [BMB95, Bou95], il nous restait à définir un outil pour indexer les éléments mêlant les deux (les Examens). Un processus d'indexation dynamique permettant de déduire l'indexation des examens à partir des indexations des Compte-Rendus et des IRMs est utilisé. Ce processus repose sur une fonction, notée Indexation_Dynamique, qui reprend le contenu des composants d'un examen et fournit en sortie une indexation pour l'examen.

Soit $S_1, \ldots, S_N$ les composants indexés de $S$
$Indexation(S) = Indexation\_Dynamique(S_1, \ldots, S_N)$

Nous utilisons une fonction ad-hoc qui définit la *remontée* des termes d'indexation des composants de $S$, notés $S_i$ pour $i = 1 \ldots N$, vers $S$. Cette fonction exprime la *remontée* des termes communs entre deux index d'images ou entre un index d'image et un index de compte-rendu de chaque examen. Cette fonction de*remontée* des termes d'indexation s'inspire du travail plus générique que nous avons mené dans le cadre du projet FERMI [CFM96, BFM+97] : le processus de recherche de PRIME-GC est une mise en oeuvre du modèle logique, pour qu'un document D soit une réponse à une requête Q, il faut démontrer que l'implication logique $d \rightarrow q$ est vérifiée, où $d$ et $q$ sont les représentations du document D et de la requête Q. Dans le modèle logique, les documents et les requêtes sont exprimés sous la forme d'expressions logiques et la pertinence est basée sur la conséquence logique.

Nous proposons des requêtes typées, c'est-à-dire adressant une catégorie particulière d'éléments : soit les examens, soit les compte-rendus, soit les images. Nous distinguons la fonction de correspondance visant les examens et les compte-rendus de celle opérant sur les images. En effet, la représentation des images contient des éléments d'information (objets spatiaux et relations spatiales) que les autres représentations n'admettent pas et qui requièrent un traitement particulier.

La mise en correspondance sur les examens et les compte-rendus proposent deux niveaux de réponse :

1. l'index complet du document répond à la requête. Il s'agit des meilleures réponses possibles.

2. il existe des termes de l'index du document qui apparaissent dans la requête. Ces documents répondent partiellement à la requête et sont considérés comme de moins bonnes réponses.

Sachant que les index utilisent le formalisme des graphes conceptuels, nous n'avons pas actuellement d'outils permettant d'établir un ordonnancement des réponses mais nous proposons des niveaux de réponses. Nous rappelons qu'avec ce formalisme, un document répond à une requête lorsque le graphe de la requête se projette dans le graphe du document. Un terme de l'index d'un document apparaît dans une requête lorsque un terme de la requête se projette dans le graphe du document.

La correspondance sur les images est constituée de plusieurs niveaux de réponses dépendant de l'usage ou non des différentes facettes du modèle de représentation des images. Cet ordonnancement prend la forme suivante:

1. les images dans lesquelles se projette la requête,
2. les images dans lesquelles les vues symboliques et spatiales de la requête se projettent indépendamment,
3. les images dans lesquelles la vue symbolique de la requête se projette,
4. les images dans lesquelles la vue spatiale de la requête se projette,
5. les images dans lesquelles un terme de la requête se projette.

A travers cet ordonnancement, nous définissons la relation de pertinence entretenue par les images avec la requête dans laquelle nous privilégions la vue symbolique par rapport à la vue spatiale.

Dans la correspondance définie pour le système PRIME-GC, la structure n'apparaît que faiblement puisqu'il n'y a pas de recherche globale sur l'ensemble des éléments d'un document. Cependant, ce système a le mérite de mettre en évidence le bien fondé de l'approche dynamique de l'indexation des documents structurés pour fournir une représentation aux éléments de structure. Par ailleurs, PRIME-GC démontre l'utilité de différents niveaux de réponses introduits dans les fonctions de correspondance et ayant un sens. Il ne s'agit pas ici d'une valeur numérique mais du fait de l'utilisation ou non de certaines informations. Ceci permet d'expliquer pourquoi tel document répond ou tel autre ne répond pas.

## 2.6 Synthèse

Nous avons présenté plusieurs types de systèmes et langages de requêtes capables de supporter différentes sortes de documents, et principalement des documents structurés.

Nous relevons que le principal défi auquel se confrontent ces systèmes et langages de requêtes relève de la flexibilité de l'interrogation. Le document structuré est complexe de par sa nature et peut donc être interrogé de multiples façons. La flexibilité dans l'interrogation se traduit par l'introduction de techniques permettant de masquer la complexité du document lors de l'interrogation et d'accéder directement aux composants structurels pertinents.

### 2.6.1 Récapitulatif des fonctionnalités

Nous résumons dans le tableau de la figure 2.8 les différentes caractéristiques qui nous intéressent dans les systèmes et langages de requêtes que nous avons plus particulièrement présentés.

| Systèmes | Opérateurs sur la Composition | Opérateurs sur la Séquence | Usage des Attributs | Usage du Contenu | Autres Caractéristiques |
|---|---|---|---|---|---|
| MULTOS [Tha90] | chemins d'accès | | opérateurs sur les attributs alphanum. | "pattern matching" sur le texte et opérateurs sur les images | Composants actifs / passifs, la granularité des réponses est fixe |
| MAESTRO [Mac91] | chemins d'accès et usage de quantificat. | opérateurs first, last et sur les listes | attributs purement textuels sur groupes de documents (globaux) | "pattern matching" sur le texte | granularité des réponses au choix |
| POQL [Chr96] | chemins d'accès avec variables | opérations sur les listes | Attributs BD et variables d'attributs | "pattern matching" sur le texte et proximité des termes | |
| PAT expressions [ST94] | inclusion et distance structurelle | | "pattern matching" sur le texte et proximité des termes | | |
| Navarro [Nav95] | inclusion et distance structurelle | opérations sur les listes | | "pattern matching" sur le texte et proximité des termes | combinaison de vues multiples |
| IOTA [CDBK86] | | | attributs globaux | opérateurs RI | pertinence et ordonnancement des réponses |
| HyQ [Kim93], SgmlQL [MMR95] | chemins d'accès, manipulation d'arborescence | manipulation de listes | attributs BD | | opérateurs de reconstruction |
| PRIME-GC [BFM+97] | | | attributs globaux | attribut de contenu sémantique dynamique | rech. sémantique sur composants multimédias, pertinence et ordonnancement des réponses |

**Figure 2.8.** Synthèse sur l'interrogation des documents structurés

**Manipulation de la composition** : il s'agit de déterminer quels types d'opérateurs sont disponibles pour interroger sur la relation de composition de la structure. Nous reconnaissons ainsi principalement les possibilités de définir des *chemins d'accès* dans la structure, d'adjoindre des variables à ces chemins d'accès ou de spécifier des distances dans la structure.

Les systèmes MULTOS [BR90], MAESTRO [Mac91] et le langage POQL [Chr96] proposent des chemins d'accès pour construire des requêtes permettant de naviguer dans la structure des documents. POQL y adjoint la notion de variable permettant ainsi de comparer ces chemins et de les dans les réponses. Les modèles basés sur les PAT expressions [ST94] et le modèle de Navarro [Nav95] privilégient des notions d'inclusion dans la structure et autorisent des spécifications sur les distances entre composants structurels.

**Manipulation de la séquence** : il s'agit de déterminer quels types d'opérateurs sont disponibles pour interroger sur la relation de séquence de la structure. L'interrogation des composants répétés (par exemple une suite de paragraphes) se fait par l'intermédiaire d'opérateurs sur les listes et plus généralement nous retrouvons l'introduction d'opérateurs tels que first, last ou la spécification d'une position dans la liste des composants.

MULTOS et les modèles basés sur les PAT expressions ignorent ce type d'interrogation alors que MAESTRO et le modèle de Navarro proposent des opérateurs de positionnement dans une liste de composants. Le langage de requête POQL intègre lui aussi la notion de position dans une liste de composants répétés (suite de paragraphes) ou dans un enchaînement de composants. Il faut noter qu'aucun opérateur spécifique n'est introduit pour le traitement de la séquence ce qui peut rendre complexe l'écriture de ces requêtes.

**Usage des Attributs** : nous déterminons deux types d'attributs, les *attributs locaux* et les *attributs globaux*. Les attributs globaux sont définis au niveau du document complet ou d'une collection de document alors que les attributs locaux sont définis au niveau des composants du document comme le propose la norme SGML. Il nous faut aussi savoir quelles sortes d'attributs sont disponibles pour chaque système et quels opérateurs pour les manipuler sont offerts.

Les PAT expressions n'admettent pas d'attributs tout comme le modèle de Navarro. Toutefois, ce dernier modèle permet de représenter certains attributs par d'autres voies: par l'intermédiaire d'une vue de présentation, il est possible de mettre en forme les attributs de présentation. Par exemple, il sera possible d'avoir les composants qui sont en italique avec la vue de présentation sans passer par un attribut. Le système MAESTRO propose uniquement des attributs textuels et globaux alors que MULTOS propose des attributs alphanumériques qui peuvent être confondus avec les composants structurels. Il en va de même avec le langage POQL qui autorise les attributs locaux sur les composants structurels des documents. De plus, POQL supporte la notion de variable d'attributs.

**Traitement du contenu des documents** : d'une part il faut savoir quels types de documents sont gérés, c'est-à-dire quels médias sont autorisés et d'autre part nous donnons les opérateurs permettant d'interroger sur le contenu de ces documents.

Excepté le système MULTOS qui traite simultanément des composants textuels et des images fixes, les autres systèmes comportent exclusivement des données textuelles. Les PAT expressions et le modèle de Navarro sont particulièrement dédiés aux documents

textuels. Un SGBD qui supporte le langage POQL peut accueillir plusieurs types de données mais les opérateurs d'interrogation sur le contenu se limitent aux attributs et aux parties textuelles des documents.

Seul les systèmes IOTA et MULTOS proposent de véritables fonctionnalités de systèmes de recherche d'informations, à savoir une phase d'indexation comportant une analyse du contenu sémantique des documents et une représentation de ce contenu conformément à un modèle. Ainsi qu'une phase d'interrogation ordonnant les réponses selon leur pertinence. Le système IOTA exploite la structure pour indexer les documents et propose une stratégie d'indexation reprise par la suite dans [Khe95], [LYYB96] ou [RF96].

**Autres caractéristiques** : les autres caractéristiques que nous pouvons relever dans ces systèmes concernent la granularité des réponses, c'est-à-dire sous quelle forme sont présentées les réponses: un document complet, un composant du document, une construction de composants du document. Nous nous intéressons aussi au fait que ces réponses soient ou non ordonnées en fonction de critères. Enfin, nous notons quelle est la structure qui a été privilégiée (structure physique, logique ou conceptuelle) ou si plusieurs sortes de structures sont combinées.

L'un des éléments qui nous semble le plus profitable pour ce type de systèmes est la forme de la réponse. Puisque nous considérons des documents structurés, c'est-à-dire des composants liés entre eux, il semble important que les réponses de ces systèmes profitent de cette notion de composants et puissent donc prendre la forme d'ensemble organisé ou non de composants. Il s'agit de définir la granularité des réponses. Le système MULTOS ne donne accès dans les réponses qu'à des documents complets. Il est impossible de spécifier dans la requête un choix de granularité. Les autres systèmes autorisent la spécification du type de réponse souhaité sauf IOTA qui restitue le composant qui correspond le mieux à la requête (d'un point de vue purement sémantique).

Afin d'assouplir l'indexation et de s'adapter à un maximum de types de documents, MULTOS définit la notion de composants actifs et de composants passifs. Les composants actifs peuvent être interrogés sur leur contenu alors que les composants passifs peuvent uniquement être interrogés sur leur existence.

Le modèle de Navarro propose des requêtes combinant de multiples vues du document. Ceci signifie qu'une requête peut mixer des conditions sur plusieurs sortes de structure, par exemple la structure logique et la structure de présentation. Les autres ne prennent qu'un unique type de structure dans leurs requêtes.

Ce récapitulatif laisse apparaître plusieurs éléments que nous souhaitons mettre en avant dans notre travail:

- Tout d'abord la capacité que doit avoir un système manipulant des documents structurés à accéder à n'importe quel élément de la structure. Cela semble impératif pour que l'utilisateur exploite la complexité de ce type de documents et qu'il n'ait pas une phase de recherche supplémentaire parmi les réponses pour accéder à l'élément de structure qu'il souhaite. Il faut donc lui fournir immédiatement cet élément.
- Ensuite, il semble aussi primordial d'utiliser l'ensemble des informations présentes dans ce type de document, que ce soit les informations instanciant les attributs des composants de structure, les relations de structure ou bien le contenu même des composants

de structure. Nous constatons ainsi que malgré les efforts réalisés pour rendre l'interrogation moins contraignante vis-à-vis des connaissances structurelles (chemins d'accès, variables d'attributs, etc), l'accès à un composant nécessite toujours une connaissance assez sûre de la structure sans que celle-ci ne semble totalement exploitée. Il est par exemple impossible d'accéder à un composant d'après un attribut particulier si cet attribut n'est pas défini sur ce composant et cela est toujours vrai même lorsque cet attribut est défini à un niveau inférieur ou supérieur dans la structure.

La puissance d'un tel système se juge donc selon nous dans sa capacité à rendre en tant que réponse tout élément *susceptible* de combler le besoin d'information de l'utilisateur. Or ce besoin ne s'exprime pas forcément en terme d'un chemin d'accès à un composant de structure mais plutôt en terme de la description de ce composant de structure. Nous avons vu que dans les systèmes présentés la structure est souvent privilégiée au détriment de la description des composants et de leur contenu. Nous souhaitons donc axer notre travail vers une meilleure prise en compte de la description des composants en conservant dans cette description le contexte structurel.

### 2.6.2 Discussion et orientation

La recherche sur les documents structurés doit combiner les aspects données et documents puisque d'une part nous souhaitons retrouver des éléments d'après leur contenu et d'autre part nous voulons aussi les retrouver d'après les caractéristiques de leur structure. La recherche de documents structurés apparaît alors comme une unification de la recherche de Données et de la recherche documentaire. Nous récapitulons ces remarques sur la recherche de documents structurés dans le tableau suivant :

| | Recherche de Données | Recherche Documentaire | *Recherche de Documents Structurés* |
|---|---|---|---|
| Modèle | Déterministe | Non Déterministe | *Déterministe et Non Déterministe* |
| Langage de requête | Déclaratif | Naturel | *Naturel, déclaratif* |
| Correspondance | Exacte | Partielle | *Partielle* |
| Les réponses | Correspondance | Pertinence | *Pertinence Structurelle* |

**Figure 2.9.** Position de la recherche de documents structurés par rapport à la recherche de données et à la recherche documentaire

Notre travail s'oriente donc plus précisément vers la description d'un modèle de représentation des documents structurés et la définition d'une stratégie de recherche adaptée à ce modèle. Il nous faut définir sur ce modèle comment se comporte les informations et comment celles-ci peuvent être utilisées. Nous reprenons ici les discussions possibles sur les points qui nous intéressent dans notre travail.

**Le Modèle** : les modèles de représentation des documents complexes ou structurés ad-

mettent des caractéristiques communes au niveau de la définition syntaxique de la structure. Ils proposent comme base la structure logique du document et admettent selon les types de documents des granularités plus ou moins fines. Les bases textuelles peuvent aller jusqu'à des granularités se situant au niveau des symboles du texte (caractères, ponctuation, etc).

Notre intention est de fournir un modèle de représentation pour des documents structurés comportant principalement du texte et des images fixes. Nous allons donc nous concentrer sur ces deux médias et fournir un modèle *unifié*. Ceci signifie que nous devons pouvoir représenter des parties textuelles et des images fixes en étant capable de comprendre chacun de ces éléments. Par ailleurs, nous ne devons pas écarter les possibles intégrations d'autres médias même si notre étude se limite ici aux textes et aux images.

L'autre aspect sur lequel nous devons nous attarder réside dans la représentation de la structure du document. Il faut tout d'abord déterminer quelle structure nous représentons puis décider comment nous la représentons. Pour déterminer la structure à représenter, il est possible de choisir une norme de représentation telle que SGML, ODA ou des normes dérivées (TEI) et proposer un modèle en accord avec ce choix. La seconde solution est de proposer un modèle qui soit indépendant des normes de représentation. C'est cette seconde solution que nous adoptons car nous travaillons sur l'accès aux documents structurés alors que les normes de représentation englobent certains aspects que nous n'abordons pas, présentation du document, gestion des documents, etc. Nous privilégions donc une représentation dans le cadre particulier de la recherche des documents structurés. Nous conservons une certaine indépendance vis-à-vis des normes tout en sachant que notre modèle doit pouvoir accueillir des documents conformes à ces normes. Ainsi, un document conforme à une définition SGML pourra être représenté dans notre modèle sans avoir à remodeler sa structure.

**Le Langage de requête** : les langages de requête des systèmes de recherche d'information privilégient une expression en langue naturelle alors que les langages de requêtes des systèmes de gestion de bases de données, et particulièrement ceux gérant des documents structurés, sont déclaratifs et ils réclament un utilisateur expert (connaissance du langage, connaissance de la structure des documents, etc). Les réflexions actuelles sur l'assouplisement des langages de requêtes déclaratifs en introduisant des variables sur les attributs ou bien des notions de chemins d'accès partiels sur les systèmes objets nécessitent une connaissance moindre sur la structure des documents et le nom des attributs de cette structure. Par ailleurs, l'intégration dans ces langages déclaratifs de modules d'interrogation plus *naturels* (par exemple recherche plein texte et recherche thématique) montre la nécessité d'un rapprochement entre les langages déclaratifs des systèmes de gestion de bases de données et les langages *naturels* des systèmes de recherche d'information.

Nous n'avons pas l'intention de fournir un nouveau langage de requêtes mais nous proposons d'orienter notre travail vers des fonctionnalités qu'un langage de requêtes sur des documents structurés devrait intégrer. Il s'agit par exemple d'une interrogation combinant des spécifications structurelles, des spécifications factuelles et des spécifications de contenu.

Les spécifications structurelles contiennent des contraintes sur les types d'éléments recherchés, la composition de ces éléments ou encore leurs positionnements respectifs dans

la structure syntaxique.

Les spécifications factuelles correspondent à des contraintes sur les attributs classiques des bases de données: une date, un auteur, un numéro de chapitre, etc.

Enfin, les spécifications de contenu abordent plus spécifiquement la problématique de la recherche d'information en permettant d'interroger sur le contenu du document, à savoir son contenu sémantique, sa thématique, ou encore les caractéristiques propres à chaque média.

**La correspondance** : la correspondance exacte des systèmes de gestion de bases de données connaît des avantages dont le principal réside dans la "certitude" de cerner exactement le contenu d'une base, mais elle ne peut être généralisée à tous les descripteurs d'un document complexe. Il faut donc choisir selon le type du descripteur le mode de correspondance le plus approprié.

L'objectif donné dans la discussion sur le modèle nous incite à considérer des attributs permettant de représenter et de décrire les caractéristiques de chaque composant structurel d'un document. Dans le domaine des bases de données, un domaine de définition est rattaché à chaque attribut. C'est ce domaine qui va fournir le mode de correspondance que nous mettrons en pratique sur l'attribut.

Nous estimons que la complexité de certaines informations qui représentent les composants nécessite des domaines de définition s'apparentant à des langages d'indexation tels qu'ils sont connus en recherche d'information. Un langage d'indexation est composé d'un vocabulaire de base et de règles de constructions des expressions de ce langage. Comme exemples de langages d'indexation en recherche d'informations, nous pouvons citer les ensembles de mots-clés, les graphes conceptuels [OP97a, Che92] ou encore les logiques descriptives [SS96]. Nous ne voulons pas par ailleurs proposer un mode de correspondance reposant sur un unique formalisme et préférons conserver une indépendance vis-à-vis de ces formalismes.

Nous admettons cependant que le mode de correspondance *partiel* est un élément essentiel pour estimer la pertinence des documents ou des parties de document et proposons donc une structure d'accueil tant pour des descripteurs utilisant une correspondance exacte que pour des descripteurs utilisant une correspondance partielle. Cette structure d'accueil sera définie au niveau du modèle de document et de la représentation des descripteurs.

**Les réponses** : dans le contexte particulier des documents structurés, nous devons particulièrement nous intéresser à cette facette de l'interrogation. Par rapport à l'approche classique des systèmes de recherche d'informations qui restitue un ensemble de documents *pertinents*, nous devons tenir compte de l'aspect complexe des documents. Ainsi, il est inutile de retourner un document complet lorsqu'une partie de ce document répond. Il est préférable, selon nous, de focaliser les réponses sur les parties *pertinentes* des documents.

Focaliser sur les parties pertinentes signifie restituer uniquement la ou les parties des documents qui correspondent aux critères de la requête. Toutefois, il faut se garder de déclarer que toute partie d'un document est une réponse potentielle. En effet, les documents structurés peuvent comporter des composants qui n'ont pas de sens soit parce

qu'ils prennent leurs sens dans un contexte particulier (au voisinage d'un autre composant par exemple), soit pour des raisons de tailles (une encyclopédie est un document structuré mais peut-on considérer qu'on a renseigné un utilisateur en lui retournant comme réponse une encyclopédie complète?). Par ce dernier exemple, nous voulons établir que toute partie de document ou tout document n'est pas forcément intéressant en tant que réponse pour l'utilisateur. Nous l'avons montré dans PRIME-GC où nous avons écarté de la structure d'indexation un élément intermédiaire qui est la liste des séries d'images médicales. Nous proposons de retrouver directement une image ou bien le compte-rendu médical qui contient la liste des séries d'images mais pas une série d'images. Il s'agit de délimiter a priori l'intérêt des types des éléments de la structure pour ne pas introduire dans les réponses des éléments qui, pris hors du contexte structurel, ne prennent pas tout leur sens.

Nous devons donc définir parmi les composants d'un document structuré quels sont ceux qui sont des réponses potentielles. Ceci doit se faire en accord avec la notion d'indexation définie au niveau du modèle de document structuré. Nous adoptons l'approche classique de la recherche d'informations qui consiste à dire qu'une réponse potentielle est un élément qui admet un descripteur.

# Deuxième partie

# Modèle de représentation du document structuré

*Nous avons consacré la première partie de ce document à la description de modèles de représentation et de recherche de documents. Nous nous sommes plus particulièrement intéressés à la problématique de la recherche de documents structurés par deux types de systèmes : les Systèmes de Gestion de Bases de Données et les Systèmes de Recherche d'Informations.*
*Dans cette partie, nous allons établir un modèle pour la représentation des documents structurés accueillant des données textuelles et des images fixes. Ce modèle, particulièrement élaboré pour la recherche de documents structurés, va s'articuler autour des caractéristiques élémentaires de ce type de documents, â savoir leurs relations de structure.*
*Dans un premier chapitre, nous étudions et caractérisons les relations de structure du document textuel. Pour cela, nous reprenons les travaux de François Paradis [Par96] et détaillons les relations de structure selon deux types d'organisation : la structure syntaxique et la structure sémantique.*
*Dans le chapitre suivant, nous formulons une descriptions structurée des images fixes. Cette description comporte d'une part la définition des niveaux de représentation des éléments présents dans une image et d'autre part les relations qui lient ces éléments.*
*Enfin, nous concluons cette partie par la présentation de notre modèle de représentation des documents structurés dans le chapitre 5. Ce modèle est articulé autour d'une intégration des relations décrites dans les deux chapitres précédents (chapitres 3 et 4). Le document structuré est alors vu comme une arborescence d'éléments typés comportant des liens transversaux ainsi que des descripteurs d'éléments conformément au modèle. Ce chapitre se conclut par un exemple mettant en oeuvre le modèle et montrant les éléments de formalisation que nous avons proposés.*

# Chapitre 3

# Les relations de structure du document textuel

## 3.1 Les Types de structure

Dans le cadre de notre travail, nous nous intéressons particulièrement à la modélisation du document prenant en compte les aspects structurels et sémantiques. Afin d'établir un modèle de représentation articulé autour de la structure du document, nous présentons les différents types de structure pouvant intervenir dans un document textuel.

Cette première partie consiste en une présentation informelle nous permettant de donner une première définition de chacun des types de structure. Nous nous concentrons sur les types de structure indépendants de l'environnement de présentation du document, c'est à dire ne prenant pas en compte les aspects liés au mode de présentation. Nous nous intéressons particulièrement à trois types de structure: la structure linguistique, la structure logique et la structure de discours.

### 3.1.1 La structure linguistique

La structure linguistique reflète l'organisation des symboles qui constituent le document textuel. En fait, tout élément qui apparaît dans un document textuel peut être considéré comme un symbole. Sa position dans le document est déterminée par la structure linguistique.

**Définition 1 (Structure Linguistique)** *Organisation des informations de contenu textuel (ou structure syntaxique des symboles) du document textuel.*

Le contenu textuel correspond à l'ensemble des symboles qui apparaîssent dans le document textuel, c'est-à-dire à l'ensemble des chaînes de caractères qui forment le texte. Ces

symboles peuvent être caractérisés par des informations linguistiques d'ordre lexical, morphologique, syntaxique, etc. De plus ces symboles se regroupent pour former d'autres symboles. On a donc différents niveaux de granularité pour les symboles d'un document textuel.

Prenons un exemple afin d'identifier les symboles, leurs niveaux de granularité et les informations d'ordre linguistique attachés à ces symboles.

**Exemple 1 (Eléments de la structure linguistique)**
Apprendre le C++ aujourd'hui.

Dans l'exemple 1, nous considérons des symboles de différents niveaux de granularité. Le niveau atomique des symboles est le *mot* et l'agrégation des symboles de niveau atomique forme la *phrase*. Les symboles de cette phrase peuvent être caractérisés par des informations linguistiques (cf. classification des informations linguistiques de Paradis [Par96] (section 3.1.1)). Une information linguistique de type *lexical* indique que le mot "Apprendre" est un verbe et une information linguistique de type *vocabulaire* indique que le mot "C++" est un terme technique.

Le contenu textuel d'un document est rendu lisible (ou compréhensible) grâce à l'organisation linguistique des symboles qui forment le contenu textuel. Nous pouvons distinguer clairement deux types de relation qui permettent d'établir cette organisation. D'une part, nous avons vu qu'il existait des symboles de granularités différentes. Les symboles de plus haut niveau de granularité sont obtenus par la composition des symboles de plus bas niveaux. Il existe donc une relation qui permet de *composer* les symboles. D'autre part, les symboles sont *ordonnés* selon le sens de lecture. Cet ordonnancement dépend de la langue. Si dans les langues romanes, on lit de gauche à droite, la lecture se faite de droite à gauche dans les langues arabes. Mis à part le cas particulier de la langue du document, les informations d'ordre linguistique sont généralement indépendantes de l'environnement et modélisables de manière générique à l'aide d'une grammaire [Par96].

La structure linguistique définit donc l'organisation des symboles du texte et les caractérise par des informations d'ordre linguistique. Elle organise et regroupe ces symboles pour les rendre compréhensibles en s'appuyant sur des règles dépendant de la langue utilisée dans le document.

Nous faisons reposer l'organisation linguistique sur deux relations : la *relation de composition* et la *relation de séquence*.

### 3.1.2 La structure logique

La structure logique du document textuel reflète l'organisation explicite d'abstractions logiques représentant des parties de document. Les abstractions logiques correspondent à des *entités d'informations*. Les abstractions logiques sont assimilables à des entités d'informations car nous considérons que ces abstractions existent indépendamment les unes des autres du fait qu'elles contiennent suffisamment d'informations pour être compréhensibles.

Ces abstractions logiques sont le plus souvent caractérisées par un type qui détermine leur position dans l'organisation logique du document. Toutefois un type d'entité logique prend son sens uniquement lorsque le *contexte du document* est entièrement décrit. Le contexte du

document est son type logique, c'est-à-dire la définition complète des types d'entités d'information qui composent le document. Nous pouvons ici faire référence à une définition de type de documents telle qu'elle est proposée dans SGML (Standard Generalized Markup Language) via une DTD (Document Type Definition) [Her90, San94].

**Définition 2 (Structure Logique)** *Organisation explicite d'entités d'informations qui représentent des parties du document textuel.*

L'organisation de la structure logique est *explicite* et elle est définie a priori. Toutefois cette organisation des entités d'informations ne repose pas sur un ensemble de règles générales toujours applicables quel que soit le document. En fait, pour chaque type de document, il est nécessaire de définir un ensemble de règles. Par exemple un document de type LIVRE est composé d'un TITRE-de-LIVRE et d'un ou plusieurs CHAPITRES, eux-même composés d'un TITRE-de-CHAPITRE et d'un ou plusieurs SOUS-CHAPITRES.

Le découpage du document en entités d'informations répond à des besoins exprimés tant par l'auteur que par le lecteur. En effet l'auteur a besoin d'un cadre structuré pour s'exprimer. Ce cadre lui est fourni par la structure logique qu'il donne à son document (ou qui lui est imposée). Le lecteur, quant à lui, peut évoluer plus aisément dans un document dont la structure logique est explicite même si celle proposée par l'auteur ne lui correspond pas. Cette organisation offre toutefois des repères qui vont faciliter sa lecture.

Dans sa définition classique, l'organisation logique du document textuel est une arborescence dont les noeuds (aussi appelés éléments logiques de structure) correspondent à des parties du document. La relation entre ces noeuds est une relation de composition structurelle, c'est-à-dire une relation qui exprime qu'une partie (le noeud source) est composée d'une autre partie (le noeud cible).

Pour expliciter ce qu'est la structure logique du document, nous pouvons nous référer au lanage de balisage SGML [Her90, San94]. Chaque élément de la structure logique du document est marqué par une balise et chaque document se réfère à un modèle de structure logique appelé DTD (cf. exemple 2). Ce modèle de structure indique quels types d'abstractions logiques sont présents dans le document et comment ces types peuvent se composer dans le document. La définition, le marquage et l'identification des abstractions logiques correspondent à la définition syntaxique de la structure d'une classe de documents, conforme à une même DTD. Le langage SGML intègre par ailleurs un niveau sémantique par l'ajout d'attributs spécifiques à chaque type d'éléments de structure (dans l'exemple 2 nous ne spécifions pas ces attributs). Dans la définition de la structure logique des documents SGML nous retrouvons la notion de composition d'éléments structurels mais aussi la notion de séquence d'éléments structurels qui détermine les enchaînements possibles d'éléments.

**Exemple 2 (Exemple de DTD)**

```
<! DOCTYPE livre [
<! ELEMENT livre – (titre-de-livre, chapitre+) >
<! ELEMENT chapitre – (titre-de-chapitre, sous-chapitre*) >
<! ELEMENT sous-chapitre – (titre, corps-texte) >
...] >
```

La structure logique d'un document considère les parties de document comme des entités

d'information et les organise selon deux relations : une *relation de composition* et *une relation de séquence*. Cette structure se réfère à un contexte, le type du document (la DTD en SGML), qui fournit les types d'entités qui vont constituer le document.

### 3.1.3 La structure de discours

La structure linguistique et la structure logique mettent en relation des parties du document en se basant sur des règles syntaxiques définies a priori : règles d'organisation des symboles du document textuel dépendant de la langue du document et règles d'organisation des entités d'informations dépendant du type du document. Nous considérons qu'il s'agit d'organisations syntaxiques. L'organisation qui rend le document compréhensible est quant à elle reflétée par ce qui est appelée la *structure de discours* du document.

**Définition 3 (Structure de Discours)** *Organisation des idées présentes dans le document textuel.*

L'organisation syntaxique ne garantit pas qu'un document soit compréhensible pour le lecteur, seule l'organisation des idées, c'est-à-dire la structure de discours, peut être le garant de la compréhension du document (même si cela n'est pas toujours suffisant, c'est nécessaire).

L'auteur d'un document a un *but* lorsqu'il rédige un document. Pour faciliter la compréhension du document, l'auteur organise ses idées. L'expression des idées s'exprime à travers des parties du document textuel. Au niveau du discours, une partie de document textuel (mot, groupe de mots, phrase, paragraphe, chapitre, etc) qui exprime une ou plusieurs idées ou thèmes est appelée *segment de discours*.

Le document est donc découpé en segments qui correspondent à l'expression d'une ou plusieurs idées. Ces segments ne sont pas indépendants les uns des autres. La première des relations qui lient ces segments est une relation qui exprime la composition des segments, c'est-à-dire la composition des idées exprimées par les segments de discours. Cette relation est classiquement appelée *relation de dominance*. Si un segment de discours, que nous notons $s_1$ domine un autre segment de discours, noté $s_2$, cela signifie que les idées exprimées dans $s_2$ participent aux idées exprimées dans $s_1$.

En d'autres termes, un segment de discours représente une partie du document et peut être décrit par un ensemble d'idées (ou thèmes). Les idées présentes dans cet ensemble sont celles qui peuvent être extraites par une analyse du contenu sémantique de la partie textuelle du document représenté. On peut donc assimiler le segment de discours à un ensemble d'idées. La relation de dominance entre un segment $s_1$ et $s_2$ ($s_1$ domine $s_2$) peut alors être assimilée à la relation d'inclusion suivante : $s_2 \subseteq s_1$; l'ensemble des idées de $s_2$ est inclus dans l'ensemble des idées de $s_1$.

Dans la structure de discours d'autres relations peuvent être mises en évidence. Parmi celles-ci, la *relation d'intention* qui déclare explicitement dans le texte du document qu'une idée est présente dans une autre partie (un segment de discours).

**Exemple 3 (Relation d'intention)**
Dans le chapitre 2 nous décrivons le surf des neiges (snowboard).

Dans l'exemple 3, l'auteur déclare explicitement que le thème du chapitre 2 est le surf des neiges, c"est-à-dire que ce thème particulier fera partie de l'ensemble des thèmes associés au chapitre 2. Cette relation permet de renforcer la présence d'un thème ou d'une idée dans un segment. On peut ici noter que les segments de discours représentent la plupart du temps des parties du document textuel qui appartiennent *soit à la structure linguistique, soit à la structure logique du document.*

## 3.2 Définition formelle des structures

Nous abordons maintenant la représentation des trois types de structure présentés : structure linguistique, structure logique et structure de discours. Nous définissons chacune des relations propres à ces structures ainsi que leurs particularités.

### 3.2.1 Definition de la structure linguistique

La structure linguistique reflète l'organisation des symboles du contenu textuel qui peuvent être caractérisées par des informations d'ordre linguistique. Nous identifions deux relations qui permettent d'organiser ces symboles ainsi que leurs propriétés : la relation de composition et la relation de séquence.

La *relation de composition* décrit une arborescence des symboles du document textuel. Les feuilles de cette hiérarchie sont les symboles atomiques (ici, le mot).

**Exemple 4 (Relation de Composition Linguistique)**
Le groupe de mots "Apprendre le C++" est obtenu par composition des mots : "Apprendre", "le" et "C++".

La *relation de séquence* décrit l'enchaînement des symboles, tant au niveau des symboles atomiques (un mot) qu'au niveau des symboles composés. Cette relation de séquence indique l'ordre de lecture des symboles.

**Exemple 5 (Relation de Séquence Linguistique)**
Dans le groupe de mots "Apprendre le C++", la relation de séquence établit que le mot "Apprendre" précède le mot "le" qui précède lui-même le mot"C++".

Paradis a établi une classification des symboles de la structure linguistique dans [Par96]. Cette classification prend en compte les caractérisations d'ordre linguistique qui peuvent s'appliquer à des symboles de différentes granularités. Nous nous limitons à une classification plus restrictive comportant uniquement les mots, les groupes de mots, les phrases et les paragraphes afin de rester générique et de rester indépendant d'un domaine d'application particulier.

#### a) Définition formelle

Formellement, la structure linguistique d'un document est une arborescence dont les noeuds sont des éléments linguistiques. Cette arborescence est obtenue par une relation de

composition et les éléments sont ordonnés par une relation de séquence. De plus, nous typons ces éléments selon leur granularité. Nous en donnons un exemple dans la figure 3.1.

La structure linguistique est donc décrite à partir de deux classes d'information : le *type de structure*, noté $ST_{ling}$, et l'*instance du document*, notée $D_{ling}$.

$$ST_{ling} = (TYPE_{ling}, \preceq_{t\_ling})$$

**$\mathbf{TYPE_{ling}}$** est l'ensemble des types d'éléments linguistiques.

**$\mathbf{\preceq_{t\_ling}}$** est la relation sur les types d'éléments linguistiques définis par $TYPE_{ling}$. Elle définit la notion de type et de sous-type sur cet ensemble. Il s'agit d'une relation binaire interne sur l'ensemble des types d'éléments linguistiques : $\preceq_{t\_ling} \subset TYPE_{ling} \times TYPE_{ling}$.

$$D_{ling} = (OS_{ling}, \prec_{comp\_ling}, \prec_{seq\_ling}, type_{ling})$$

**$\mathbf{OS_{ling}}$** est l'ensemble des *éléments linguistiques*. Un élément linguistique représente un symbole ou une agrégation de symboles du document textuel.

**$\mathbf{\prec_{comp\_ling}}$** est la *relation de composition linguistique* décrivant la composition hiérarchique dans la structure linguistique. La relation de composition linguistique est une relation binaire interne sur l'ensemble des éléments linguistiques : $\prec_{comp\_ling} \subset OS_{ling} \times OS_{ling}$.

**$\mathbf{\prec_{seq\_ling}}$** est la *relation de séquence linguistique* décrivant l'enchaînement des éléments de la structure linguistique, c'est-à-dire le parcours linéaire du texte. Il s'agit d'une relation binaire interne sur l'ensemble des éléments linguistiques : $\prec_{seq\_ling} \subset OS_{ling} \times OS_{ling}$.

**$\mathbf{type_{ling}}$** est la fonction surjective d'assignation d'un type d'élément linguistique pris dans $TYPE_{ling}$ à un élément linguistique de $OS_{ling}$.

$$type_{ling} : OS_{ling} \rightarrow TYPE_{ling}$$

### b) La relation de composition linguistique

La relation de composition linguistique est une relation binaire qui décrit la relation *père/fils* de l'arborescence structurelle du document. Cette relation met en jeu des éléments linguistiques pris dans l'ensemble $OS_{ling}$.

La relation $o_i \prec_{comp\_ling} o_j$ signifie que $o_j$ est un composant (fils) direct de $o_i$ dans l'arborescence structurelle linguistique. Les éléments linguistiques $o_i$ et $o_j$ appartiennent à l'ensemble des éléments qui décrivent un document textuel : $o_i, o_j \in OS_{ling}$.

Dans l'exemple de la figure 3.1 nous décrivons les relations de composition linguistique pour la phrase La révolution informationnelle. Elle est composée directement de 2 éléments linguistiques : $o_2$ et $o_1$ qui représente le groupe de mots La révolution informationnelle, lui-même composé de trois éléments linguistiques feuilles : $o_{11}, o_{12}, o_{13}$.

La relation de composition linguistique a les propriétés suivantes : non réflexive, non transitive et asymétrique (voir annexe A, section A.1.1).

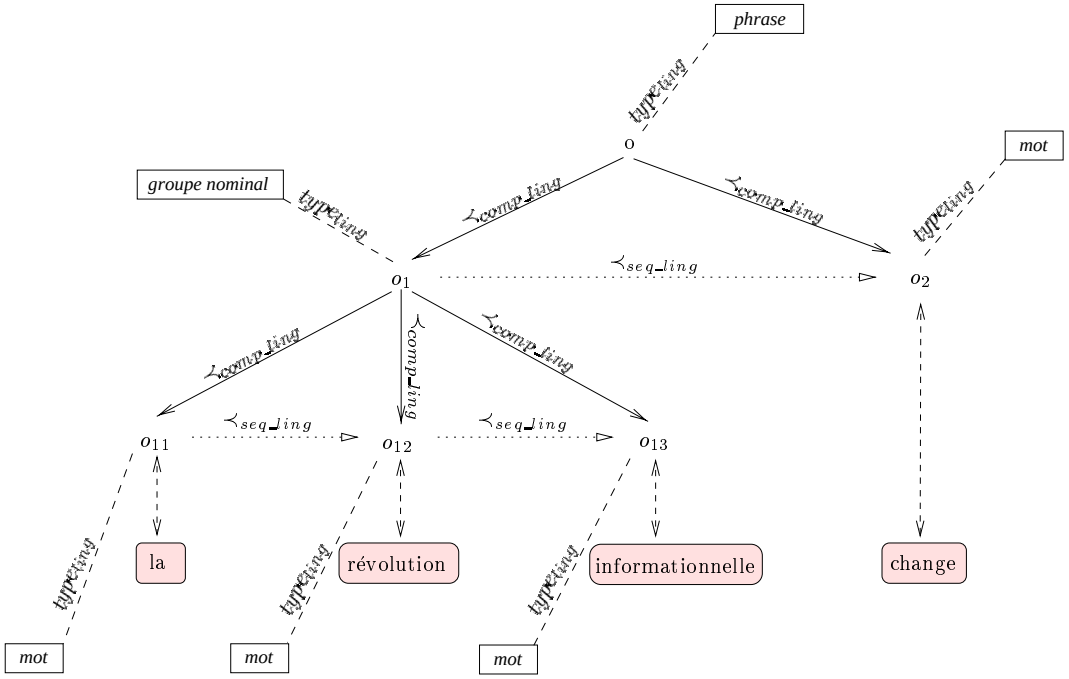



**Figure 3.1.** Relation de Composition et de Séquence de la Structure linguistique

### c) La relation de séquence linguistique

La relation $o_i \prec_{seq\_ling} o_j$ avec $o_i \in OS_{ling}$ et $o_j \in OS_{ling}$ signifie que l'élément linguistique $o_j$ est le successeur direct de l'élément linguistique $o_i$ dans la séquence. Pour que la relation de séquence entre deux éléments linguistiques soit valide, les éléments $o_j$ et $o_i$ doivent avoir le même père dans la composition structurelle définie par la relation de composition linguistique ($\prec_{comp\_ling}$). Nous exprimons cette contrainte de la manière suivante :

$o_i \prec_{seq\_ling} o_j$
**ssi**
$\exists\, o \in OS_{ling}$ tel que $o \prec_{comp\_ling} o_j$ et $o \prec_{comp\_ling} o_i$

Cette relation de séquence sur les éléments linguistiques possède les propriétés suivantes : non réflexive, non transitive et asymétrique (voir annexe A, section A.1.2).

### d) Les types d'éléments linguistiques

Nous définissons un ensemble de types noté $TYPE_{ST\_ling}$ qui caractérisent les éléments linguistiques selon leur granularité.

La relation sur l'ensemble des types linguistiques a le sens suivant : $t_i \preceq_{t\_ling} t_j$ avec $t_i \in TYPE_{ST\_ling}$ et $t_j \in TYPE_{ST\_ling}$ signifie que le type $t_i$ est un *sous-type au sens*

*structurel* de $t_j$, c'est-à-dire qu'un élément linguistique de type $t_i$ peut être un fils d'un élément linguistique de type $t_j$. Nous dirons alors que le type $t_i$ est un super-type de $t_j$ et que le type $t_j$ est un sous-type de $t_i$. Cette relation décrit les possibilités de composition *directe* d'un type d'élément par un autre type d'élément.

Les propriétés de la relation sur les types d'éléments linguistiques

La reflexivité sur la relation entre les types d'éléments linguistiques signifie qu'il est possible d'avoir au niveau instance un élément linguistique de type $t$ composé d'un autre élément de même type $t$ (par exemple, un paragraphe composé d'un autre paragraphe). Cependant ceci n'est pas toujours possible (une phrasse ne peut pas être composée d'une autre phrase). Nous ne donnons donc pas de propriété de réflexivité sur cette relation.

### e) Caractéristiques de la structure linguistique

Nous donnons quelques éléments caractéristiques de la structure linguistique fournis par les relations de cette structure.

**élément racine linguistique** : un élément racine linguistique est un élément linguistique qui n'a pas d'élément linguistique père. Cet élément racine est unique, il est noté $o_{racine\_ling}$ et il est décrit de la manière suivante :

$$\exists\,!\; o_{racine\_ling} \in OS_{ling} \textbf{ vérifiant } \nexists\, o \in OS_{ling} \text{ tel que } o \prec_{comp\_ling} o_{racine\_ling}$$

L'élément racine, que nous avons noté $o_{racine\_ling}$, contient chacun des autres éléments linguistiques du document.

**éléments feuilles linguistiques** : un élément feuille linguistique est un élément linguistique qui n'admet pas d'élément fils. Cet élément feuille est décrit de la manière suivante :

$$\forall o \in OS_{ling}, o \text{ est un élément feuille } \textbf{ssi } \nexists o_i \in OS_{ling} \text{ tel que } o \prec_{comp\_ling} o_i$$

Nous définissons l'ensemble des éléments linguistiques feuilles d'un document textuel et nous notons cet ensemble $OS_{f\_ling}$ : $OS_{f\_ling} \subseteq OS_{ling}$.

$$\forall o \in OS_{f\_ling}, o \text{ est un élément feuille}$$

**Ordonnancement des éléments** : les éléments de la structure linguistique sont tous comparables entre eux dans l'espace linéaire de lecture du document. Ils appartiennent tous au même espace de représentation à une dimension et peuvent être comparés selon les 7 relations de Allen [All83].

**Type racine linguistique** : un type racine linguistique est un type linguistique de l'ensemble $TYPE_{ling}$ qui n'a pas de super-type dans cet ensemble. Nous considérons un type racine unique. Ce type identifie une classe de documents. Nous le notons $t_{racine\_ling}$ et il est décrit de la manière suivante :

$$\exists\,!\; t_{racine\_ling} \in TYPE_{ling}$$
$$\textbf{vérifiant } \nexists\, t_i \in TYPE_{ling} \text{ tel que } t_i \prec_{t\_ling} t_{racine\_ling}$$

### 3.2.2 Définition de la structure logique

La structure logique est l'organisation selon une relation de composition et une relation de séquence d'entités d'informations correspondant à des parties de document. Cette organisation prend la forme d'une arborescence dont les noeuds sont les entités et dont la racine représente le document complet. A la différence de la structure linguistique, la définition des entités dépend du contexte, c'est-à-dire du type des documents considérés. Une norme telle que SGML exprime ce contexte par la définition d'une DTD à laquelle les documents doivent se conformer.

Dans notre travail, nous ne cherchons pas à vérifier qu'un document se conforme à des règles, mais nous souhaitons représenter les éléments de structure du document pour caractériser le document et ensuite pouvoir les utiliser lors d'une session de recherche d'information. Nous spécifions la structure logique d'un document textuel par la définition de deux relations : la *relation de composition logique* et la *relation de séquence logique*.

La *relation de composition* décrit l'agrégation des entités (éléments de structure) dans l'arborescence. Dans l'exemple 6, une entité (le chapitre) se compose d'autres entités (titre de chapitre et section). La DTD de SGML décrit explicitement quels types d'éléments fils sont autorisés pour un type donné, ainsi que leur ordonnancement et une indication sur le nombre d'apparition de chacun. Pour notre part, nous spécifions uniquement des contraintes sur la composition des éléments et leur ordonnancement sans contraindre le nombre d'apparitions. Nous lions les contraintes sur la composition à la définition des types d'entités.

**Exemple 6 (Relation de Composition Logique)**
Un chapitre est composé d'un titre de chapitre et d'une ou plusieurs sections. Une section est elle-même composée d'un titre de section et d'un ou plusieurs paragraphes.

La relation de séquence décrit l'enchaînement (l'ordonnancement) des entités, c'est-à-dire leur sens de lecture dans le document.

**Exemple 7 (Relation de Séquence Logique)**
Un titre de chapitre est suivi d'une section, qui peut elle-même être suivie d'une autre section.

Par la suite, nous donnons une formalisation de ces deux relations et leurs propriétés, ainsi qu'une définition des types d'entités. L'ensemble de ces informations est défini dans la structure logique du document textuel, notée $ST_{log}$.

Nous pouvons déjà noter de fortes similarités avec la définition de la structure linguistique que nous avons précédemment décrite. Nous résumons ces similarités dans le tableau suivant :

#### a) Définition formelle

La structure logique est une arborescence d'entités logiques. L'arborescence d'entités logique est formée à partir de la relation de composition logique et l'ordonnancement des entités est donnné par la relation de séquence. Les entités logiques sont typées et une relation de sous-typage est décrite sur l'ensemble des types d'entités.

Nous décrivons la structure logique du document textuel avec les deux classes d'information suivantes : le *type de structure logique*, noté $ST_{log}$, et l'*instance du document*, notée

| | Structure Linguistique | Structure Logique |
|---|---|---|
| Eléments | agrégation de symboles | entités logiques |
| Composition | relation de composition linguistique | relation de composition logique |
| Séquence | relation de séquence linguistique | relation de séquence logique |
| Types des éléments | types linguistiques | types logiques |
| Relations sur les types | relation contraignant la composition des types linguistiques | relation contraignant la composition et la séquence des types logiques |

**Figure 3.2.** Structure linguistique et logique : une comparaison informelle

$D_{log}$.

$$ST_{log} = (TYPE_{log}, \preceq_{t\_log}, \preceq_{tseq\_log})$$

**$\mathbf{TYPE_{log}}$** est l'ensemble des *types d'entités logiques.*

**$\preceq_{\mathbf{t\_log}}$** est une relation binaire interne sur les types d'entités logiques pris dans $TYPE_{log}$ qui définit les sous-types et super-types dans l'ensemble des types d'entités logiques, $TYPE_{log}$, c'est-à-dire les compositions possibles entre types d'entités : $\preceq_{t\_log} \subset TYPE_{log} \times TYPE_{log}$.

$\preceq_{tseq\_log}$ est une relation binaire interne sur les types d'entités logiques pris dans $TYPE_{log}$ qui définit les enchaînements possibles entre types d'entités : $\prec_{tseq\_log} \subset TYPE_{log} \times TYPE_{log}$.

$$D_{log} = (OS_{log}, \prec_{comp\_log}, \prec_{seq\_log}, type_{log})$$

**$\mathbf{OS_{log}}$** est l'ensemble des *entités logiques.* Une entité logique représente une partie du document.

**$\prec_{\mathbf{comp\_log}}$** est la *relation de composition logique* décrivant la composition des entités logiques dans la structure logique.

Cette relation de composition logique est une relation binaire interne définie sur l'ensemble des entités logiques : $\prec_{comp\_log} \subset OS_{log} \times OS_{log}$. Si deux entités admettent cette relation, on dira que l'un est le père (et réciproquement le fils) structurel direct de l'autre.

**$\prec_{\mathbf{seq\_log}}$** est la *relation de séquence logique* qui décrit l'enchaînement séquentiel des entités logiques de la structure logique, c'est-à-dire le parcours linéaire des entités logiques. Cette relation de séquence logique est une relation binaire définie sur l'ensemble des entités logiques : $\prec_{seq\_log} \subset OS_{log} \times OS_{log}$.

Elle est uniquement définie entre entités frères, c'est-à-dire entre entités ayant même père structurel.

**type$_{log}$** est une fonction surjective d'assignation d'un type d'entité logique pris dans $TYPE_{log}$ à un entité logique de $OS_{log}$. Chaque entité de l'ensemble $OS_{log}$ est donc typée.

$$type_{log} : OS_{log} \rightarrow TYPE_{log}$$

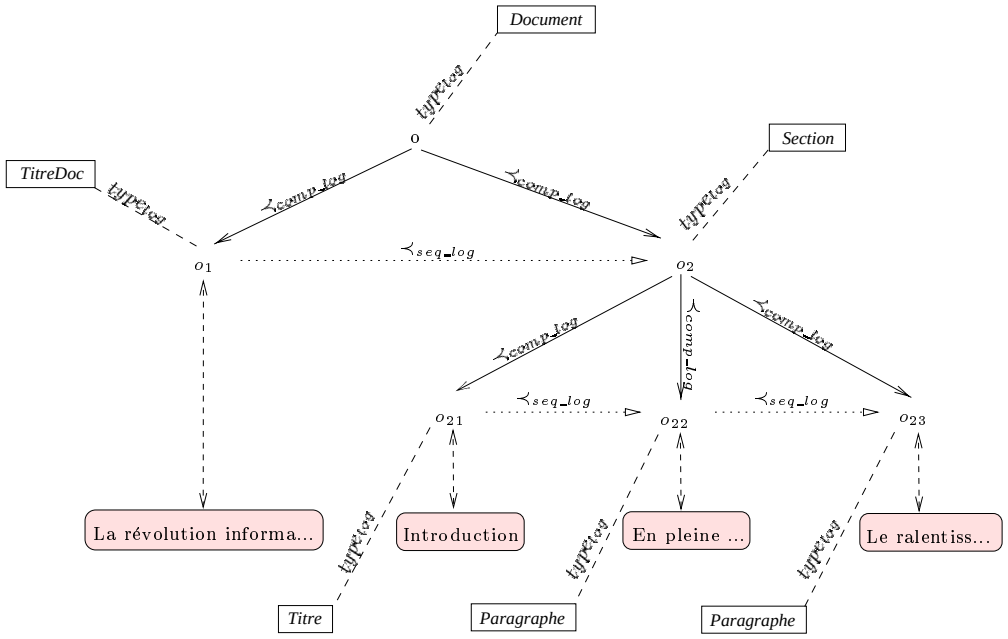



**Figure 3.3.** Relation de Composition et de Séquence de la Structure Logique

### b) La relation de composition logique

La relation de composition logique, notée $\prec_{comp\_log}$, est une relation binaire qui décrit la relation de composant/composé entre les entités dans l'arborescence structurelle du document.

La relation $o_i \prec_{comp\_log} o_j$, avec $o_i, o_j \in OS_{log}$, signifie que l'entité logique $o_j$ est un composant (fils direct) de $o_i$ dans l'arborescence structurelle logique du document textuel.

La relation de composition logique $\prec_{comp\_log}$ doit vérifier certaines propriétés pour que la structure logique soit bien une arborescence, et pour que la relation s'établisse uniquement entre une entité père et son fils direct. La relation de composition logique est non réflexive, non transitive et asymétrique (voir annexe A, section A.2.1).

### c) La relation de séquence logique

Comme nous l'avons dit précédemment, la relation de séquence logique décrit les enchaînements séquentiels entre entités logiques. Deux entités sont donc en relation si elles se succèdent directement. Nous ajoutons toutefois la condition supplémentaire suivante: les deux entités

doivent aussi être soeurs, c'est-à-dire avoir le même père structurel. Cela rend cette relation de séquence logique similaire à celle définie sur les éléments linguistiques.

La relation $o_i \prec_{seq\_log} o_j$, avec $o_i, o_j \in OS_{log}$, signifie donc que l'entité logique $o_j$ est un successeur direct de l'entité logique $o_i$, et que $o_j$ et $o_i$ ont même père dans l'arborescence structurelle définie par la relation de composition logique ($\prec_{comp\_log}$).

Nous retrouvons les mêmes propriétés de non réflexivité, de non transitivité et d'asymétrie pour la relation de séquence logique (voir annexe A, section A.2.2). Ces propriétés permettent de vérifier que l'enchaînement des entités de la structure logique ne provoquent pas de cycles.

### d) Les types d'entités logiques

Cet ensemble et les relations sur cet ensemble reprennent une partie des informations décrites lors de la définition des entités d'une DTD de SGML (Exemple 2), c'est-à-dire la définition des types d'entités logiques d'une classe de documente et les compositions et enchaînements possibles entre ces types.

L'ensemble des types dépend du contexte d'applications, ce qui n'est pas le cas des types linguistiques. Dans le cadre d'une application portant sur la représentation d'articles scientifiques, cet ensemble aura la forme suivante:

$TYPE_{log} = \{Article, Titre, Resume, Section, Titre\_Section, \dots\}$.

Deux relations sont définies sur cet ensemble: d'une part la relation $\preceq_{t\_log}$ qui définit les compositions possibles et d'autre part la relation $\prec$ *tseq_log* qui définit les enchaînements possibles.

*Propriétés de la relation* $\preceq_{t\_log}$

La relation $t_i \preceq_{t\_log} t_j$ avec $t_i, t_j \in TYPE_{ST\_log}$ signifie que le type $t_i$ est un sous-type au sens structurel de $t_j$, c'est-à-dire qu'une entité logique de type $t_i$ peut être un composant d'une entité logique de type $t_j$.

Les propriétés de cette relation sont identiques à celles de la relation sur les types linguistiques (section 3.2.1).

*Propriétés de la relation* $\prec_{tseq\_log}$

La relation $t_i \prec_{tseq\_log} t_j$ signifie que dans le document, une entité de type $t_i$ peut être suivie d'une entité de type $t_j$. Afin de permettre que deux entités de même type puissent se suivre dans la séquence, nous ne donnons pas de propriété de réflexivité sur cette relation.

Nous pouvons noter que, pour deux entités qui se suivent dans la séquence, c'est-à-dire $o_i \prec_{seq\_log} o_j$, la relation suivante doit être vérifiée: $type_{log}(o_i) \prec_{tseq\_log} type_{log}(o_j)$.

### e) Caractéristiques de la structure logique

Les éléments caractéristiques de la structure logique sont similaires à ceux que nous avons définis pour la structure linguistique dans la section e) (page 76). Nous avons donc une *entité racine* qui représente le document complet et qui est notée $o_{racine}$. L'ensemble des *entités feuilles* de la structure logique est noté $OS_{f\_log}$ et il est inclus dans l'ensemble $OS_{log}$. Comme

pour la structure linguistique, les entités de la structure logique sont toutes comparables entre elles dans l'espace linéaire de lecture du document. Enfin, nous considérons les *types racines logiques* et les *types feuilles logiques* pris dans l'ensemble des types de la structure logique, $TYPE_{log}$. L'ensemble des types feuilles de la structure logique est noté : $TYPE_{max\_log}$.

### 3.2.3 Définition de la structure de discours

Avant de donner plus précisément les définitions des relations de la structure de discours, nous revenons un instant sur les termes que nous employons dans cette définition et le sens que nous leur accordons afin d'éviter toute ambiguïté.

Tout d'abord la structure de discours d'un document s'attache à l'organisation des idées ou des thèmes d'un document. Ces idées ou ces thèmes correspondent à ce qui est compris d'un document ou d'une partie de document. Dans la structure de discours, les éléments de base sont les *segments de discours*, comparable aux entités logiques de la structure logique et aux symboles de la structure linguistique. Un segment de discours correspond à une partie du document et il est caractérisé par la ou les idées exprimées dans cette partie de document. Nous considérons le segment de discours comme un ensemble d'idées (nous confondons idées et thèmes) qui se réfèrent à une partie du document.

Lors de la définition de la structure de discours, section 3.1.3, nous avons mis en évidence deux relations : la *relation de dominance* et la *relation d'intention*. Nous revenons maintenant sur ces relations.

Pour comprendre la progression thématique d'un texte et donc comprendre la relation de dominance entre segments de discours, considérons un article (Exemple 8) qui traite des activités hivernales dans les stations de ski. Un premier segment de discours va traiter du ski de piste et un second traite du surf des neiges. S'il existe un segment de discours (par exemple l'article complet) qui agrège ces deux segments, il va traiter du ski de piste **et** du surf des neiges.

**Exemple 8**
**1-** L'activité principale dans les stations alpines est le ski de piste. Cette activité est maintenant pratiquée intensivement par des milliers d'amateurs de tout âge depuis plus de trente ans.
**2-** Une activité en plein développement depuis maintenant 5 ans dans les stations alpines est le surf des neiges, aussi appelé snowboard. Cette activité a attiré une clientèle jeune souhaitant découvrir de nouvelles sensations.

Les idées vont s'agréger selon la relation de dominance, c'est-à-dire que si un segment de discours domine un second segment, les idées de ce second segment sont *incluses* dans celles du premier. Autrement dit, les idées du segment dominé participent aux idées du segment dominant. La relation de dominance va donc décrire une arborescence dont les noeuds sont les segments de discours, c'est-à-dire les idées exprimées dans le texte. La relation de dominance correspond à la composition de ces idées et la racine de l'arborescence correspond aux idées exprimées dans l'ensemble du document.

La seconde relation qui nous intéresse, la relation d'intention, est une relation transversale à la relation de dominance. Considérons l'exemple 9 :

**Exemple 9 (Relation d'intention)**
Dans la partie 2, nous avons rappelé l'essor pris par le **surf des neiges**.

Dans cette phrase, l'auteur déclare explicitement l'intention de la partie 2. On retrouve donc dans le texte les intentions (les idées) que l'auteur a voulu développer. Il déclare explicitement le *but* d'un segment : le surf des neiges. Cette relation, appelée ici *relation d'intention*, permet d'enrichir la connaissance du segment cible : la partie 2. Même si l'idée le surf des neiges était déjà présente dans le segment de discours représentant la partie citée, cette relation est en quelque sorte une preuve que cette idée est centrale dans cette partie. La présence de cette relation permet de donner une importance plus grande à l'idée déclarée explicitement.

Nous considérons donc deux types de relation dans la structure de discours : la *relation de dominance* qui décrit la structure hiérarchique des idées du texte et la *relation d'intention* qui indique explicitement le but (les idées) d'un segment de discours.

### a) Définition formelle

La définition de la structure de discours que nous donnons décrit un graphe de segments de discours qui supporte deux relations : la relation de dominance et la relation d'intention. La seule prise en compte de la relation de dominance fournit une arborescence des segments de discours dont la racine est le document complet. Les relations d'intention sont des relations transversales entre les éléments de cette arborescence.

Les informations que nous avons retenues pour décrire la structure de discours du document textuel, les segments de discours, les relations de dominance et d'intention, sont décrites dans la classe d'information suivante :

$$D_{disc} = (OS_{disc}, \prec_{domine}, \preceq_{intention})$$

- $\mathbf{OS_{disc}}$ : les segments de discours d'un document sont représentés par un ensemble d'éléments de discours.
- $\prec_{\mathbf{domine}}$ : relation de dominance qui exprime la composition des idées du document textuel. Cette relation est une relation binaire interne définie sur $OS_{disc}$ : $\prec_{domine} \subset OS_{disc} \times OS_{disc}$.

  Si la relation de dominance entre deux éléments de discours $o_i$ et $o_j$ existe, c'est-à-dire $o_i \prec_{domine} o_j$, alors on dit que l'élément de discours $o_i$ domine l'élément de discours $o_j$.
- $\preceq_{\mathbf{intention}}$ : relation d'intention qui décrit les idées d'un élément de discours à l'aide des idées exprimées dans un autre élément de discours. Cette relation est une relation binaire interne définie sur l'ensemble des éléments de discours : $\preceq_{intention} \subset OS_{disc} \times OS_{disc}$.

  Si la relation d'intention entre deux éléments de discours $o_i$ et $o_j$ existe, c'est-à-dire $o_i \prec_{intention} o_j$, alors on dit que l'élément de discours $o_i$ a pour intention l'élément de discours $o_j$.

Nous pouvons noter que pour la structure de discours, il n'existe pas de type de structure pré-définie comme pour les structures linguistiques et logiques.

### b) La relation de dominance

La hiérarchie des idées dans un document textuel est décrite par la relation de dominance que nous notons $\prec_{domine}$. La notation $o_i \prec_{domine} o_j$ avec $o_i, o_j \in OS_{disc}$ signifie que les idées exprimées dans l'élément de discours $o_j$ participent pleinement aux idées de l'élément de discours $o_i$. L'élément de discours $o_i$ reprend les idées de tous les éléments de discours qu'il domine.

Afin de décrire une arborescence d'éléments de discours, la relation de dominance notée $\prec_{domine}$ est assimilée à une relation de composition et doit vérifier les propriétés suivantes: non réflexive, non transitive et asymétrique (voir Annexe A, section A.3.1).

### c) La relation d'intention

Elle représente les déclarations explicites d'intention dans un document. La notation $o_i \preceq_{intention} o_j$ avec $o_i, o_j \in OS_{disc}$ signifie que l'élément $o_i$ a pour intention les idées exprimées dans l'élément $o_j$. Il ne faut donc pas confondre ici les éléments qui permettent d'identifier les relations d'intention et les éléments sur lesquels portent effectivement cette relation.

Comme le montre François Paradis dans [Par96], ces relations peuvent être extraites du texte à l'aide d'indices linguistiques: verbes descriptifs, termes de localisation, ... etc. Par ailleurs la relation d'intention peut être étendue de manière à l'appliquer à d'autres caractéristiques que les idées exprimées dans un texte. Prenons l'exemple suivant:

**Exemple 10 (Relation d'intention - attributs)**
La section 2 a pour auteur **Echenoz**.

Dans cet exemple précis, il ne s'agit pas des idées exprimées dans le texte mais de l'auteur du segment représenté par la section 2. Cette relation peut donc s'appliquer à différentes sortes d'attributs.

La relation d'intention est réflexive, non transitive et asymétrique (voir Annexe A, section A.3.2). La relation d'intention décrit les relations explicites qui apparaissent dans le texte, c'est-à-dire que l'auteur déclare explicitement certaines idées d'un segment du discours. La transitivité de cette relation signifierait que si l'élément $o_2$ est explicité par le contenu de l'élément $o_1$ et que l'élément $o_3$ est explicité par le contenu de l'élément $o_2$ alors l'élément $o_3$ est explicité par le contenu de l'élément $o_1$. Ceci ne peut être vérifié.

$\forall o_1, o_2, o_3 \in OS_{disc}$
$o_1 \preceq_{intention} o_2$ et $o_2 \preceq_{intention} o_3$ n'implique pas $o_1 \preceq_{intention} o_3$

L'asymétrie se montre assez aisément en prenant l'exemple 9 (page 81). Il est clair que l'élément de discours associé à la partie surf des neiges n'a pas pour intention la partie 2.

## 3.3 Relations entre les structures

Nous avons défini de manière indépendante trois types de structure présents dans le document textuel. Le modèle de document textuel doit les intégrer. La structure linguistique et la structure logique du document ont des propriétés analogues et nous les désignons sous le terme générique de *structure syntaxique*. Cette structure reflète l'organisation de parties du document textuel sans se préoccuper de leur contenu sémantique.

Par la suite nous décrivons la *structure sémantique* du document textuel en fonction de la structure de discours, c'est-à-dire selon l'organisation des idées du document. Ces deux structures syntaxiques et sémantiques ne sont pas indépendantes et nous mettons en évidence les dépendances entre leurs éléments.

### 3.3.1 Structure Syntaxique

Un document textuel quelconque peut être décrit selon sa structure linguistique et/ou sa structure logique. Des caractéristiques similaires apparaissent à travers les formalisation de ces deux structures. Ainsi les relations décrivant l'organisation des structures logiques et linguistiques sont semblables et les caractérisations des éléments reposent sur un typage avec des notions de sous-types et super-types pour contraindre les compositions selon des règles de construction syntaxique. Nous définissons la structure syntaxique de la manière suivante:

**Définition 1 (Structure Syntaxique)** *La structure syntaxique est l'organisation des symboles du document textuel qui repose sur la composition des symboles formant une arborescence et sur l'enchaînement séquentiel des symboles fournissant l'ordonnancement et donc le sens de la lecture.*

La structure linguistique décrit l'agrégation et l'enchaînement des symboles du document textuel. Elle représente une arborescence d'éléments linguistiques dont l'élément racine représente le document complet et dont les autres éléments représentent des symboles de plus faibles granularités. A un niveau d'abstraction différent, la structure logique du document textuel décrit l'organisation d'entités dont la granularité est plus grande que celle des éléments linguistiques. Là encore, il s'agit d'une arborescence dont l'élément racine représente le document complet et dont les autres éléments représentent les entités logiques. Le type de ces entités est contraint par le type du document.

Les relations d'organisation des éléments sont similaires dans les deux structures, elles ont les mêmes fonctionnalités et les mêmes propriétés. Ces structures se différencient par le fait que la structure linguistique est générique alors que la structure logique est contrainte par le type du document considéré. Une autre différence provient de la granularité des éléments logiques et linguistiques. On peut en effet considérer que les entités logiques sont composés de symboles du document textuel, c'est à dire qu'un élément logique est composé d'éléments linguistiques. Nous montrons ceci dans la figure 3.4, en définissant la structure logique du document comme une vue étendue de l'élément racine de la structure linguistique.

Dans la figure 3.4, on combine les deux structures linguistiques et logiques. Dans cette arborescence, la racine de la structure linguistique a été remplacée par l'arborescence de la structure logique. L'élément racine linguistique représentait le document complet qui est do-

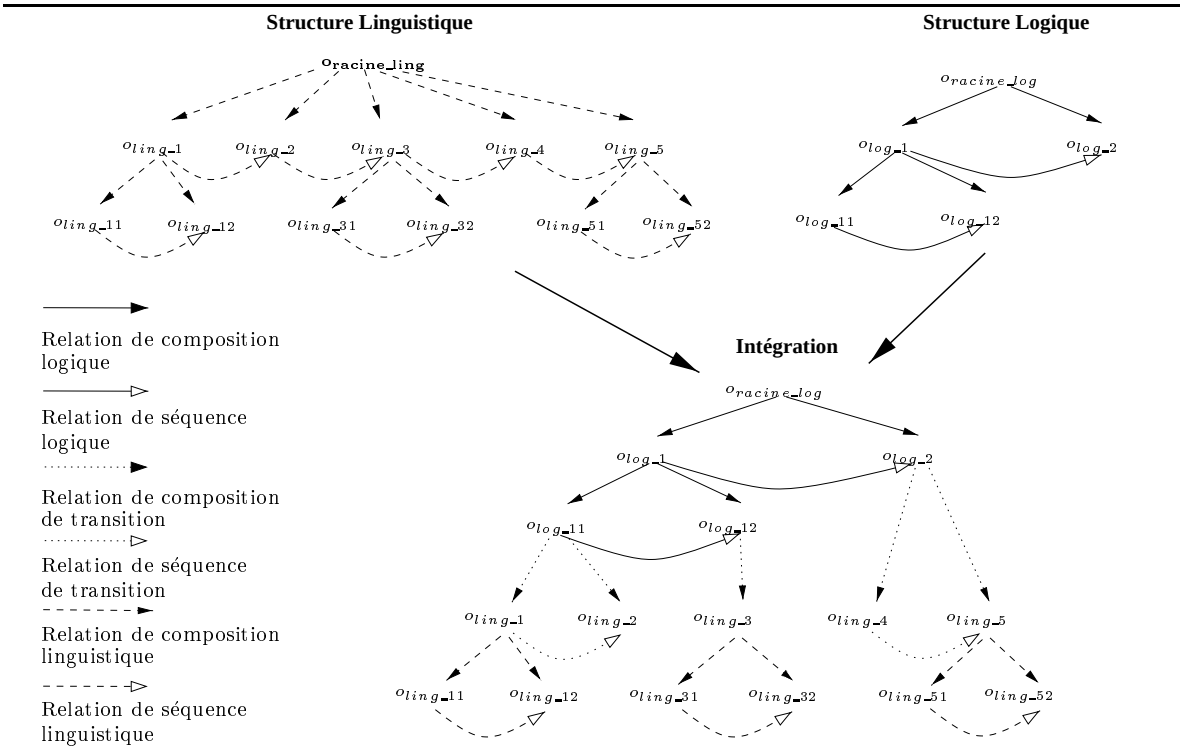



**Figure 3.4.** Lien entre structure linguistique et logique : structure syntaxique

rénavant représenté par l'élément racine logique. D'autre part, les éléments logiques feuilles sont mis en relation avec les éléments linguistiques qui étaient fils directs de la racine linguistique. Les relations de séquence entre ces éléments ont été modifiées par rapport à celles de la structure linguistique originale puisqu'ils ne sont plus fils d'un même élément dans cette nouvelle arborescence.

Par exemple, les éléments linguistiques $o_{ling\_2}$ et $o_{ling\_3}$ avaient même père dans la structure linguistique et la relation de séquence linguistique entre ces deux éléments existait. Dans la structure syntaxique complète, ces deux éléments ne sont plus frères et ne peuvent donc plus admettre de relation de séquence même si leur disposition dans le document textuel n'a pas été modifiée. Le fait qu'ils se succèdent dans le document est établi par les relations provenant de leurs parents.

Il apparaît donc que la combinaison des structures linguistiques et logiques pour former la structure syntaxique du document textuel requiert la définition de nouvelles relations :

1. Définition d'un nouvel ensemble d'éléments linguistiques excluant la racine.
2. Possibilité de composition d'éléments logiques par des éléments linguistiques de ce nouvel ensemble.
3. Nouvelle définition des relations de séquences sur les éléments linguistiques fils directs de l'élément racine linguistique.

4. Définition de la compatibilité des types des éléments logiques feuilles avec des types linguistiques.

Nous reprenons chacun de ces points en définissant les éléments qui permettent d'établir la combinaison des structures linguistiques et logiques et qui permettent de construire la structure syntaxique du document.

1. Nous redéfinissons donc l'ensemble des éléments linguistiques de la structure syntaxique comme un sous-ensemble de l'ensemble des éléments linguistiques. Nous notons cet ensemble $OS^*_{ling}$.

   $$OS^*_{ling} = OS_{ling} \Leftrightarrow \{o_{racine\_ling}\}$$

2. La relation de composition linguistique permet uniquement des compositions d'éléments linguistiques avec des éléments linguistiques. De même, la relation de composition logique permet de composer des éléments logiques avec d'autres éléments logiques.

   Nous définissons donc une relation de composition, que nous notons $\prec_{comp\_trans}$ et que nous appelons *relation de composition de transition*, qui permet la composition des éléments feuilles de la structure logique par des éléments linguistiques provenant de l'ensemble $OS^*_{racine\_ling}$. Cet ensemble décrit l'ensemble des éléments racines de $OS^*_{ling}$.

   Cette relation de composition est donc définie sur les ensembles $OS_{f\_log}$ et $OS^*_{racine\_ling}$ et a les propriétés de *non réflexivité*, de *non transitivité* et d'*asymétrie*.:

   $$\prec_{comp\_trans} \subseteq OS_{f\_log} \times OS^*_{racine\_ling}$$

   La structure logique est une extension de la structure linguistique. Donc chaque élément logique feuille doit être rattaché à au moins un élément linguistique :

   $$\forall o \in OS_{f\_log}, \exists\, o_i \in OS^*_{racine\_ling} \text{ tel que } o \prec_{comp\_trans} o_i$$

3. Considérons maintenant l'ensemble $OS^*_{ling}$. La non prise en compte de la racine $o_{racine\_ling}$ entraîne la non validité des relations de séquence sur ses éléments fils. En effet, une relation de séquence existe entre deux éléments si ces deux éléments sont frères dans l'arborescence linguistique. Comme nous avons supprimé leur père de l'ensemble $OS^*_{ling}$, les relations de séquence entre ces éléments n'existent pas lorsqu'on applique la relation $\prec_{seq\_ling}$ sur $OS^*_{ling}$.

   Nous introduisons une relation de séquence définie sur les éléments de l'ensemble $OS^*_{racine\_ling}$, appelée *relation de séquence de transition* et notée $\prec_{seq\_trans}$. La contrainte des éléments frères se fera avec la relation de composition que nous venons de définir entre entités logiques feuilles et éléments linguistiques de l'ensemble $OS^*_{racine\_ling}$. La relation $\prec_{seq\_trans}$ se définit par :

   $$\prec_{seq\_trans} \subseteq OS^*_{racine\_ling} \times OS^*_{racine\_ling}$$

Cette relation est déduite de la relation de séquence linguistique sur l'ensemble $OS_{ling}$, c'est-à-dire que si deux éléments linguistiques fils de la racine linguistique sont mis en relation par la relation de séquence linguistique, notée $\prec_{seq\_ling}$, et qu'ils ont le même père parmi les éléments logiques feuilles alors ces deux éléments sont des successeurs directs selon la relation de séquence notée $\prec_{seq\_trans}$.

$$o_i \prec_{seq\_trans} o_j \textbf{ si } \begin{cases} o_i \prec_{seq\_ing} o_j \\ \textbf{et} \\ \exists\, o_{log} \in OS_{f\_log} \text{ tel que } \begin{cases} o_{log} \prec_{comp\_trans} o_i \\ \textbf{et} \\ o_{log} \prec_{comp\_trans} o_j \end{cases} \end{cases}$$

4. La relation $\prec_{seq\_trans}$ doit prendre en compte des contraintes liées aux types des éléments qui sont mis en relation, c'est-à-dire que nous ne pouvons pas mettre en relation n'importe quel type d'entité logique avec n'importe quel type d'élément linguistique. Nous devons donc introduire une relation décrivant le sous-typage des types logiques feuilles par des types linguistiques.

   Du fait de la prise en compte de l'ensemble $OS^*_{ling}$, l'ensemble des types linguistiques est réduit à un ensemble, noté $TYPE^*_{ling}$ défini par :

   $$TYPE^*_{ling} = TYPE_{ling} \Leftrightarrow \{t_{racine\_ling}\}$$

   Cette relation de sous-typage, notée $\prec_{t\_lien}$ et appelée *relation de sous-typage de transition*, se définit de la manière suivante :

   $$\prec_{t\_lien} \subseteq TYPE_{max\_log} \times TYPE^*_{ling}$$

   La notation $t_i \prec_{t\_lien} t_j$ avec $t_i \in TYPE_{max\_log}$ et $t_j \in TYPE_{ling}$ signifie que le type logique $t_i$ est un super-type du type linguistique $t_j$, c'est-à-dire qu'un élément logique de type $t_i$ peut être composé d'un élément linguistique de type $t_j$.

   Tout comme les relations $\prec_{t\_ling}$ et $\prec_{t\_log}$ définies respectivement sur les ensembles $TYPE_{ling}$ et $TYPE_{log}$, cette relation est *non réflexive*, *non transitive* et *asymétrique*.

Pour établir la structure syntaxique du document textuel nous avons considéré la structure logique comme une extension de la structure linguistique et nous avons défini les relations permettant d'établir la transition entre les arborescences logiques et linguistiques.

### 3.3.2 Structure Sémantique

Si la structure syntaxique décrit l'organisation des symboles du document textuel, la structure sémantique exprime l'organisation du contenu sémantique de ce document. Dans le premier cas, certaines règles liées à la définition de la structure syntaxique (définition des types de symboles et de leurs compositions) permettent de dire si un document est bien

formé. Dire qu'un document est bien formé d'un point de vue sémantique revient à dire que ce document est compréhensible, c'est-à-dire que la disposition des idées développées dans ce document a permis au lecteur d'atteindre le but fixé par l'auteur ou son propre but.

Nous allons voir selon la définition de la structure sémantique les éléments qui s'y rattachent et comment il est possible d'établir un lien avec la structure syntaxique.

**Définition 2 (Structure Sémantique)** *La structure sémantique est l'organisation du contenu sémantique du document textuel.*

La structure de discours que nous avons décrite dans la partie 3.2.3 reflète l'organisation des idées du document textuel ce qui correspond bien à la définition de la structure sémantique. Nous avons retenu deux relations pour décrire cette organisation, la relation de dominance assimilée à une relation de composition des idées et la relation d'intention. Nous avons établi ces relations entre des éléments du discours, c'est-à-dire des segments de discours représentant une ou plusieurs idées d'un morceau du document.

Pour établir le lien entre la structure sémantique (structure de discours) et la structure syntaxique, il nous faut définir les parties de document qui participent à l'élaboration du discours. En d'autres termes, il faut établir une relation entre les éléments de la structure syntaxique et ceux de la structure sémantique.

**Hypothèse 1** *Tout élément de la structure syntaxique du document textuel participe à l'élaboration du discours de celui-ci.*

Cette première hypothèse nous permet de considérer que chaque élément de la structure syntaxique est un élément de la structure sémantique, c'est-à-dire qu'un élément quelconque de la structure syntaxique peut être vu sous la forme des idées qui participent à cet élément. La représentation d'un élément de la structure syntaxique dans la structure sémantique est donc un ensemble d'idées, en d'autres termes un élément de la structure sémantique.

**Hypothèse 2** *Tout élément de la structure de discours correspond à un élément de la structure syntaxique du document.*

Cette seconde hypothèse, couplée à la première, établit la bijection entre les éléments de la structure syntaxique et ceux de la structure sémantique. Pour tout élément de la structure syntaxique, il existe un unique élément de la structure sémantique qui lui est associé, et réciproquement. Cette hypothèse est restrictive car il est toujours possible de trouver dans un document textuel, des thèmes qui apparaissent par exemple sur un paragraphe et sur la moitié du suivant.

Nous adoptons cette approche, basée sur ces deux hypothèses, afin de garder un lien fort entre la manière dont le document a été structuré et les idées véhiculées par ce document. De plus des travaux basés sur la segmentation des documents selon les thèmes de celui-ci [HP93], ou bien des expérimentations de recherche d'information portant sur des documents découpés en segments de texte [Wil94, Cal94], ont montré que cette relation entre éléments de structure définis par l'auteur (phrases, paragraphes, sections) et thèmes émergeants est forte. De plus, cette solution nous permet d'éviter des recouvrements de segments, c'est-à-dire des segments de discours correspondant à des segments de texte non définis comme éléments syntaxiques.

### 3.3.3 La relation d'intention et les relations de référence

Nous introduisons ici les relations de référence, c'est-à-dire les relations liant des éléments du texte en ne suivant ni les relations de composition ni celles de séquence. Ces relations de référence lient des éléments linguistiques et des éléments logiques. C'est pour cette raison que nous les introduisons après l'intégration de ces éléments dans la structure syntaxique.

Nous notons une relation de référence $\prec_{ref}$ et la définissons de la manière suivante :

$$\prec_{ref} \subset OS^*_{ling} \cup OS_{log} \times OS^*_{ling} \cup OS_{log}$$

La notation $o_i \prec_{ref} o_j$, avec $o_i \in OS^*_{ling} \cup OS_{log}$ et $o_j \in OS^*_{ling} \cup OS_{log}$, signifie que l'élément $o_i$ réfère l'élément $o_j$. L'élément $o_i$ sera appelé *élément source* et l'élément $o_j$ sera appelé *élément destination*.

D'après notre définition, une relation de référence lie un élément linguistique ou logique à un élément linguistique ou logique. La phrase suivante en est un exemple.

**Exemple 11 (Relation de Référence)**
Dans le chapitre 2 nous décrivons le surf des neiges (snowboard).

La phrase complète de l'exemple 11 correspond à un élément linguistique et il réfère l'élément logique identifié par les termes "chapitre 2". Nous avons déjà utilisé ce même exemple pour décrire une relation d'intention (exemple 3 page 72). Nous posons comme hypothèse que chaque relation d'intention peut être considérée comme une relation de référence au niveau de la structure syntaxique du document. Nous notons cependant qu'une relation d'intention a certaines particularités provenant du type d'information qu'elle précise : il peut s'agir comme nous l'avons présenté initialement des idées du texte mais aussi des différents types d'attributs qui caractérisent un élément de structure (auteur, date, etc.).

Une étude particulière des relations de référence, et parmi celles-ci des relations hypertextuelles, permettraient de dire si la réciproque de l'hypothèse précédente (chaque relation de référence peut être considérée comme une relation d'intention au niveau de la structure sémantique) se vérifie. Il ne s'agit pas là du propos de notre étude et nous nous contentons ici de l'hypothèse 1.

Toutefois, il est possible de reconnaître les relations de référence qui peuvent être considérées comme des relations d'intention en anlaysant le contenu de l'élément source. Si ce contenu contient des informations permettant de préciser la description de l'élément cible alors cette relation de référence correspond à une relation d'intention au niveau sémantique. Nous utilisons cette caractéristique pour différencier les relations de référence ayant une relation d'intention qui leur correspond des autres relations de référence. Nous les noterons de la manière suivante : $\prec^a_{ref} \subset \prec_{ref}$.

La relation $\prec^a_{ref}$ est une relation de référence qui a une relation d'intention qui lui correspond au niveau de la structure sémantique. La caractérisation du contenu est donné par $a$ qui précise le type d'information. Ces relations vérifient la propriété suivante :

$\forall o_i, o_j \in OS^*_{ling} \cup OS_{log}$
**Si** $o_i \prec^a_{ref} o_j$ **Alors** $o_i \prec_{intention} o_j$

Sur l'exemple 11 dans lequel c'est la description du contenu du "chapitre 2" qui est précisée, nous l'indiquons en notant cette relation $\prec_{ref}^{contenu}$.

### 3.3.4 Synthèse

L'introduction de la structure syntaxique et de la structure sémantique du document textuel nous ont permis d'établir plus clairement les relations entre les différentes structures que nous avons définies. Ces relations reposent bien entendu sur certaines hypothèses:

- D'une part, nous considérons les structures linguistique et logique comme des structures établissant la syntaxe structurelle du document. La structure logique est alors une extension de la structure linguistique par l'introduction d'abstractions offrant des éléments de granularité plus grande (chapitre, section, ...). Nous avons donc redéfini certaines relations et les ensembles d'éléments et de types d'éléments afin de pouvoir donner une structure syntaxique unifiant la structure linguistique et la structure logique.
- D'autre part, nous établissons les hypothèses qui nous permettent de considérer que chaque élément syntaxique est un élément du discours, et réciproquement. Ceci, ramené au domaine de la recherche d'informations, signifie que tout élément de la structure syntaxique peut être indexé, et a donc un contenu sémantique qui lui est particulier.

Nous avons aussi montré à travers les notions de structure syntaxique et de structure sémantique du document structuré que chaque relation de la structure sémantique a une relation qui lui correspond au niveau de la structure syntaxique. La relation de dominance repose sur la relation de composition de la structure syntaxique et la relation d'intention repose sur les relations de référence. Dans notre travail, nous n'avons pas introduit au niveau sémantique de relation correspondant à la relation de séquence de la structure syntaxique. Il nous semble en effet que si des relations de dominance ou d'intention permettent de préciser les descriptions des éléments structurels, une relation reposant sur la relation de séquence permettrait plutôt de montrer les coupures ou les changements de thème dans le document [HP93]. Nous avons choisi délibérément de ne pas intégrer ce type de relation.

Nous abordons maintenant la formalisation du modèle intégré de représentation de documents textuels structurés en décrivant les intégrations entre les différentes structures.

## 3.4 Modèle intégré de texte

Nous donnons une définition du modèle intégré de représentation des relations de structure des documents textuels ainsi que quelques caractéristiques de ce modèle.

### 3.4.1 Intégration et Définition

Comme le propose par exemple la norme SGML, la structure d'un document textuel est décrite à partir des classes d'informations suivante: le *type de structure*, noté $ST_{text}$, et l'*instance du document* se conformant à ce type de structure, noté $D_{text}$.

*Le type de structure*

$$ST_{text} = (TYPE_{ST}, \preceq_{tcomp\_text}, \preceq_{tseq\_text})$$

**$TYPE_{ST}$** est *l'ensemble des types d'éléments structurels* s'appliquant aux éléments de la structure syntaxique. L'ensemble des types d'éléments structurels, noté $TYPE_{ST}$, est en fait l'union de l'ensemble des types d'origine linguistique démuni de son type minimal, noté $TYPE^*_{ling}$, avec l'ensemble des types d'origine logique, noté $TYPE_{log}$.

$$TYPE_{ST} = TYPE^*_{ling} \cup TYPE_{log}$$

**$\preceq_{tcomp\_text}$** est la *relation définie sur l'ensemble des types d'éléments structurels* qui décrit les relations de composition possibles entre les types d'éléments structurels. Il s'agit d'une relation binaire interne sur l'ensemble des types d'éléments structurels : $\preceq_{tcomp\_text} \subset TYPE_{ST} \times TYPE_{ST}$.

Cette relation provient en fait des relations définies sur l'ensemble des types linguistiques, sur l'ensemble des types logiques et sur la liaison entre ces deux ensembles.

$$t_i \preceq_{tcomp\_text} t_j \Leftrightarrow \begin{cases} t_i \preceq_{t\_ling} t_j & \textbf{si } t_i, t_j \in TYPE^*_{ling} \\ t_i \preceq_{t\_trans} t_j & \textbf{si } t_i \in TYPE_{log} \text{ et } t_j \in TYPE^*_{ling} \\ t_i \preceq_{t\_log} t_j & \textbf{si } t_i, t_j \in TYPE_{log} \end{cases}$$

Les propriétés de cette relation sont dirigées par les types comme nous l'avons déjà indiqué dans la définition de la structure linguistique et de la structure logique.

**$\preceq_{tseq\_text}$** est la relation définie sur l'ensemble des types d'éléments structurels qui décrit les enchaînements possibles d'éléments logiques. Il s'agit d'une relation binaire interne sur l'ensemble des types d'éléments structurels : $\preceq_{tseq\_text} \subset TYPE_{ST} \times TYPE_{ST}$.

Nous décrivons cette relation uniquement sur les types d'éléments logiques car elle n'apporte rien au niveau des éléments linguistiques. L'ensemble $TYPE_{ST}$ est donc réduit à l'ensemble $TYPE_{log}$ et il s'agit alors d'une relation binaire interne sur $TYPE_{log}$ : $\preceq_{tseq\_text} \subset TYPE_{log} \times TYPE_{log}$.

*L'instance du document structuré*

$$D_{text} = (OS_{text}, \prec_{comp\_text}, \prec_{seq\_text}, \preceq_{ref}, type_{st})$$

**$OS_{text}$** est *l'ensemble des éléments représentant les différentes parties d'un document textuel.* Cet ensemble est défini à partir des ensembles d'éléments linguistiques ($OS_{ling}$), d'éléments logiques ($OS_{log}$) et d'éléments du discours ($OS_{disc}$).

$$OS_{text} = OS^*_{ling} \cup OS_{log} \cup OS_{disc}$$

Les ensembles $OS^*_{ling}$ et $OS_{log}$ sont disjoints. De plus, les hypothèses émises à la section 3.3.2 (page 87) nous permettent d'avoir la relation liant $OS_{disc}$ avec $OS^*_{ling}$ et $OS_{log}$ :

$$OS_{disc} = OS^*_{ling} \cup OS_{log}$$

**$\prec_{\text{comp\_text}}$** est la *relation de composition* permettant d'agréger les différentes parties d'un document textuel. Cette relation de composition est une relation binaire interne sur l'ensemble des éléments textuels : $\prec_{comp\_text} \subset OS_{text} \times OS_{text}$.

Cette relation de composition est définie à partir de relations provenant des structures syntaxiques et sémantiques. Si les deux éléments mis en relation sont des éléments linguistiques, il s'agit de la relation de composition linguistique. Si l'un des deux éléments mis en relation est un élément logique et l'autre un élément linguistique, il s'agit de la relation de composition entre éléments feuilles de la structure logique et éléments linguistiques. Si les deux éléments mis en relation sont des éléments logiques alors il s'agit de la relation de composition logique.

$$o_i \prec_{comp\_text} o_j \Leftrightarrow \begin{cases} o_i \prec_{comp\_ling} o_j & \textbf{si } o_i, o_j \in OS^*_{ling} \\ o_i \prec_{comp\_trans} o_j & \textbf{si } o_i \in OS_{f\_log} \text{ et } o_j \in OS_{racine\_ling} \\ o_i \prec_{comp\_log} o_j & \textbf{si } o_i, o_j \in OS_{log} \end{cases}$$

Dans la description ci-dessus, nous donnons la définition de la relation de composition $\prec_{comp\_text}$ en fonction des éléments de structure syntaxique. Au niveau sémantique, cette relation est assimilée à la relation de dominance de la structure de discours, c'est-à-dire que pour chaque élément mis en relation le contenu de l'un participe au contenu de l'autre.

La notation $o_i \prec_{comp\_text} o_j$ signifie :

- *niveau syntaxique* : l'élément $o_i$ est le père structurel de l'élément $o_j$, c'est-à-dire que l'élément $o_i$ est composé de l'élément $o_j$.
- *niveau sémantique* : le contenu sémantique de l'élément $o_j$ participe au contenu sémantique de l'élément $o_i$, c'est-à-dire que le contenu sémantique prend en compte le contenu sémantique de $o_j$.

Les propriétés de cette relation de composition sont celles qui sont communes aux relations de composition syntaxiques et à la relation de dominance : *non réflexive*, *non transitive*, et *asymétrique*.

**$\prec_{\text{seq\_text}}$** est la *relation de séquence* qui décrit l'enchaînement des éléments au niveau de la structure syntaxique. C'est une relation binaire interne sur l'ensemble des éléments textuels : $\prec_{seq\_text} \subset OS_{text} \times OS_{text}$.

Elle décrit l'enchaînement des éléments linguistiques et logiques et elle reprend les spécifications des relations de séquence respectives. Ces relations sont uniquement définies

entre des éléments frères et décrivent uniquement le successeur (prédécesseur) direct d'un élément.

Si les deux éléments mis en relation sont des éléments linguistiques, il s'agit soit de la relation de séquence linguistique, soit de la relation de séquence de transition. Si les deux éléments mis en relation sont des éléments logiques alors il s'agit de la relation de séquence logique. On peut noter qu'il est impossible de mettre en relation de séquence un élément linguistique et un élément logique. En effet, la relation de composition définie précédemment ne permet pas d'avoir deux éléments frères provenant de deux structures différentes.

$$o_i \prec_{seq\_text} o_j \Leftrightarrow \begin{cases} o_i \prec_{seq\_log} o_j & \textbf{si } o_i, o_j \in OS_{log} \\ o_i \prec_{seq\_trans} o_j & \textbf{si } o_i, o_j \in OS^*_{racine\_ling} \\ o_i \prec_{seq\_ling} o_j & \textbf{si } o_i, o_j \in OS^*_{ling} \Leftrightarrow OS^*_{racine\_ling} \end{cases}$$

La notation $o_i \prec_{seq\_text} o_j$ signifie que les éléments $o_i$ et $o_j$ proviennent d'une même structure (linguistique ou logique), qu'ils sont frères dans cette structure et que l'élément $o_j$ est le successeur direct de l'élément $o_i$.

**$\prec_{ref}$** est la relation de référence définie dans la structure syntaxique. Elle exprime la référence d'un élément (source) à un autre élément (cible). Il s'agit en fait de relations transversales à la hiérarchie structurelle décrite par les relations de composition et de séquence. Il s'agit d'une relation binaire interne sur l'ensemble $OS_{text}$ notée : $\prec_{ref} \subset OS_{text} \times OS_{text}$.

La notation $o_i \prec_{ref} o_j$, avec $o_i \in OS^*_{ling} \cup OS_{log}$ et $o_j \in OS^*_{ling} \cup OS_{log}$, signifie que l'élément $o_i$ réfère l'élément $o_j$. Cette relation est *réflexive*, *non transitive* et *asymétrique*.

Parmi les relations de référence, nous retrouvons les *relations d'intention* définies dans la structure de discours du document textuel. Elles expriment qu'un élément de la structure de discours a pour intention les idées provenant d'un élément de la structure de discours ou plus généralement elle précise la description de l'élément cible à l'aide du contenu de l'élément source.

Les relations d'intention correspondent aux relations de référence caractérisées par le type d'information qu'elles précisent.

$$\prec_{intention} \subset \prec_{ref}$$

**$\mathbf{type_{st}}$** est la fonction surjective d'assignation d'un type d'élément structurel pris dans $TYPE_{ST}$ à un élément structurel de $OS_{text}$; $type_{st} : OS_{text} \rightarrow TYPE_{ST}$.

$$type_{st}(o) = \begin{cases} t_{ling} \text{ avec } t_{ling} \in TYPE_{ling} & \textbf{si } o \in OS_{ling} \\ t_{log} \text{ avec } t_{log} \in TYPE_{log} & \textbf{si } o \in OS_{log} \end{cases}$$

### 3.4.2 Caractéristiques du modèle intégré

Les caractéristiques de ce modèle sont similaires à celles que nous avons énoncées pour les structures linguistiques et logiques (pages 76 et 80). Nous caractérisons ainsi un *élément racine* qui correspond à l'entité logique qui est la racine de la structure logique et qui est noté $o_{racine\_log}$. L'*ensemble des éléments feuilles* d'un document est l'ensemble des éléments linguistiques feuilles noté $OS_{f\_ling}$.

L'ensemble des *types racines* correspond à l'ensemble des types racines logiques. Chacun de ces types minimaux correspond en fait à une classe particulière de document : Livre, Article, Documentation Technique, etc. De manière similaire, l'ensemble des types feuilles est l'ensemble des types feuilles linguistiques. Enfin, les éléments de $OS_{text}$ sont ordonnés dans l'espace linéaire de lecture du document, c'est-à-dire qu'il est possible de comparer la position de deux éléments sur cet axe.

## 3.5 Synthèse

Nous identifions clairement trois relations de structure syntaxique : la relation de composition, la relation de séquence et la relation de référence. Ces relations organisent et lient l'ensemble des éléments structurels qui représentent chaque fragment du document. Cette organisation prend la forme d'une arborescence comportant des relations transversales : séquence et référence.

Par ailleurs, nous montrons que des relations sémantiques correspondent à ces relations de structure syntaxiques. Dans la plupart des systèmes que nous avons rencontrés, ces relations syntaxique ont elles aussi été identifiées et représentées plus ou moins complètement mais elles ne sont que faiblement prises en compte pour l'instanciation des descripteurs des éléments structurels. Ceci signifie que la liaison entre la structure syntaxique et le sens ou l'utilité de celle-ci n'est que trop rarement étudiée. L'idée que nous avons est donc d'utiliser la structure syntaxique lors d'un protocole d'indexation en utilisant les relations sémantiques correspondant à ces relations syntaxiques.

# Chapitre 4

# Modèle de représentation des images

## 4.1 Introduction

Pour la modélisation des images fixes, nous avons comme idée directrice de mettre en évidence la *structure* d'une image. Le problème est plus difficile dans le cas d'une image, car contrairement au texte, nous ne pouvons pas nous baser sur une structure syntaxique existante (structure linguistique et logique). Dans un premier temps, nous cherchons à décrire les caractéristiques de l'image en montrant quelles structures apparaissent.

### 4.1.1 Caractéristiques de l'image

Dans ces définitions informelles, nous nous attachons à établir les liens entre les caractéristiques de l'image et celles du document textuel. Nous allons tout d'abord tâcher de décrire l'image selon deux visions : une vision physique et une vision sémantique. Nous rappelons que nous représentons des images dites 2D.

**Aspect Physique** : dans notre contexte, une image est formée par un assemblage d'un ensemble de pixels dans un espace à deux dimensions. C'est la disposition de cet ensemble de pixels qui donne sa cohérence à l'image. Quel que soit le type d'image (noir et blanc, niveaux de gris ou couleur) un pixel est physiquement caractérisé. Nous pouvons également établir des regroupements de pixels connectés en utilisant des critères physiques communs à ces pixels (couleur, texture, etc). D'un point de vue physique, une image est représentable sous la forme d'un ensemble de régions; une *région* se définit alors comme un ensemble connecté de pixels de l'image ayant une ou plusieurs caractéristiques physiques communes.

**Aspect Sémantique** : une image est une représentation dans un plan à deux dimensions d'objets provenant d'un espace de dimension égale ou supérieure (3 dimensions le plus souvent).

Les objets de l'image ont une sémantique que nous sommes capables de fournir si nous connaissons l'espace duquel ils proviennent. Par exemple, nous dirons que la région 1 de l'image de la figure 4.1 est une personne, que la région 3 est le ciel, ... etc.

La description sémantique dépendra du référent sémantique adopté, c'est-à-dire d'un ensemble de connaissances faisant office de référentiel. Par exemple si l'image est décrite par une personne quelconque, le référent sera l'ensemble des connaissances de cette personne. Pour un système, le référent dépendra du contexte d'application et des connaissances relatives à ce contexte.

L'identification des éléments *sensibles ou intéressants* de l'image est nécessaire pour un système de recherche d'informations. Cette identification ne donne toutefois pas de sens à l'image. Elle permet de donner les composants symboliques de l'image et aussi la manière dont ils se composent à l'intérieur de l'image, par exemple un arbre est composé d'un tronc et de branches. Cette relation de composition entre éléments permet de structurer la description sémantique de l'image.

D'autre part, il apparaît que certaines images peuvent supporter une description supplémentaire : la description de la ou des scènes qui sont représentées par l'image. Une scène dans une image fixe se définit de la manière suivante : *action ou état de fait mettant en jeu plusieurs éléments de l'image (au moins deux)*. L'introduction de la notion de scène dans la description de l'image permet d'avoir une description plus complète de l'image.

Une description basée uniquement sur l'identification des éléments fournit une description *statique*. Nous savons par exemple que l'image contient un terrain de tennis, un terrain de golf, et deux personnes mais nous ne pouvons dire dans quelle situation se trouvent ces personnes ou ce qu'elles font. Une scène regroupant le terrain de golf, et les deux personnes indique que deux personnes jouent au golf. *La scène est un ensemble d'éléments de l'image qui, lorsqu'ils sont regroupés, supportent une sémantique particulière.*

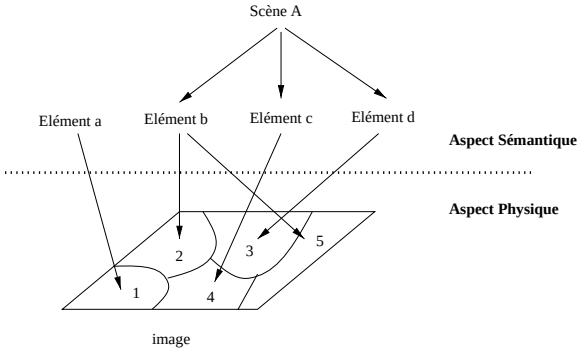



**Figure 4.1.** Combinaison des aspects physique et sémantiques d'une image

En séparant les aspects physiques et sémantiques de la description d'une image fixe nous avons mis en évidence de manière informelle les éléments à considérer. De l'aspect physique de l'image nous retenons que l'image est composée d'un ensemble connecté de points (pixels) admettant des regroupements selon des caractéristiques physiques. L'image est alors un ensemble de régions, une région étant un ensemble connecté de points de l'image vérifiant un critère physique: couleur moyenne des pixels, texture, ... etc. De l'aspect sémantique, nous pouvons retenir que l'image peut être décrite par un ensemble d'éléments identifié selon un référentiel donné pouvant se composer pour former des scènes au sein de l'image.

Comme nous le montrons dans la figure 4.1, nous considérons la combinaison des deux aspects; un élément sémantique provient de la composition d'un ou plusieurs ensembles connectés de pixels. En effet nous ne pouvons pas considérer qu'un élément *sémantique* provient toujours d'un ensemble de pixels vérifiant les mêmes critères physiques.

Nous décrivons par la suite plus précisément les éléments que nous utilisons dans la représentation d'une image, leurs noms ainsi que les règles que nous proposons pour les composer. Cette description reste toutefois informelle puisque nous donnons une formalisation de la structure de l'image dans la section qui suit (Section 4.2.1).

### 4.1.2 Définitions

Nous introduisons quatre types d'objets afin de décrire la structure d'une image: les points, les régions, les éléments et les scènes. Les deux premiers types proviennent d'une description physique de l'image, les deux suivants de la description sémantique.

**Un point** est un pixel de l'image. Les points sont donc les unités physiques de base; c'est en effet le plus petit élément *visible* d'une image.

Par analogie avec les objets qui composent le texte, on peut faire correspondre aux points de l'images les caractères du texte. Leurs rôles sont similaires; pris indépendamment les uns des autres ils n'ont pas de sens, organisés dans un espace donné par un auteur ils s'accordent pour former le document. Cette organisation fournit une structure syntaxique ou physique qui ne garantit en aucun cas que le document sera compréhensible. Ceci est vrai pour l'image et pour le texte.

Un point (ou pixel) comporte deux types d'informations qui sont à la source d'autres informations (par exemple l'histogramme des couleurs qui est défini sur un ensemble de points):

1. sa *position* dans la matrice de pixels de l'image,
2. sa *couleur*, un code pris dans un espace de couleur donné (RGB, HSB, etc)

**Une région** représente un *ensemble connecté de pixels* de l'image. Cet ensemble de pixels provient d'une segmentation de l'image selon des critères physiques, par exemple la couleur, la texture, ... etc.

Nous considérons qu'il existe un processus d'analyse de l'image fournissant en sortie un ensemble de régions qui recouvrent complètement l'image. Ces régions constituent une partition de l'image.

Une région est alors caractérisée par :

1. *l'ensemble des pixels* qui composent la région,
2. sa *position* dans l'image, ceci peut se traduire en terme de centre de gravité de la région, de contour de la région, de surface intérieure, de surface extérieure, ... etc,
3. ses *relations spatiales* par rapport aux autres régions de l'image,
4. un *histogramme des couleurs*, ou une couleur moyenne, qui fournira un indicateur visuel de référence,
5. une *texture*, si celle-ci est pertinente, c'est-à-dire reconnue dans un ensemble de référence de textures connues.

**Un élément** représente un *objet du monde réel* qui a été identifié dans l'image. Un objet du monde réel est défini selon le référent sémantique adopté.

En termes de composition structurelle, un élément est défini comme la composition d'une ou plusieurs régions de l'image. Un élément comporte deux facettes distinctes : la facette physique qui se réfère à une ou plusieurs régions de l'image et une facette sémantique qui se réfère à l'objet dans son monde originel.

Un élément est caractérisé par :

1. *l'ensemble des régions* qui composent l'élément,
2. les *relations spatiales* entre les éléments de l'image prises dans le plan de l'image,
3. les *relations spatiales* entre les éléments de l'image prises dans l'espace original des éléments s'il est différent du plan de l'image,
4. la *valeur sémantique* de l'élément provenant du référent adopté, en d'autres termes son sens,
5. les *caractéristiques physiques* ayant une signification pour l'élément.

**Une scène** représente la *mise en relation de plusieurs éléments de l'image* afin de former une action ou un fait particulier qui apparaît dans l'image. Elle peut par ailleurs permettre de déterminer le contexte particulier de l'image.

Une scène est caractérisée par :

1. *l'ensemble des éléments* qui composent la scène,
2. la *valeur sémantique* de la scène toujours selon un référent,

Nous considérons ici qu'une image peut correspondre à une mise en scène d'un ensemble d'éléments et cette mise en scène provient d'un choix de l'auteur de l'image. Nous choisissons donc d'associer un sens particulier à cet ensemble d'éléments. Toutefois, nous pensons qu'une image ne comporte pas nécessairement de scène et dans ce cas nous nous limiterons à une description des éléments de l'image.

Nous prenons en compte dans la description de l'image des objets provenant de la description physique et des objets provenant de la description sémantique. Les objets d'ordre *sémantique* sont des compositions d'objets d'ordre *physique*. La description sémantique est

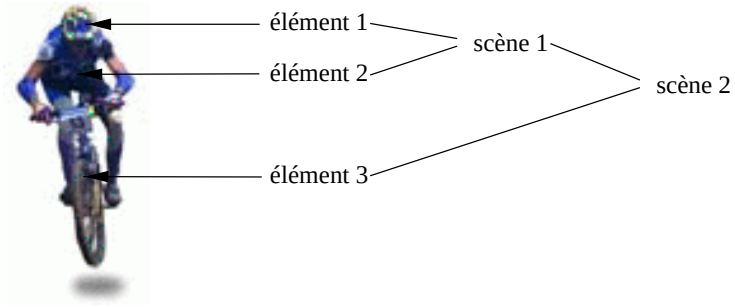



**Figure 4.2.** Décomposition d'une image

basée sur des régions de l'image ou sur l'image complète. Nous donnons par la suite une formalisation de la structure afin de définir plus précisément la notion de composition entre objets *physiques* et objets *sémantiques*.

Nous montrons avec la figure 4.2 comment décrire une image avec les types d'objets que nous proposons. Dans cette image nous avons indiqué les éléments et les scènes que nous considérons. Nous identifions 3 éléments distincts : le casque (élément 1), la personne (élément 2) et le vélo (élément 3). Nous mettons ensuite ces éléments en relation par l'intermédiaire d'objets scènes : la scène 1 qui est composée des éléments 1 et 2 indique que l'image comporte une personne qui porte un casque, la scène 2 indique que l'image contient une personne qui porte un casque et qui fait du vélo.

## 4.2 La structure dans l'image fixe

Nous donnons tout d'abord les éléments formels caractérisant la structure dans l'image fixe puis nous complétons par des propriétés sur ces éléments, particulièrement sur la relation de composition et sur les types d'objets.

### 4.2.1 Formalisation de la structure

On décrit une image et sa structure à l'aide de quatre types d'objets, d'une relation de composition et d'un ensemble de relations spatiales. L'image est décrite par un ensemble d'objets typés qui sont mis en relation d'une part par une relation de composition déterminant la structure et d'autre part par des relations spatiales.

Les informations structurelles de l'image sont donc représentées par deux classes d'informations : la *définition de la structure*, notée $ST_{pict}$, et l'*instance de l'image* se conformant à cette structure, notée $I_{pict}$.

*La définition de la structure dans l'image*

$$ST_{pict} = (TYPE_{pict}, \preceq_{tcomp\_pict}, R_{sp})$$

**$\mathbf{TYPE_{pict}}$** est l'ensemble des types d'objets structurels d'une image. Nous allons considérer l'ensemble des types suivants pour décrire l'image: point, région, élément et scène. Nous définissons plus précisément ces types et nous revenons sur les propriétés particulières de chacun d'eux par la suite (section 4.2.3).

Toutefois, nous pouvons considérer les définitions suivantes :

- les objets de type *point* sont des pixels,
- les objets de type *région* sont des ensembles de pixels connectés de l'image,
- les objets de type *élément* sont des agrégations d'objets de type région représentant un objet du monde réel,
- les objets de type *scène* sont des agrégations d'objets de type élément représentant une action ou un fait particulier de l'image .

**$\preceq_{\mathbf{tcomp\_pict}}$** est la relation de sous-typage sur les types d'objets structurels définis dans l'ensemble $TYPE_{pict}$. Il s'agit d'une relation binaire interne sur les types d'objets structurels : $\preceq_{tcomp\_pict} \subset TYPE_{pict} \times TYPE_{pict}$.

**$\mathbf{R_{sp}}$** contient la définition des relations spatiales ainsi que les contraintes spécifiant entre quels types d'objets ces relations sont disponibles.

*L'instance de l'image*

$$I_{pict} = (OS_{pict}, \prec_{comp\_pict}, type_{pict}, \Delta_{sp})$$

**$\mathbf{OS_{pict}}$** : ensemble des objets structurels d'une image. Un objet structurel est un composant physique de l'image, c'est-à-dire un ensemble non vide de pixels de l'image.

Un objet structurel de l'image est défini comme un élément singulier de l'image sur lequel différents types d'informations vont pouvoir être définis: contenu sémantique, caractéristiques physiques, ...

**$\prec_{\mathbf{comp\_pict}}$** : relation de composition structurelle entre les objets structurels d'une image. Cette relation indique qu'un objet structurel est composé d'un autre objet structurel. Il s'agit d'une relation binaire interne sur l'ensemble $OS_{pict}$ : $\prec_{comp\_pict} \subset OS_{pict} \times OS_{pict}$.

Les propriétés et les contraintes associées à cette relation dépendent du type d'objets structurels et sont décrits plus loin (section 4.2.2).

**$\mathbf{type_{pict}}$** est une fonction surjective d'assignation d'un type structurel à un objet structurel. Chaque objet structurel de l'ensemble $OS_{pict}$ est typé, et reçoit donc un type pris dans l'ensemble des types structurels $TYPE_{pict}$.

$$type_{pict} : OS_{pict} \rightarrow TYPE_{pict}$$

**$\mathbf{R_{sp}}$** est l'ensemble de relations spatiales définies entre des objets structurels d'une image.

$$\forall\ \delta_{sp} \text{ on a } \delta_{sp} \subset OS_{pict} \times OS_{pict}$$

Ces relations sont de différentes natures et leur application est contrainte par les types d'objets et par leurs définitions données dans $\Delta_{sp}$ de $ST_{pict}$. Il existe des relations spatiales dans le plan de l'image et des relations spatiales dans l'espace réel des objets de l'image. Nous donnons en annexe C des compléments sur ces relations.

### 4.2.2 La relation de composition structurelle

La relation de composition structurelle est une relation binaire qui décrit la relation père/fils dans la structure de l'image. Elle définit l'agrégation des objets structurels de l'image. La notation $o_i \prec_{comp\_pict} o_j$ signifie que l'objet structurel $o_i$ est composé de l'objet structurel $o_j$. En d'autres termes $o_i$ est le père structurel de $o_j$.

$$o_i, o_j \in OS_{pict}, o_i \prec_{comp\_pict} o_j$$

Cette relation de composition des objets structurels de l'image peut être interprétée au niveau physique de la manière suivante: *sachant qu'un objet structurel est une représentation logique d'un ensemble non vide de pixels, si un objet $o_i$ est composé d'un objet $o_j$ alors l'ensemble de pixels représenté par $o_j$ est inclus dans l'ensemble de pixels représenté par $o_i$.*

La relation de composition structurelle supporte les propriétées suivantes: *non réflexivité*, *non transitivité* et *asymétrie.*

La sémantique générale de la relation de composition est liée à la composition physique des pixels de l'image. Nous distinguons plusieurs cas :

1. un objet $o$ de type élément composé d'objets de type élément $o_i, i = 1 \ldots n$: la relation de composition indique qu'il existe une relation au niveau sémantique entre chacun des objets $o_i$. Le plus souvent cette relation est une relation du type *partie de*. Par exemple, un objet $o_1$ représentant une roue sera un composant d'un objet $o$ représentant un véhicule; *la roue est une partie de la voiture.*

2. un objet $o$ de type scène composé d'objets de type élément $o_i, i = 1 \ldots n$: la relation de composition indique les n objets de types élément participent à une action ou à un fait particulier de l'image. Par exemple, dans la figure 4.2, l'objet scène 1 est composé des objets représentant la personne et le casque, et cet objet scène 1 indique le fait suivant : *une personne porte un casque.*

3. un objet $o$ de type scène composé d'objets de type scène $o_i, i = 1 \ldots n$: la relation de composition indique que les scènes, représentées par les objets $o_i, i = 1 \ldots n$, participent à la scène représentée par l'objet $o$. Dans l'image de la figure 4.2 l'objet noté scène 2, composé des objets élément 3 et scène 1, représente une *personne portant un casque (scène 1) qui fait du vélo.*

### 4.2.3 Définition et propriétés structurelles des types d'objets

Dans les travaux sur les images la classification adoptée comporte le plus souvent trois types d'éléments : le polygone, le segment et le point pour Mechkour [Mec95c] et la région

spatiale (areal object), le segment (lineal object) et le point (point object) pour Kankanhalli, Xunda et Wu [KXW95]. Les SGBD géographiques confrontés à des problèmes de modélisation d'objets géométriques dans un plan adoptent une classification similaire composée de zones, de segments et de points [SVP+96].

Ces types tiennent uniquement compte de l'aspect 2D de l'image. Ces restrictions adoptées dans la plupart des modèles d'image [Mec95c, KXW95, Meg95] proviennent en grande partie des difficultés rencontrées pour raisonner sur les relations spatiales dans un espace à trois dimensions. Notre approche diffère car nous considérons le contenu de l'image selon deux aspects combinés : l'aspect physique et l'aspect sémantique. Selon nous, l'image n'est pas seulement une composition d'objets identifiés dans l'image, mais c'est aussi une mise en relation de ces objets mettant les objets identifiés dans des situations particulières. Cette vision de l'image nous a amenés à définir quatre types d'objets : le point, la région, l'élément et la scène. La figure 4.3 donne les relations de sous-typage entre types d'objets. La relation de composition structurelle est alors un moyen de décrire les zones de l'image, les objets identifiés et leurs compositions.

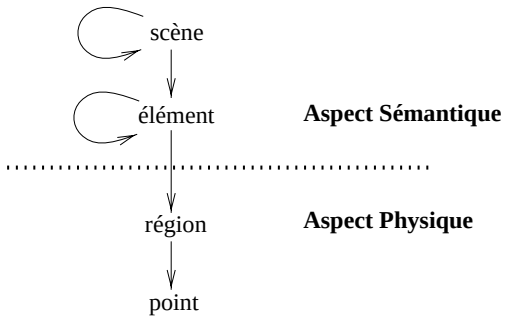



**Figure 4.3.** Types et relations de sous-typage $\preceq_{tcomp\_pict}$

Cette figure représente la relation binaire $\prec_{comp\_pict}$ définie sur $TYPE_{pict}$. Nous définissons pour chaque type d'objets les contraintes associées :

**point** : le point est une représentation logique du pixel de l'image. Les coordonnées de ce pixel dans l'image vont permettre de définir la localisation du point. De plus ce point peut être caractérisé par sa couleur (code dans un espace de couleur, RGB ou HSB).

Le point est l'*élément de base* dans la structure. Il ne peut donc être le père d'aucun autre objet structurel, ce qui s'exprime par :

$$\forall\, o_p \in OS_{pic},\ type_{pic}(o_p) = point \textbf{ implique}$$
$$\nexists\, o \in OS_{pic},\ o_p \prec_{comp\_pic} o$$

**région** : un objet de type région représente un ensemble connecté de pixels de l'image. La région est décrite à l'aide d'objets de type point qui la composent et des caractéristiques

physiques de ces objets. Ceci s'exprime de la manière suivante :

$\forall\, o_r \in OS_{pic},\ type_{pic}(o_r) = region$ **implique**
$\exists\, o_{pi} \in OS_{pic},\ i > 1,\ o_r \prec_{comp\_pic} o_{pi}$ et $type_{pic}(o_{pi}) = point$

De plus nous devons vérifier que l'ensemble des objets de type point composant deux objets de type région sont disjoints. Soit la fonction $Desc_{pic}$ qui fournit pour chaque objet l'ensemble des objets qui le compose : $Desc_{pic} : OS_{pic} \rightarrow 2^{OS_{pic}}$. La contrainte sur les ensembles d'objets composant une région s'exprime alors de la manière suivante :

$\forall\, o_1, o_2 \in OS_{pic},\ type_{pic}(o_1) = point$ et $type_{pic}(o_2) = point$ **implique**
$Desc_{pic}(o_1) \cap Desc_{pic}(o_2) = \emptyset$

On peut noter que tous les objets des ensembles $Desc_{pic}(o_1)$ et $Desc_{pic}(o_2)$ sont des objets de type point. Ceci provient de la contrainte sur la composition des objets de type région énoncée précédemment. D'où $\forall o \in OS_{pic}$ tel que $type_{pic}(o) = region$ alors $\forall o_i \in Desc_{pic}(o),\ type_{pic}(o_i) = point$.

**élément** : un objet de type élément provient de la composition d'objets de type région et/ou élément. Le référent physique dans l'image de cet objet est un ensemble de pixels (qui n'est pas forcément connecté). Au niveau sémantique, cet objet représente un *objet du monde réel*.

$\forall\, o_e \in OS_{pic},\ type_{pic}(o_e) = element$ **implique**
$\exists\, o_i \in OS_{pic},\ o_e \prec_{comp\_pic} o_i$ et $type_{pic}(o_i) \in \{region, element\}$

**scène** : un objet de type scène provient de la composition d'objets de type élément et/ou scène. Comme pour les objets de type élément, le référent physique est un ensemble de pixels qui n'est pas forcément connecté. Cet objet représente une action ou un fait mettant en jeu plusieurs objets du monde réel identifiés dans l'image.

$\forall\, o_s \in OS_{pic},\ type_{pic}(o_s) = scene$ **implique**
$\exists\, o_i \in OS_{pic},\ o_s \prec_{comp\_pic} o_i$ et $type_{pic}(o_i) \in \{element, scene\}$

Nous synthétisons dans le tableau de la figure 4.4 les représentations physiques et symboliques des quatre types d'objets que nous utilisons. Nous donnons la représentation au niveau physique et/ou symbolique d'un type d'objets ainsi que les contraintes de composition structurelle de ces types d'objets selon deux critères : physique et/ou symbolique. Une contrainte structurelle physique signifie que la composition d'un objet est valide si des critères de composition physique sont respectés. Les contraintes symboliques sont introduites pour garantir une cohérence symbolique, par exemple un objet représentant une personne ne peut pas composer un objet représentant une voiture.

Notre représentation structurelle se fait donc selon deux axes complémentaires : une composition structurelle d'objets (point et région) provenant d'une caractérisation physique de l'image et une composition structurelle d'objets (élément et scène) provenant d'une caractérisation sémantique de l'image.

| | Représentation Physique | Représentation Symbolique | Contraintes de Composition | |
|---|---|---|---|---|
| | | | physiques | symboliques |
| **point** | un pixel de l'image | | n'admet pas de fils structurel | |
| **région** | un ensemble connecté de pixels de l'image | | objets fils de type point et caractéristique physique commune (1) | |
| **élément** | un ensemble de pixels de l'image | un objet du monde réel O | objets fils de type région et/ou élément | composition d'objets représentant l'objet O (2) |
| **scène** | un ensemble de pixels de l'image | une mise en relation de plusieurs objets du monde réel | objets fils de type élément et/ou scène | composition d'objets du monde réel (3) |

**Figure 4.4.** Niveaux de représentation des images et contraintes de composition associées

1. un objet $o$ de type *région* est uniquement composé d'objets de type point. Ce type d'objets peut être extrait de manière automatique par l'usage de techniques liées à l'analyse de l'image.

2. un objet $o$ de type *élément* représente un objet du monde réel, noté O. Cet objet peut être décomposé selon sa sémantique en autres objets du monde réel : une voiture est composée de ses roues, de sa carosserie, etc. Cette composition peut provenir d'une base de connaissance permettant de donner une cohérence symbolique.

   Les fils structurels directs de $o$ sont soit des objets $o_R$ de type région, soit des objets $o_E$ de type élément. Dans le premier cas, les objets $o_R$ n'ont pas de sémantique associée et ils correspondent aux différentes parties visibles de l'objet O dans l'image. Pris indépendamment les uns des autres, ces objets ne forment pas un objet identifiable du monde réel. Dans le second cas la sémantique associée aux objets $o_E$ est cohérente avec la sémantique de l'objet O (cf. base de connaissance).

3. un objet de type *scène* $o$ provient d'une composition d'objets de l'image ayant une sémantique associée. Ce sont donc soit des objets de type élément soit des objets de type scène.

## 4.3 Description d'une image

Les informations décrivant chaque objet structurel sont représentées par l'intermédiaire d'attributs spécifiques. Ainsi les caractéristiques physiques ou sémantiques d'un objet structurel sont représentées par des attributs attachés à cet objet.

Nous distinguons pour l'image fixe trois sortes d'attributs : les *attributs d'ordre physique*, les *attributs de contenu sémantique* et les *attributs génériques*. Nous détaillons par la suite les attributs physiques ainsi que les attributs sémantiques. Les attributs génériques expriment des informations qui complètent la description de l'image en indiquant une date de création, un auteur ou tout autre information de ce type qui ne peut être extraite de l'image.

### 4.3.1 Attributs physiques

Une image est décrite par les symboles physiques qui la forment, autrement dit les objets structurels que nous avons définis. Ces symboles ont des caractéristiques qu'il faut pouvoir utiliser lors d'une session de recherche. Nous allons tout d'abord définir les attributs qui peuvent s'appliquer de manière générale à tout type d'objets dans une image et à l'image complète. Puis nous définissons certaines fonctionnalités propres aux caractéristiques des symboles d'une image. Enfin nous spécifions ces caractéristiques selon les types d'objets structurels de l'image.

#### a) Attributs généraux

Une image fixe est caractérisée par des informations relatives à sa définition physique dans le mode *raster*. Dans ce mode, aussi appelé format matriciel, l'image est une matrice de pixels indépendants au sein de laquelle les objets qui composent l'image ne sont pas distingués. Il s'agit donc des informations physiques de base qui sont naturellement pauvres d'un point de vue sémantique telles que :

- Largeur de l'image : $l$
- Hauteur de l'image : $h$
- Ensemble des pixels formant l'image : $I_{pic} = \{p_i\}$ où les $p_i$ sont les pixels.
- Type de l'image définissant un espace de représentation des couleurs de l'image.
- Histogramme des couleurs.

Nous nous limitons ici à ce type d'informations. D'autres caractéristiques telles que le format (GIF, JPEG,TIFF, etc) spécifiant un type d'encodage pourront être repris dans les *attributs génériques*. Nous pouvons aussi donner de manière générale les informations caractérisant les pixels de l'image : position du pixel dans la matrice et la couleur du pixel, donnée par un code dans l'espace de représentation des couleurs de l'image (RGB, CMJ ou HLS).

### b) Fonctionnalités

Pour définir les fonctions suivantes, nous utilisons les travaux de Wu & al. [KXW95]. Ces fonctions permettent de caractériser les objets de l'image et leur disposition dans l'image.

- définition des voisins d'un pixel. Nous distinguons deux types de voisins, les voisins directs et indirects respectivement décrits par les fonctions $voisin_d$ et $voisin_i$. Ces deux fonctions ont la forme suivante :

  $$v : I_{pic} \rightarrow 2^{I_{pic}}$$

- la fonction notée $pixel(objet)$ fournit l'ensemble des pixels qui forment l'objet structurel $objet$. Elle est définie comme suit :

  $$pixel : OS_{pic} \rightarrow I_{pic}$$

Les fonctions pour définir l'ensemble des pixels d'un objet ($pixel$), la surface intérieure ($surf\_int$), la fermeture ($fermeture$), la surface extérieure ($surf\_ext$) et le contour ($contour$) ont la forme de la fonction f suivante : $f : I_{pic} \rightarrow 2^{I_{pic}}$.

On introduit par ailleurs les opérateurs de projection $\pi_x$ et $\pi_y$ qui fournissent la liste des coordonnées des objets selon les axes x et y (cf. figure 4.5.

$$\pi_x : OS_{pic} \rightarrow [1, l] \quad \text{et} \quad \pi_y : OS_{pic} \rightarrow [1, h]$$

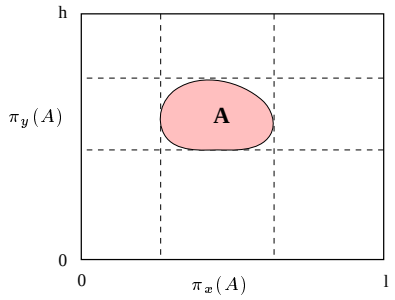



**Figure 4.5.** Opérateurs de projection

De plus nous utilisons les opérateurs *supérieur*, *inférieur* et *inclus* sur les intervalles résultant.

**supérieur** , opérateur binaire noté $>$, un segment A est supérieur à un segment B si et seulement si $min(A) > max(B)$.

**inférieur** , opérateur binaire noté $<$, un segment A est inférieur à un segment B si et seulement si $max(A) > min(B)$.

**inclus** , opérateur binaire noté $\sqsubset$, un segment A est inclus dans un segment B si et seulement si $min(A) > min(B)$ et $max(A) < max(B)$.

### c) Attributs particuliers

L'image et les principales informations s'y rattachant ayant maintenant été données, nous pouvons reprendre plus précisément la caractérisation de chaque type d'objets.

**Un point** $o_p$ est assimilé à un pixel $p$ de $I_{pic}$, et sera donc représenté par les coordonnées du pixel. A partir de ces coordonnées on peut obtenir son contour et sa surface extérieure par une fonction.

| | | |
|---|---|---|
| pixel de $o_p$ | $pixel(o_p) = \{p\}$ | |
| position de $o_p$ | $(x, y)$, coordonnées du pixel $p$ | $\pi_x(o_p) = \{x\}$ et $\pi_y(o_p) = \{y\}$ |
| contour de $o_p$ | $contour(o_p) = \{p\}$ | |
| surface intérieure de $o_p$ | $surf\_int(o_p) = pixel(o_p)$ | |
| surface extérieure de $o_p$ | $surf\_ext(o_p) = I_{pic} \Leftrightarrow pixel(o_p)$ | |

**Une région** $o_r$ est assimilée à un ensemble de pixels *connectés*.

| | | |
|---|---|---|
| pixels de $o_r$ | $pixel(o_r) = O_R$ | |
| position de $o_r$ | liste des coordonnées $(x_i, y_i)$ des $p_i$ de l'ensemble $O_R$ | $\pi_x(o_r) = \{x_i\}$ et $\pi_y(o_r) = \{y_i\}$ |
| surface intérieure de $o_r$ | $surf\_int(o_r) = pixel(o_r)$ | |
| surface extérieure de $o_r$ | $surf\_ext(o_r) = I_{pic} \Leftrightarrow pixel(o_r)$ | |
| contour de $o_r$ | $contour(o_r) = \{p \mid p \in O_R$ et $\exists\, p_i \in voisin_d(p)$ tel que $p_i \notin O_R\}$ | |

**Un élément** $o_e$ est assimilé à un ensemble de pixels regroupant un ou plusieurs objets de type région.

| | | |
|---|---|---|
| pixels de $o_e$ | $pixel(o_e) = O_E$ | |
| position de $o_e$ | liste des coordonnées $(x_i, y_i)$ des pixels $p_i$ de l'ensemble $O_E$ | $\pi_x(o_e) = \{x_i\}$ et $\pi_y(o_e) = \{y_i\}$ |
| surface intérieure de $o_e$ | $\forall o_i,\ o_i \prec_{str\_pic} o_e$ alors $surf\_int(o_e) = \bigcup_i (surf\_int(o_i))$ | union des surfaces intérieures des composants de $o_e$ |
| surface extérieure de $o_e$ | $surf\_ext(o_e) = I_{pic} \Leftrightarrow pixel(o_e)$ | |

**Une scène** $o_s$ est assimilée à un ensemble de pixels provenant de plusieurs objets de type élément ou scène.

| | | |
|---|---|---|
| pixels de $o_s$ | $pixel(o_s) = O_E$ | |
| position de $o_s$ | liste des coordonnées $(x_i, y_i)$ des pixels $p_i$ de l'ensemble $O_S$ | $\pi_x(o_s) = \{x_i\}$ et $\pi_y(o_s) = \{y_i\}$ |

### 4.3.2 Attribut de contenu Sémantique

Le contenu sémantique d'un document se réfère à un type d'information particulier véhiculé par le document et par les symboles qui composent ce document. En reprenant l'hypothèse de base de la plupart des travaux ayant abordé la classification de l'information, nous pouvons considérer trois niveaux d'information :

- le *signifiant*, c'est-à-dire la forme des symboles du document,
- le *signifié*, c'est-à-dire la signification véhiculée par les symboles du document,
- la *pragmatique*, c'est-à-dire la signification véhiculée par les symboles du document dans un contexte particulier.

Dans le cas de l'image, les symboles la composant sont des objets structurels : point, région, élément et scène. Pour définir le contenu sémantique d'une image, il faut choisir une signification particulière pour ces objets parmi celles possibles. Cette signification dépendra du contexte qui peut être celui de l'image elle-même, d'une application particulière ou du document complet à laquelle elle appartient.

Par exemple, si l'image est isolée du reste du document, nous ne considérons que les informations qui proviennent du contenu de l'image. Les caractéristiques physiques permettent de valuer les attributs physiques et les informations sémantiques proviennent de l'identification des objets présents dans l'image puis de leur regroupement éventuel selon la notion de scène.

Ces informations, démunies du contexte particulier dans lequel se situe l'image, sont limitées. Par exemple, on identifie une personne ou un monument dans une image, mais sans l'aide du contexte ou d'une connaissance plus précise on ne peut pas dire de quelle personne il s'agit ou bien quel est le monument, sur quel site il est visible, ... etc. Nous devons donc déterminer d'où proviennent ces informations, dites contextuelles, comment on peut les récupérer et comment on peut les intégrer de manière à les rendre utilisables.

Intéressons nous tout d'abord à la constitution du contenu sémantique hors contexte. L'élaboration de ce contenu sémantique correspond à la garniture des attributs de contenu sémantique des objets structurels de l'image. Nous avons dit par ailleurs que seuls les objets de type *élément* et *scène* ont une représentation sémantique qui leur est associée. Ceci étant une hypothèse de base incontournable, nous allons considérer les objets feuilles de type *élément* (composés uniquement d'objets de type *région*).

Ces objets sont identifiés comme des objets du monde réel. Donc ils peuvent être décrits par un concept ou un terme. Par exemple, nous avions décrit les objets de l'image de la figure 4.2 par les termes vélo, casque et personne. Ces termes (ou concepts) permettent de garnir sommairement les attributs des objets non composés. Maintenant il nous semble aussi souhaitable de donner une signification aux objets composés; les objets de type *scène* et les objets non feuilles de type *région*.

Il nous faut donc garnir les attributs de contenu sémantique de ces objets. Nous considérons ici deux possibilités liées à l'interprétation de la relation de composition structurelle. Afin de montrer ces deux possibilités, prenons un exemple. Dans une première image nous considérons un objet de type élément représentant une roue et qui est composant d'un autre objet de type

élément représentant une voiture. Au niveau sémantique, la roue est un composant de la voiture. Il semble donc que l'attribut de contenu sémantique de l'objet représentant la voiture sera garni par le concept voiture sans autre considération pour ses composants.

Considérons maintenant l'image prise en exemple dans la figure 4.2. Dans cette image, les objets représentant le casque et la personne forment un nouvel objet. Dans ce cas, on préférera exprimer que ce nouvel objet correspond à un une personne portant un casque. Nous n'introduisons donc plus ici un nouveau concept comme dans le cas précédent mais une relation symbolique liant les composants. Cette stratégie va bien entendu dépendre du langage d'indexation selon qu'il accepte ou non les relations symboliques.

### 4.3.3 Référent des attributs sémantiques

Afin de pouvoir manipuler le contenu des attributs sémantiques, nous considérons le référent de connaissance noté $REF$ qui est constitué des informations suivantes :

$$REF = (B, \prec_B, RS, M_{RS})$$

- **B** est un ensemble de termes, $B = \{t_i\}$.
- $\prec_{\mathbf{B}}$ est une relation décrivant la composition des termes de $B$. Cette relation exprime qu'un terme $t_i$ peut être le composant du terme $t_j$ : $t_j \prec_B t_i$.

  $$\prec_B \subseteq B \times B$$

  Par exemple, la relation ($voiture \prec_B roue$) exprime qu'une voiture est composée d'une roue. La relation $\prec_B$ est non réflexive, non transitive et asymétrique.
- **RS** est un ensemble de relations symboliques pouvant lier les termes de $B$.
- $\mathbf{M_{RS}}$ est un ensemble définissant l'application des relations symboliques de l'ensemble $RS$ sur l'ensemble des termes de $B$.

  $$M_{RS} \subseteq RS \times B \times B$$

On considère les fonctions $index_t$ et $index_{rs}$ qui associent à chaque objet structurel de type *élément* et *scène* un attribut sémantique. Ces fonctions déterminent respectivement les termes qui décrivent le contenu sémantique de l'objet et les relations sémantiques entre ces termes. Le résultat se conforme à un référent défini comme précédemment.

$$index_t : OS_{pic} \rightarrow 2^B \text{ et } index_{rs} : OS_{pic} \rightarrow 2^{M_{RS}}$$

Le résultat de la fonction $index_{rs}$ doit vérifier des contraintes : soit $(rs, t_1, t_2) \in index_{rs}(o_i)$, ceci est valide si et seulement si

- $\{t_1, t_2\} \subseteq index_t(o_i)$
- $(rs, t_1, t_2) \in M_{RS}$, ce triplet représente une instance des définitions des relations de $M_{RS}$.

### 4.3.4 Comportement des attributs sémantiques

L'ensemble des objets structurels admettant un attribut sémantique est défini par $OS_{sem}$. Il existe dans cet ensemble deux sortes d'objets structurels : les objets structurels n'ayant pas de fils structurels appartenant à cet ensemble, ensemble d'objets noté $OS_{feuille\_sem}$, et ceux ayant des fils structurels de cet ensemble, ensemble d'objets noté $OS_{noeuds\_sem}$. L'union des deux ensembles forme l'ensemble des objets structurels admettant un attribut sémantique :

$$OS_{sem} = OS_{feuille\_sem} \cup OS_{noeuds\_sem}$$

Il existe un lien entre le contenu des attributs sémantiques et la relation de composition structurelle définie sur les objets de $OS_{sem}$. Tout d'abord nous traitons les objets de l'ensemble $OS_{feuille\_sem}$ puis nous établissons comment les informations peuvent se propager vers les objets de l'ensemble $OS_{noeuds\_sem}$ selon la relation de composition structurelle.

L'assignation d'une valeur à l'attribut sémantique d'un objet de l'ensemble $OS_{feuille\_sem}$ correspond à l'identification des objets du monde réel. Un terme pris dans $B$ et représentant cet objet est alors affecté à l'attribut sans contrainte particulière. Nous admettons simplement que la relation $index_{rs}$ appliquée sur ces objets est nulle, c'est-à-dire que ces objets sont uniquement décrits par des concepts.

$$\forall o \in OS_{feuille\_sem} \begin{cases} index_t(o) = \{t_o\} \quad t_o \in B \\ \\ index_{rs}(o) = \emptyset \end{cases}$$

Considérons maintenant les objets de l'ensemble $OS_{noeuds\_sem}$. Il s'agit d'objets ayant des fils. Implicitement leur description sémantique est induite par la description sémantique de leurs fils. Les informations qui décrivent les objets fils participent à la construction de l'attribut sémantique de l'objet père.

$\forall o \in OS_{noeuds\_sem}$ tel que $o \prec_{comp\_pic} o_i$ et $index_t(o_i) = T_i$
On a

$$index_t(o) = \begin{cases} \{t_o\} & \text{si } \Sigma_i(T_i) \rightarrow_B t_o \\ ou & \\ \bigcup\limits_i (T_i) & \end{cases}$$

$$index_{rs}(o) \neq \emptyset \text{ si } |index_t(o)| > 1$$

Nous considérons donc deux possibilités pour la formation du contenu de l'attribut sémantique de l'objet père à l'aide du contenu des attributs sémantiques des objets composants :

1. Introduction d'un concept *père* suppléant les concepts provenant des attributs sémantiques des composants. Nous proposons ce nouveau concept $t_o$ lorsqu'il existe un terme dans $B$ qui vérifie la propriété exprimée par $\Sigma_i(T_i) \rightarrow_B t_o$.

Par exemple, lorsque plusieurs composants d'une voiture ont été identifiées, nous instancions l'objet père par le terme voiture.

2. Introduction d'une *relation symbolique* entre les concepts des attributs sémantiques des composants. Dans ce cas il n'existe pas de termes englobant les concepts exprimés dans les fils et l'objet est donc décrit par l'introduction de relations sémantiques liant les termes représentant les objets fils.

Quelle que soit la solution proposée, l'introduction d'un concept ou d'une relation symbolique, correspond à un enrichissement du contenu sémantique par rapport à une description de l'image reposant sur une liste de termes. Toutefois nous ne pouvons proposer une automatisation de ce processus de formation des attributs sémantiques. Cette automatisation signifierait que nous sommes capables, sans intervention humaine, d'une part d'inférer les bons concepts et d'autre part de reconnaître les relations sémantiques.

L'inférence automatique des concepts est possible, mais ce processus ne garantit pas une adéquation parfaite entre la description réelle de l'image et l'indexation fournie automatiquement. D'autre part, la reconnaissance des relations entre objets dans l'image peut uniquement provenir d'un choix humain.

Du fait que nous considérons que la garniture des attributs sémantiques des objets est une partie du processus d'indexation des images, au même titre que l'identification des objets structurels ou que l'extraction des caractéristiques physiques des objets, la garniture se doit d'être aussi précise que possible. La garniture reste donc un processus interactif au sein duquel le choix des termes et des relations entre ces termes doit être assuré par un indexeur humain. Nous devons pouvoir proposer à l'indexeur humain les différentes possibilités d'indexation provenant des connaissances assimilées par la machine. Ces connaissances proviennent de la base $B$ : ensemble des termes, composition des termes et spécification des relations sémantiques entre les termes.

## 4.4 Conclusion

Avant de passer à la description du modèle de document structuré, nous souhaitons positionner le modèle que nous avons défini pour les images vis-à-vis des relations de structure du document textuel structuré. Dans un document structuré, une image est un élément feuille au même titre qu'un fragment de texte. Nous avons vu dans la définition des relations de structure du document textuel que ces fragments de texte sont divisibles à plusieurs niveaux : niveau linguistique et niveau logique. Ce genre de division n'existe pas de manière naturelle sur les images. Dans notre modèle de représentation des images, nous avons cherché à rendre l'entité image divisible et structurée de manière assez similaire à ce que nous avons fait pour le texte.

Ainsi une image se décompose en scènes qui elles-mêmes se décomposent en éléments et ainsi de suite jusqu'à aboutir aux points de l'image. Chacun de ces composants est caractérisé par des attributs leur donnant une description. A partir de cette représentation, il est possible d'établir le parallèle avec la représentation des documents textuels. Au niveau linguistique, une phrase se décompose en groupes nominaux qui eux-mêmes se décomposent en mots et ainsi de suite jusqu'aux symboles de plus petite granularité du texte.

Dans le document textuel, nous avons rattaché les éléments du niveau linguistique aux éléments de la structure logique du document. Par exemple, un paragraphe (élément de la structure logique) est composé de phrases (éléments de la structure linguistique). Si nous considérons maintenant qu'il ne s'agit non plus d'un document purement textuel mais d'un document mêlant du texte et des images, la structure logique va intégrer des éléments pouvant contenir des images. Ainsi, une image de ce document, qui sera alors un élément de la structure logique, sera composée de scènes (éléments structurels de l'image).

Au même titre que les éléments textuels de la structure logique peuvent se décomposer en éléments linguistiques, les images admettent une décomposition dans notre modèle. Il apparaît de nombreuses similitudes dans la définition des relations de structure du texte et des images. Nous allons les expliciter dans le chapitre suivant.

# Chapitre 5

# Modèle de document structuré

Les études sur les relations de structure du document textuel et des images mettent en évidence de nombreuses similitudes que nous allons intégrer dans un modèle de document comprenant à la fois du texte et des images.

Ce modèle présente le document structuré en suivant des règles de construction syntaxiques décrites par des relations de structure : la relation de composition, la relation de séquence et les relations de référence. Ces relations décrivent l'arborescence structurelle, le sens de lecture linéaire et les relations transversales à l'arborescence. Les relations de séquence et de composition doivent par ailleurs vérifier des contraintes dictées par un type de structure. Le modèle de document propose des descripteurs, sous la forme d'attributs, qui contiennent des informations relatives à chaque type d'élément structurel ou à chaque sorte de médias.

Nous décrivons tout d'abord l'intégration des relations provenant des différents médias, puis nous définissons notre modèle avant de l'appliquer à un exemple.

## 5.1 Intégration des relations

Pour un document structuré mêlant du texte et des images, nous pouvons conserver les caractéristiques du texte en y joignant celles des images. Dans les définitions que nous donnons ci-dessous, nous considérons que les éléments de la structure logique peuvent mêler du texte et des images. Les éléments feuilles de la structure logique sont quant à eux soit des éléments purement textuels, soit des images.

**L'ensemble des éléments des structure** $OS$ est donné par l'union des éléments de structure textuels et des éléments de structure des images : $OS = OS_{log} \cup OS_{ling} \cup OS_{pict}$.

**La relation de composition** $\prec_{comp}$ donne la composition des éléments de structure du document. Ses propriétés sont contraintes par les types d'éléments structurels qu'elles lient. Il s'agit d'une relation binaire interne sur $OS$ : $\prec_{comp} \subset OS \times OS$.

Nous définissons cette relation en fonction des relations définies pour les éléments textuels et pour les éléments des images. Nous donnons ici la définition pour deux éléments $o_i$ et $o_j$ de l'ensemble $OS$ et spécifions quelle est la relation sous-jacente selon l'origine de ces éléments. Indépendamment de ces origines, la notation $o_i \prec_{comp} o_j$ signifie que l'élément $o_i$ est composé de l'élément $o_j$.

$$o_i \prec_{comp} o_j \Leftrightarrow \begin{cases} o_i \prec_{comp\_log} o_j & \textbf{si } o_i, o_j \in OS_{log} \\ o_i \prec_{comp\_ling} o_j & \textbf{si } o_i, o_j \in OS_{ling} \\ o_i \prec_{comp\_trans} o_j & \textbf{si } o_i \in OS_{f\_log} \text{ et } o_j \in OS_{racine\_ling} \\ o_i \prec_{comp\_pict} o_j & \textbf{si } o_i, o_j \in OS_{pict} \\ o_i \prec_{comp\_trans2} o_j & \textbf{si } o_i \in OS_{f\_log} \text{ et } o_j \in OS_{racine\_pict} \end{cases}$$

Comme lors de la définition du modèle de texte intégré nous devons définir des relations établissant la liaison entre les éléments des ensembles de la structure logique et ceux des ensembles de la structure linguistique et des images. La relation $\prec_{comp\_trans}$ qui lie les éléments feuilles de la structure logique aux éléments racines de la structure linguistique a déjà été définie dans le modèle intégré de texte. La relation $\prec_{comp\_trans2}$ assure la liaison entre les éléments feuilles de la structure logique et les éléments racines de la structure des images.

**La relation de séquence** $\prec_{seq}$ décrit les enchaînements entre les éléments de structure du document. Il s'agit d'une relation binaire interne sur $OS$ : $\prec_{seq} \subset OS \times OS$.

Cette relation est uniquement définie sur des éléments de structure logique et des éléments textuels. Nous rappelons que dorénavant un élément de structure logique peut être une image ou contenir une image. Le symbole $OS_{text}$ englobe donc ici tous les éléments de structure logique et les éléments de structure linguistique du texte. Les éléments représentant l'image, c'est-à-dire l'ensemble $OS_{pict}$, sont ici exclus.

$$\prec_{seq} \subset OS_{log} \cup OS_{ling} \times OS_{text} \cup OS_{ling}$$

La définition de cette relation est similaire à celle que nous avons donnée pour la relation de séquence des documents purement textuels du modèle intégré (page 93).

La notation $o_i \prec_{seq} o_j$, avec $o_i, o_j \in OS_{log}$ ou $o_i, o_j \in OS_{ling}$, signifie que l'élément $o_i$ précède directement l'élément $o_j$ dans la séquence.

**Les relations de référence** $\prec_{ref}$ décrivent les références entre des éléments structurels. Il s'agit d'une relation binaire interne sur $OS$ : $\prec_{ref} \subset OS \times OS$.

Nous souhaitons ici accorder de l'importance aux relations entre différents types de médias et permettons donc qu'un élément de structure textuel puisse référer un élément de structure d'une image.

**Les types d'éléments structurels** $TYPE$ indiquent l'ensemble des types d'éléments de structure qui peuvent être rencontrés dans un document structuré.

Cet ensemble correspond à l'union de l'ensembles des types de la structure logique avec l'ensemble des types de la structure linguistique et avec l'ensemble des types de la structure des images.

$$TYPE = TYPE_{log} \cup TYPE_{ling} \cup TYPE_{pict}$$

L'ensemble $TYPE_{log}$ correspond aux types de la structure logique, l'ensemble $TYPE_{ling}$ correspond aux types et de la structure linguistique du texte et l'ensemble $TYPE_{pict}$ correspond aux types de la structure des images.

**La relation** $\preceq_{tcomp}$ définit les compositions possibles entre les types d'éléments structurels. Il s'agit d'une relation binaire interne sur $TYPE$ : $\preceq_{tcomp} \subset TYPE \times TYPE$.

La notation générale $t_i \preceq_{tcomp} t_j$, avec $t_i, t_j \in TYPE$, signifie que l'élément structurel de type $t_i$ peut être composé d'un élément structurel de type $t_j$. Cette relation est contrainte par la provenance des types structurels, soit la structure logique, soit la structure linguistique, soit la structure des images :

$$t_i \preceq_{tcomp} t_j \Leftrightarrow \begin{cases} t_i \prec_{tcomp\_log} t_j & \textbf{si } t_i, t_j \in TYPE_{log} \\ t_i \prec_{tcomp\_ling} t_j & \textbf{si } t_i, t_j \in TYPE^*_{ling} \\ t_i \prec_{tcomp\_trans} t_j & \textbf{si } t_i \in TYPE_{log} \text{ et } t_j \in TYPE^*_{ling} \\ t_i \prec_{tcomp\_pict} t_j & \textbf{si } t_i, t_j \in TYPE_{pict} \\ t_i \prec_{tcomp\_trans2} t_j & \textbf{si } t_i \in TYPE_{log} \text{ et } t_j \in TYPE_{pict} \end{cases}$$

La relation $\prec_{tcomp\_trans}$ a été définie pour le modèle de texte intégré. Nous introduisons ici la relation $\prec_{tcomp\_trans2}$ pour lier les types structurels feuilles de la structure logique et les types structurels racines de la structure des images.

**La fonction d'assignation d'un type** *type* à un élément structurel est une fonction surjective.

$$type : OS \rightarrow TYPE$$

Cette fonction dépend de la provenance des éléments de structure :

$$type(o) = t \Leftrightarrow \begin{cases} t \in TYPE_{log} & \textbf{si } o \in OS_{log} \\ t \in TYPE_{ling} & \textbf{si } o \in OS_{ling} \\ t \in TYPE_{pict} & \textbf{si } o \in OS_{pict} \end{cases}$$

## 5.2 Le modèle de document

Le modèle de document se définit à partir du septuplet suivant permettant de définir la structure d'un document, les médias et les attributs du document ainsi que les relations qui lient ces informations.

$$S = (ST, M, A, \mathcal{T}, \mathcal{R}_{ST}, \mathcal{R}_M) \tag{5.1}$$

Cette définition suppose que nous avons défini les ensembles suivants : $TYPE$ (ensemble des types d'éléments structurels), $NAME$ (ensemble des noms d'attributs) et MEDIA (ensemble des médias du document). Nous revenons sur la description de ces éléments par la suite.

**ST** définit la structure des documents. Cette définition de structure correspond à un squelette sur lequel chaque document structuré doit pouvoir s'intégrer. Ce squelette est une structure syntaxique du document combinant les éléments que nous avons vus dans la modélisation du texte et des images.

**M** décrit les média ainsi que leurs modèles de représentation. Les modèles de représentation reprennent les caractéristiques de chacun des médias. Nous retrouvons ces caractéristiques à travers des attributs particuliers que nous nommons des *attributs des médias*.

**A** définit les attributs décrivant les éléments structurels. Ces informations indiquent à la fois le nom des attributs disponibles ainsi que leur domaine de définition. L'une des particularités de ce modèle est de considérer le domaine des attributs comme *un ensemble d'expression d'un langage d'indexation* et la valeur de l'attribut comme une expression particulière de ce langage d'indexation.

$\mathcal{T}$ lie les médias aux types d'éléments structurels. Il s'agit d'une relation entre un ou plusieurs médias défini dans $M$ et un type d'élément structurel défini dans $ST$. Les ensembles $TYPE$ et $MEDIA$ contiennent respectivement les médias et les types d'éléments structurels.

$$\mathcal{T} \subset MEDIA \times TYPE$$

La relation $\mathcal{T}(m, t)$, avec $m \in MEDIA$ et $t \in TYPE$, signifie que le média $m$ peut apparaître dans un élément structurel de type $t$.

$\mathcal{R}_{\mathbf{ST}}$ lie les attributs à des types d'éléments structurels. Cette relation associe à chaque type d'élément structurel défini dans $ST$ les attributs qui pourront être attachés à ces types. Nous dirons que cette relation définit les *attributs structurels* des documents. L'ensemble $NAME$ contient l'ensemble des noms d'attribut.

$$\mathcal{R}_{ST} \subset TYPE \times NAME$$

La relation $\mathcal{R}_{ST}(t, \alpha)$, avec $t \in TYPE$ et $\alpha \in NAME$, signifie que l'attribut de nom $\alpha$ caractérise les éléments structurels de type $t$. Autrement dit, un élément de type $t$ peut admettre un *attribut structurel* de nom $\alpha$.

$\mathcal{R}_{\mathbf{M}}$ lie les attributs à des médias. Cette relation décrit les attributs de l'ensemble $NAME$ qui sont spécifiques à certains médias. Cette spécificité est donnée par le modèle de représentation spécifique à chaque média. Cette relation définit les *attributs média*.

$$\mathcal{R}_M \subset MEDIA \times NAME$$

La relation $\mathcal{R}_M(m, \beta)$ , avec $m \in MEDIA$ et $\beta \in NAME$, signifie que l'attribut de nom $\beta$ caractérise les éléments structurels du média $m$, c'est-à-dire qu'un élément de média $m$ peut admettre un *attribut média* de nom $\beta$.

## 5.3 Le type de structure

Il s'agit ici du type de structure d'un document structuré complet dont la définition est similaire aux types de structure décrits pour les structures du texte ainsi que pour celles de l'image.

### 5.3.1 Définition

Les informations concernant le type de structure d'un ensemble de document permettent de définir quels types d'éléments structurels apparaissent dans ces documents mais aussi comment ces éléments peuvent se composer entre eux et de quelle manière ils s'enchaînent.

$$ST = (TYPE, \preceq_{tcomp}, \preceq_{tseq}) \tag{5.2}$$

**TYPE** est l'ensemble des types d'éléments structurels qui peuvent apparaître dans le type de document décrit par $ST$.

Nous avons utilisé dans les modèles de représentation du texte et des images des types d'éléments structurels. Il s'agit de ces mêmes types qui sont définis dans l'ensemble $TYPE$ mais de manière générale pour le document complet, c'est-à-dire pour le texte et pour les images.

$\preceq_{\mathbf{tcomp}}$ est la relation qui décrit comment les éléments structurels d'un certain type peuvent se composer, c'est-à-dire quels types d'éléments ils peuvent admettre comme composant. Il s'agit d'une relation binaire interne sur l'ensemble des types : $\preceq_{tcomp} \subset TYPE \times TYPE$.

La notation $t_i \preceq_{tcomp} t_j$, avec $t_i, t_j \in TYPE$ signifie qu'un élément de type $t_i$ peut avoir pour composant direct un élément de type $t_j$.

$\preceq_{\mathbf{tseq}}$ est la relation qui décrit comment les éléments structurels d'un certain type peuvent s'enchaîner dans la séquence. Elle définit quels types d'éléments structurels peuvent succéder à chaque type d'élément structurel. Il s'agit d'une relation binaire interne sur l'ensemble des types : $\preceq_{tseq} \subset TYPE \times TYPE$.

La notation $t_i \preceq_{tseq} t_j$, avec $t_i, t_j \in TYPE$ signifie qu'un élément de type $t_j$ peut être le successeur direct d'un élément de type $t_i$.

Nous développons maintenant les propriétés et contraintes sur les éléments du type de structure.

### 5.3.2 La relation de composition sur les types

Nous avons défini ce type de relation sur les types d'éléments structurels dans les documents textuels et sur les types d'éléments structurels dans les images fixes. Il s'agit d'une relation similaire puisque cette relation indique les types d'éléments qui peuvent se composer. A travers cette relation, nous confondons la composition des types d'éléments de la structure logique d'un document, les types d'éléments spécifiques aux documents textuels et les types d'éléments des images fixes que nous avons précédemment définis.

Il apparaît donc que nous ne pouvons énoncer de propriété générale de réflexivité ou de non réflexivité sur cette relation. Il existe des éléments structurels qui peuvent être composés par des éléments de même type que le leur, par exemple dans le modèle d'image, un élément scène peut être composée d'un autre élément scène. Les propriétés de cette relation dépendent du contexte et particulièrement de la vérification ou non de la réflexivité sur les types concernés. Ainsi, les propriétés de non transitivité et d'asymétrie ne peuvent être énoncées que sur les types sur lesquels la propriété de non réflexivité est vérifiée. La non transitivité de la relation est toujours vérifiée. Par contre, l'asymétrie et la réflexivité dépendent des types mis en jeu. Lorsqu'il n'y a pas réflexivité, l'asymétrie est elle aussi vérifiée mais la réflexivité implique le non respect d'une propriété d'asymétrie. Ces remarques prennent toute leur importance quand on sait que le processus de recherche peut utiliser les propriétés de cette relation.

### 5.3.3 La relation de séquence sur les types

Comme nous l'avons défini pour les documents textuels, cette relation indique quels types peuvent se succéder (et donc se précéder) dans l'ordonnancement linéaire des éléments structurels du document. Dans cette relation, seuls les types d'éléments de la structure logique et ceux spécifiques aux documents textuels sont concernés puisque nous n'avons pas jugé utile de donner un ordonnancement linéaire pour les éléments de l'image.

Comme pour la relation précédente, les propriétés que nous pouvons énoncer sur cette relation dépendent de la possibilité ou non pour un élément d'un type donné de pouvoir précéder ou succéder directement un autre élément de même type (réflexivité). Par exemple, un paragraphe peut être suivi d'un autre paragraphe et l'application de cette relation sur le type paragraphe est réflexive.

Dans le cas où la non réflexivité est vérifiée, nous pouvons énoncer les propriétés de *non transitivité* et d'*asymétrie*.

## 5.4 Les médias

Nous considérons des documents structurés mêlant des images et du texte. Nous abordons donc ici la représentation spécifique de ces médias dans le modèle de document structuré que nous proposons. Nous ne donnons pas explicitement les modèles de chacun des médias mais

à travers cette démarche nous voulons indiquer que les caractéristiques spécifiques à chaque média doivent être présentes dans la représentation complète du document structuré. Il s'agit par exemple d'indiquer que les attributs physiques d'une image peuvent être pris en compte dans notre modèle aussi bien qu'un attribut de contenu sémantique.

Par ailleurs nous posons comme hypothèse que les caractéristiques qui sont communes aux modèles monomédias sont représentées par des attributs de même nom.

$$M = (MEDIA, \mathcal{M}, model) \tag{5.3}$$

**MEDIA** est l'ensemble des types de média que nous autorisons dans nos documents. Nous aurions par exemple l'ensemble suivant pour des médias non temporels: $MEDIA = \{texte, image\}$.

$\mathcal{M}$ est un ensemble de modèles monomédia qui peuvent s'appliquer pour décrire chacune des données monomédia d'un document.

Dans la littérature, il existe de nombreuses manières de décrire chacune des données monomédia, que ce soit les données textuelles [SG83, Sav90, Par96], les données graphiques [Mec95a] ou bien les images fixes [CR90, CDY90, KKT94, Meg95, Mec95c]. Nous avons donné dans le chapitre précédent sur la représentation des images fixes, la plupart des éléments de représentation du média image.

Le modèle permet en fait de déterminer quelles sont les caractéristiques particulières de chacun des médias et de donner une vision structurée de ces caractéristiques. C'est en effet à partir de ces modèles que nous déterminons les attributs spécifiques à chaque média, par exemple les attributs perceptifs pour l'image ou encore les relations spatiales.

**model** est une fonction totale assignant à chaque média un modèle particulier de $\mathcal{M}$.

$$model : MEDIA \rightarrow \mathcal{M}$$

Nous définissons ici des modèles pour des éléments monomédias. Cependant, dans le document structuré, les éléments structurels peuvent être composés à partir de plusieurs médias. Il n'existe pas un modèle de représentation particulier pour ces éléments structurels dits "multimédia". Leur représentation sera assurée par la représentation de leurs composants monomédias.

Nous gardons une certaine indépendance vis-à-vis des médias qui sont utilisés dans le type de document structuré. Cette indépendance nous permet de garder le modèle ouvert pour l'accueil de nouveaux médias.

## 5.5 Les attributs du document structuré

Les attributs des éléments sont définis par le triplet $A$ suivant:

$$A = (NAME, DOMAIN, domain) \tag{5.4}$$

**NAME** est l'ensemble des noms d'attributs disponibles.

**DOMAIN** est l'ensemble des valeurs possibles des attributs, c'est-à-dire le domaine de définition des attributs. En fait, nous considérons qu'un domaine de définition correspond à un ensemble d'expression d'un langage donné. Ce langage, aussi appelé langage d'indexation, a sa propre syntaxe et sa propre sémantique.

Le fait de considérer que le domaine est un ensemble d'expressions d'un langage d'indexation nous permet d'englober la majorité des attributs que nous rencontrons dans les différentes applications. Ainsi, nous pouvons considérer des domaines de type base de données telles que les entiers ou les chaînes de caractères mais aussi des langages plus complexes tels que ceux utilisés dans les systèmes de recherche d'information : ensemble de mots-clés, vecteurs de termes, graphes conceptuels, logiques terminologiques, etc.

**domain** est une fonction partielle qui associe à chaque nom d'attributs de l'ensemble $NAME$ un domaine de définition particulier de l'ensemble $DOMAIN$.

$$domain : NAME \rightarrow DOMAIN$$

## 5.6 La base de documents

Nous avons donné les spécifications structurelles des documents dans les sections précédentes. Nous allons maintenant nous intéresser aux documents qui doivent vérifier ces spécifications. Nous allons donc d'une part donner la représentation du document structuré, puis nous donnerons celle de la base de documents.

Nous pouvons d'ores et déjà noter que la représentation du document est similaire à celle que nous avons donnée pour les instances de document textuel (page 73 et page 77) et d'images fixes (page 99).

### 5.6.1 Le document structuré

Le document structuré se définit comme un assemblage d'éléments structurels typés issus de différents médias et auxquels des attributs les décrivant sont rattachés.

$$D = (OS, \prec_{comp}, \prec_{seq}, \prec_{ref}, type, media, value) \tag{5.5}$$

**OS** est l'ensemble des éléments structurels qui donnent la représentation d'un document structuré et de chacun de ses fragments, au niveau de sa structure syntaxique (structure logique, structure linguistique et structure interne des images).

Nous appelons élément structurel un élément $o_i$ de l'ensemble $OS$. Chaque fragment du document structuré est représenté par un élément structurel $o_i$.

**$\prec_{comp}$** est la relation de composition entre les éléments structurels de l'ensemble $OS$. Il s'agit d'une relation binaire interne sur l'ensemble $OS$ : $\prec_{comp} \subset OS \times OS$.

La notation $o_i \prec_{comp} o_j$, avec $o_i, o_j \in OS$, signifie que l'élément structurel $o_i$ est composé directement par l'élément structurel $o_j$. Cette relation est contrainte par la relation de composition sur les types d'élément structurels du type de structure auquel se conforme le document.

$\prec_{\mathbf{seq}}$ est la relation de séquence entre les éléments structurels de l'ensemble $OS$. Il s'agit d'une relation binaire interne sur l'ensemble $OS$ : $\prec_{seq} \subset OS \times OS$.

La notation $o_i \prec_{seq} o_j$, avec $o_i, o_j \in OS$, signifie que l'élément structurel $o_i$ a pour successeur direct l'élément structurel $o_j$. Cette relation est contrainte par la relation de séquence sur les types d'élément structurels du type de structure auquel se conforme le document.

$\prec_{\mathbf{ref}}$ est la relation de référence entre les éléments structurels de l'ensemble $OS$. Il s'agit d'une relation binaire interne sur l'ensemble $OS$ : $\prec_{ref} \subset OS \times OS$.

La notation $o_i \prec_{ref} o_j$, avec $o_i, o_j \in OS$, signifie que l'élément structurel $o_i$, *l'élément source*, réfère l'élément structurel $o_j$, *l'élément cible*. Nous avons donné dans le modèle de représentation des documents textuels 3.3.3 la définition de ces relations de référence en établissant le passage de ces relations syntaxiques vers des relations sémantiques, c'est-à-dire en donnant le sens de telles relations. Il s'agit le plus souvent de descriptions complémentaires sur l'élément cible de la relation qui peuvent se traduire par des transferts d'informations.

**type** est la fonction assignant à chaque élément structurel un type d'élément structurel.

$$type : OS \rightarrow TYPE$$

L'ensemble des types d'élément, noté $TYPE$, est donné par la relation liant le document à un type de document structuré. Cette relation est décrite au niveau de la base de document.

**media** est la fonction qui indique quels sont les médias qui sont contenus dans un élément structurel. Un élément structurel peut utiliser plusieurs média, il s'agit alors d'un élément multimédia. Dans le cas où la fonction retourne un ensemble singleton, il s'agit d'un élément monomédia.

$$media : OS \rightarrow 2^{MEDIA}$$

La notation $media(o_i) = M_i$, avec $o_i \in OS$ et $M_i = \{m_j\}$ et $\forall m_j \in M_i$ on a $m_j \in MEDIA$, signifie que l'élément structurel $o_i$ utilise chacun des médias $m_j$.

**value** est la fonction qui fournit la valeur d'un attribut pris dans l'ensemble $NAME$ pour un élément structurel de l'ensemble $OS$.

$$value : NAME \times OS \rightarrow domain(\alpha)$$

Nous utilisons par la suite l'abréviation suivante $value_\alpha$ pour la fonction qui fournit la valeur de l'attribut $\alpha$. La notation $value(\alpha, o_i) = a$, équivalente à $value_\alpha(o_i) = a$, avec

$o_i \in OS$, $a \in domain(\alpha)$ et $\alpha \in NAME$, signifie que l'attribut $\alpha$ de l'élément $o_i$ a pour valeur $a$ [1].

Par rapport à ce que nous avons décrit dans les chapitres précédents sur les documents textuels et les images fixes, nous introduisons dans $D$ les fonctions *media* et *value*. Ces deux fonctions permettent d'intégrer dans le modèle de document structuré et donc dans la représentation de chaque document les différents médias ainsi que les descripteurs sous forme d'attributs.

### 5.6.2 La base de documents structurés

Nous définissons une base de documents structurés comme un ensemble de spécifications de types de documents structurés associé à un ensemble de documents structurés. Un document structuré doit se conformer à l'un des types de documents de la base pour appartenir à cette base de documents. Nous définissons donc la base de documents structurés par le triplet suivant que nous appelons $\mathcal{B}$.

$$\mathcal{B} = (\mathcal{S}, \mathcal{D}, \zeta) \tag{5.6}$$

$\mathcal{S}$ est l'ensemble des types de structure que comporte la base de documents.

$$\mathcal{S} = \{ST_i\}$$

Les $ST_i$ sont des triplets comportant les informations relatives à un type de structure particulier, c'est-à-dire une classe de documents.

$\mathcal{D}$ est l'ensemble des documents structurés de la base de document. Ces documents sont représentés par les informations que nous avons données dans la section précédente.

$$\mathcal{D} = \{D_i\}$$

$\zeta$ est la relation qui établit le lien entre un document structuré de $\mathcal{D}$ et un type de structure de l'ensemble $\mathcal{S}$.

$$\zeta \subset \mathcal{S} \times \mathcal{D}$$

La notation $\zeta(ST_i, D_j)$, avec $ST_i \in \mathcal{S}$ et $D_i \in \mathcal{D}$, signifie que le document structuré $D_j$ se conforme au type de structure défini par $ST_i$, c"est-à-dire que les éléments structurels du document $D_j$ prennent leurs types dans l'ensemble $TYPE$ de $ST_i$ et que la composition et l'enchaînement de ces éléments sont respectivement contraints par les relations $\preceq_{tcomp}$ et $\preceq_{tseq}$ de $ST_i$.

Un document structuré $D_j$ de la base *se conforme à un et un seul type de structure* décrit dans $\mathcal{S}$. Nous exprimons ceci de la manière suivante :

$$\forall D_j \in \mathcal{D},\ \exists\,!\ ST_i \in \mathcal{S} \text{ tel que } \zeta(ST_i, D_j)$$

1. Nous utiliserons aussi la notation $value_\alpha(o_i) = o_i.\alpha = a$.

### 5.6.3 Contraintes

Soit le document $D_i$ de la base de document $\mathcal{B}$ dans laquelle la relation de conformité entre le type de structure $S_i$ et $D_j$ est vérifiée : $\zeta(S_i, D_j)$. Ce type de structure contient les éléments suivants : $S_i = (ST_i, M_i, A_i, \mathcal{T}, \mathcal{R}_{ST}, \mathcal{R}_M)$.

#### a) Contraintes sur les types

La fonction d'assignation du type à chaque élément structurel du document s'établit dorénavant avec l'ensemble des types structurels du type de structure $S_i$. Cet ensemble est pris dans le triplet définissant $ST_i$ : $ST_i = (TYPE_i, \preceq_{tcomp}, \preceq_{tseq})$. Ainsi, la fonction *type* se définit de la manière suivante :

$$type : OS \rightarrow TYPE_i$$

Cela signifie que les éléments structurels de ce document prennent leurs types dans $TYPE_i$, le type de structure auquel le document se conforme.

#### b) Contraintes sur la composition

La relation de composition décrit les compositions directes d'éléments structurels dans la hiérarchie qui représente la structure du document. Cette composition est contrainte par la relation définie dans $ST_i$ qui indique comment les types d'éléments structurels peuvent se composer.

$$\forall o_i, o_j \in OS \ : \ o_i \prec_{comp} o_j \ \textbf{ssi} \ type(o_i) \preceq_{tcomp} type(o_j)$$

#### c) Contraintes sur la séquence

La relation de séquence décrit les enchaînements des éléments structurels qui déterminent le sens linéaire de lecture. Ces enchaînements sont contraints par la relation définie dans $ST_i$ qui indique quels enchaînements de types d'éléments structurels sont autorisés dans le document.

$$\forall o_i, o_j \in OS \ : \ o_i \prec_{seq} o_j \ \textbf{ssi} \ type(o_i) \preceq_{tseq} type(o_j)$$

#### d) Contraintes sur les médias

L'ensemble des médias pris en considération pour la fonction *media* du document structuré est donné dans le triplet $M_i$ qui décrit les éléments monomédias du type de document $S_i$. Ce triplet contient les informations suivantes : $M_i = (MEDIA_i, \mathcal{M}, model)$. La fonction se définit donc de la manière suivante :

$$media : OS \rightarrow 2^{MEDIA_i}$$

Par ailleurs, cette fonction est contrainte par la relation $\mathcal{T}$ qui intervient au niveau des types d'éléments structurels.

Soit l'élément structurel $o_i$ de l'ensemble $OS$. Cet élément structurel a pour type $t_i$ et utilise les médias de l'ensemble $Med_i$. Les relations suivantes indiquant que le type $t_i$ admet les médias contenus dans $Med_i$ doivent être vérifiées :

Soit $o_i \in OS$
tel que $type(o_i) = t_i$ avec $t_i \in TYPE_i$
et $media(o_i) = Med_i$ avec $Med_i \subseteq MEDIA_i$
Alors $\forall m \in Med_i$ on a $\mathcal{T}(m, t_i)$

A partir de la composition des éléments structurels, nous devons vérifier les relations suivantes sur la fonction *media* :

Un élément structurel $o_i$ composé d'un élément structurel $o_j$ comporte au moins les médias utilisés par l'élément qui est le composant, c'est-à-dire $o_j$ .

$$\forall o_i, o_j \in OS \ : \ o_i \prec_{comp} o_j \text{ alors } media(o_j) \subset media(o_i)$$

Un élément structurel $o$ composé des éléments structurels $o_i$ comporte l'union des médias utilisés par les éléments structurels qui sont les composants, c'est-à-dire les $o_i$.

$$\text{Soit } o \in OS, \ \forall o_i \in OS \ : \ o \prec_{comp} o_i \text{ alors } media(o) = \bigcup_i media(o_i)$$

### e) Contraintes sur les attributs

Dans notre modèle les attributs proviennent soit des types d'éléments structurels, soit des médias. Considérons tout d'abord les attributs provenant des types d'éléments structurels, c'est-à-dire donnés par la relation $\mathcal{R}_{ST}$, puis ceux provenant des médias, c'est-à-dire donnés par la relation $\mathcal{R}_M$.

Nous rappelons tout d'abord que le type de structure $S_i$ comporte des informations spécifiques aux attributs dans le triplet $A_i$ : $A_i = (NAME, DOMAIN, domain)$.

*Attributs structurels*

Les attributs structurels sont définis sur des types d'éléments structurels particuliers par la relation $\mathcal{R}_{ST}$. Par exemple, c'est cette relation qui indique si un élément de type chapitre admet un attribut auteur.

L'application de la fonction *value* suit donc les contraintes fournies par cette relation : pour que l'attribut $\alpha$ admette une valeur au niveau de l'élément structurel $o_i$, il faut que le type de cet élément admette l'attribut $\alpha$ comme descripteur.

Soit $o_i \in OS$
tel que $type(o_i) = t_i$ avec $t_i \in TYPE_i$
et $\mathcal{R}_{ST}(t_i, \alpha)$ avec $\alpha \in NAME$
Alors $value_\alpha(o_i) = \alpha_i$ avec $\alpha_i \in domain(\alpha)$

*Attributs des médias*

Les attributs des médias sont définis sur chacun des médias et c'est en fonction des médias utilisés par un élément de structure que nous pouvons déterminer quels attributs peuvent décrire un élément de structure. Par exemple, un élément structurel comportant une image et du texte sera décrit par les attributs provenant des deux médias.

Soit $o_i \in OS$
tel que $media(o_i) = \{m_j\}$ avec $m_j \in MEDIA_i$
et $\mathcal{R}_M(m_j, \beta)$ avec $\beta \in NAME$
Alors $value_\beta(o_i) = \beta_i$ avec $\beta_i \in domain(\alpha)$

Il faut noter que le choix de considérer qu'un attribut décrivant un élément monomédia puisse aussi décrire un élément multimédia comporte certaines limites. Le choix de valuer ou non ces attributs au niveau des éléments multimédia va fortement dépendre de la nature des attributs.

Considérons les deux médias image et texte ainsi que des attributs provenant de ces médias : la couleur et le contenu sémantique pour l'image et le contenu sémantique pour le texte. Considérons maintenant deux cas de figure : dans le premier cas, nous avons un élément structurel $o_x$ composé d'un élément comportant uniquement une image et d'un élément comportant du texte et dans le second cas, nous avons un élément structurel $o_y$ composé de deux éléments comportant chacun uniquement une image :

1. D'après la définition que nous avons donnée des attributs provenant des médias, l'élément structurel $o_x$ admet les attributs définis sur le média image et sur le média texte puisqu'il utilise ces médias : $media(o_x) = \{texte, image\}$.

   Ceci signifie que l'élément $o_x$ pourra être décrit par un attribut couleur et un attribut contenu (contenu sémantique) puisqu'il contient une image. Et il pourra aussi être décrit par un attribut contenu (contenu sémantique) puisqu'il contient du texte.

   Nous considérerons ici un unique attribut contenu pour décrire le contenu sémantique de l'élément $o_x$. Nous ne discutons pas pour l'instant de la valeur que pourrait prendre cet attribut mais uniquement du fait que cet attribut doive décrire l'élément structurel multimédia $o_x$.

2. Toujours d'après la définition que nous avons donnée des attributs provenant des médias, l'élément structurel $o_y$ composé de deux images admet les attributs définis sur le média image : $media(o_y) = \{image\}$.

   L'élément $o_y$ pourra donc, par définition, être décrit par les deux attributs couleur et contenu spécifiques aux images. Toutefois, dans ce cas précis, nous pouvons nous demander quel va être le sens d'un attribut indiquant la couleur sur un élément structurel comportant deux images.

## 5.7 Synthèse

Nous représentons à l'aide de notre modèle le document de la figure 5.1, la première page d'un quotidien d'informations extraite d'un site Web [2]. Nous établissons tout d'abord les caractéristiques du type de structure de ce document, c'est-à-dire la définition des types d'éléments de structure ainsi que les compositions et enchaînements possibles de ces types d'éléments. Nous donnons ensuite quelques exemples d'attributs définis sur ces types d'éléments de structure. Enfin, nous décrivons une partie de la représentation du document structuré présentée dans la figure 5.1. Dans cette représentation, nous explicitons les éléments de structure, leurs types, les relations qui les lient ainsi que leur contenu.


**Figure 5.1.** Exemple de Document - version électronique

### 5.7.1 Les types de structure et leurs relations

Le document que nous présentons ici en tant qu'exemple est extrait d'un quotidien d'information. Nous décrivons le type journal à travers le triplet $ST$ :

$ST_{journal} = (TYPE_{journal}, \preceq_{tcomp}, \preceq_{tseq})$

Soit l'ensemble $TYPE_{journal}$ contenant les types d'éléments structurels que nous retrouvons dans les documents de type journal. Nous donnons ici uniquement les types extraits de

2. Site web du journal Libération : http://www.liberation.com du 23 Septembre 1997

la structure logique de ces documents.

$TYPE_{journal}$ ={Journal, Titre, SousTitre, Date, Editorial, CorpsJournal, LaUne, LesColonnes, Illustration, Paragraphe, Rubrique, Manchettes, Article, CorpsArticle, Bloc, ...}

La relation $\preceq_{tcomp}$ indiquant les compositions directes possibles entre types d'éléments structurels se caractérise alors par les informations suivantes de la table suivante :

TAB. 5.1 – *Relations de composition d'un journal*

| Commentaires | Relations de Composition |
|---|---|
| Un Journal est composé d'un Titre, d'une Date, d'un Editorial et du corps du journal, CorpsJournal | Journal $\preceq_{tcomp}$ Titre<br>Journal $\preceq_{tcomp}$ Date<br>Journal $\preceq_{tcomp}$ Editorial<br>Journal $\preceq_{tcomp}$ CorpsJournal |
| Un Editorial est composé d'une Une du journal, LaUne, et de colonnes, LesColonnes | Editorial $\preceq_{tcomp}$ LaUne<br>Editorial $\preceq_{tcomp}$ LesColonnes |
| La une du journal est composée d'un Titre, d'un Sous-Titre, d'une Illustration et d'un Paragraphe | LaUne $\preceq_{tcomp}$ Titre<br>LaUne $\preceq_{tcomp}$ SousTitre<br>LaUne $\preceq_{tcomp}$ Illustration<br>LaUne $\preceq_{tcomp}$ Paragraphe |
| Les Colonnes du journal sont composées de Rubrique | LesColonnes $\preceq_{tcomp}$ Rubrique |
| Une Rubrique est composée de Titre et de Manchettes | Rubrique $\preceq_{tcomp}$ Titre<br>Rubrique $\preceq_{tcomp}$ Manchettes |
| Une manchette est composée d'un Titre et d'un Paragraphe | Manchettes $\preceq_{tcomp}$ Titre<br>Manchettes $\preceq_{tcomp}$ Paragraphe |
| Le corps du journal, CorpsJournal est composé d'Articles | CorpsJournal $\preceq_{tcomp}$ Article |
| Un article est composé d'un Titre, d'un Sous-Titre (optionnel), du corps de l'article, CorpsArticle, de Références et d'informations complémentaires sur l'article, les Compléments | Article $\preceq_{tcomp}$ Titre<br>Article $\preceq_{tcomp}$ SousTitre<br>Article $\preceq_{tcomp}$ CorpsArticle<br>Article $\preceq_{tcomp}$ Références<br>Article $\preceq_{tcomp}$ Compléments |
| Le corps d'un article est composé de Titre et de Bloc | CorpsArticle $\preceq_{tcomp}$ Bloc<br>CorpsArticle $\preceq_{tcomp}$ Titre |
| Un bloc est composé d'un Paragraphe ou d'une Illustration | Bloc $\preceq_{tcomp}$ Paragraphe<br>Bloc $\preceq_{tcomp}$ Illustration |
| Une Référence est composée d'un paragraphe | Référence $\preceq_{tcomp}$ Paragraphe |
| Un complément d'article est composé d'un Titre (optionnel) et de Paragraphe | Compléments $\preceq_{tcomp}$ Titre<br>Compléments $\preceq_{tcomp}$ Paragraphe |

La relation $\preceq_{tseq}$ qui détermine les enchaînements possibles entre types d'éléments se caractérise alors par les informations suivantes :

TAB. 5.2 – *Relations de séquence d'un journal*

| Commentaires | Relations de Séquence |
|---|---|
| Un titre de journal est suivi d'une date, puis d'un éditorial et enfin du corps du journal | Titre $\preceq_{tseq}$ Date<br>Date $\preceq_{tseq}$ Editorial<br>Editorial $\preceq_{tseq}$ CorpsJournal |
| La une du journal est suivie des colonnes | LaUne $\preceq_{tseq}$ LesColonnes |
| Le titre de la une du journal est suivi d'un sous-titre, puis d'une illustration et d'un paragraphe | Titre $\preceq_{tseq}$ SousTitre<br>SousTitre $\preceq_{tseq}$ Illustration<br>Illustration $\preceq_{tseq}$ Paragraphe |
| Une rubrique des colonnes peut être suivie d'une rubrique | Rubrique $\preceq_{tseq}$ Rubrique |
| Un titre de rubrique est suivi d'une manchette qui peut être suivie d'une manchette | Titre $\preceq_{tseq}$ Manchettes<br>Manchettes $\preceq_{tseq}$ Manchettes |
| Un titre de manchette est suivi d'un paragraphe qui peut être suivi d'un paragraphe | Titre $\preceq_{tseq}$ Paragraphe<br>Paragraphe $\preceq_{tseq}$ Paragraphe |
| Dans un corps d'article, un titre peut être suivi par un bloc, et un bloc peut être suivi par un titre | Titre $\preceq_{tseq}$ Bloc<br>Bloc $\preceq_{tseq}$ Titre |

### 5.7.2 Liaison avec les médias

Parmi les types d'éléments structurels logiques que nous avons présentés, nous donnons la liaison avec les médias(le texte et l'image fixe) pour les types d'éléments feuilles de la structure logique. La liaison pour les types des éléments non feuilles est donnée par la fonction *media* et la composition de ces éléments.

TAB. 5.3 – *Liaison avec les médias*

| Commentaires | $\mathcal{T}$ |
|---|---|
| Le média image est présent uniquement dans les éléments de type Illustration | $\mathcal{T}(image, Illustration)$ |
| Le média texte est celui qui est utilisé par tout les types d'éléments structurels autres que Illustration | $\mathcal{T}(texte, Titre)$<br>$\mathcal{T}(texte, Date)$<br>$\mathcal{T}(texte, SousTitre)$<br>$\mathcal{T}(texte, Paragraphe)$ ... |

### 5.7.3 Les attributs

Nous allons définir les attributs des éléments structurels que nous avons précédemment décrits. Ces attributs sont des descripteurs classiques dont nous donnons aussi le domaine de définition. Nous définissons ici les informations contenues dans le triplet $A_{journal} : A_{journal} = (NAME_{journal}, DOMAIN, domain)$. Nous décrivons quelques éléments de l'ensemble des noms d'attributs, $NAME_{journal}$, ainsi que leurs rattachements aux types d'éléments structurels via la relation $\mathcal{R}_{ST}$.

TAB. 5.4 – *Attributs d'un journal*

| Commentaires | Attribut | $\mathcal{R}_{ST}$ |
|---|---|---|
| Un journal est décrit par un directeur de publication, une date, ... | DirPub | $\mathcal{R}_{ST}(Journal, DirPub)$ |
| | Date | $\mathcal{R}_{ST}(Journal, Date)$ |
| Une illustration est une image prise par un photographe | Auteur | $\mathcal{R}_{ST}(Illustration, Auteur)$ |
| Un éditorial et les articles sont écrits par un ou plusieurs journalistes | Auteur | $\mathcal{R}_{ST}(Editorial, Auteur)$ |
| | | $\mathcal{R}_{ST}(Article, Auteur)$ |

### 5.7.4 Représentation du document

Nous allons donner une partie de la représentation du document présenté dans la figure 5.1 de la page 126. Nous donnons ici uniquement la structure logique de ce document.

Nous considérons que le document complet est représenté par l'élément structurel $o$ de type Journal. Cet élément est la racine de la hiérarchie structurelle. Nous donnons dans la figure qui suit (figure 5.2) quelques-uns des éléments structurels qui sont des composants de l'élément $o$.

- Soit l'élément structurel $o_2$ de type Titre qui est un composant direct de l'élément $o$ : $o \prec_{comp} o_2$.
- Soit l'élément structurel $o_3$ de type SousTitre qui est un composant direct de l'élément $o$ : $o \prec_{comp} o_3$, et qui est le successeur direct de l'élément $o_2$ : $o_2 \prec_{seq} o_3$.
- Soit l'élément structurel $o_4$ de type LaUne qui est un composant direct de l'élément $o$ : $o \prec_{comp} o_4$, et qui est le successeur direct de l'élément $o_3$ : $o_3 \prec_{seq} o_4$. Cet élément est composé d'autres éléments logiques :
  - Soit l'élément structurel $o_{41}$ de type Titre qui est un composant direct de l'élément $o_4$ : $o_4 \prec_{comp} o_{41}$.
  - Soit l'élément structurel $o_{42}$ de type SousTitre qui est un composant direct de l'élément $o_4$ : $o_4 \prec_{comp} o_{42}$, et qui est le successeur direct de l'élément $o_{41}$ : $o_{41} \prec_{seq} o_{42}$.

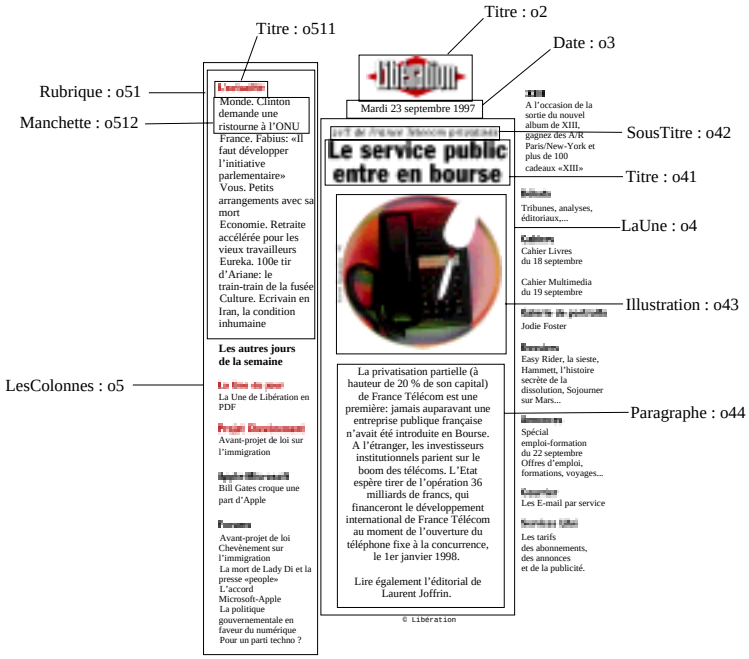



**Figure 5.2.** Représentation d'un Document Structuré

- Soit l'élément structurel $o_{43}$ de type Illustration qui est un composant direct de l'élément $o_4$ : $o_4 \prec_{comp} o_{43}$, et qui est le successeur direct de l'élément $o_{42}$ : $o_{42} \prec_{seq} o_{43}$.
- Soit l'élément structurel $o_{44}$ de type Paragraphe qui est un composant direct de l'élément $o_4$ : $o_4 \prec_{comp} o_{44}$, et qui est le successeur direct de l'élément $o_{43}$ : $o_{43} \prec_{seq} o_{44}$.

- Soit l'élément structurel $o_5$ de type LesColonnes qui est un composant direct de l'élément $o$: $o \prec_{comp} o_5$, et qui est le successeur direct de l'élément $o_4$: $o_4 \prec_{seq} o_5$. Comme le montre la figure 5.3, cet élément structurel est composé d'autres éléments de type Rubrique, Titre, Manchettes, etc.

Conformément au modèle que nous proposons, nous devons aussi considérer les autres relations de structure c'est-à-dire les relations de référence présentes dans ce document. La plupart des éléments de structure de ce document présente la particularité de référer un autre élément structurel de ce même document. En effet, l'exemple que nous considérons correspond à une première page de journal qui contient les "titres" des articles contenus dans le document complet. Par exemple, l'élément structurel correspondant à la "Une" du journal réfère des articles du journal correspondant à d'autres éléments structurels dans la représentation du document. En fait, cette page est extraite d'un site web et elle comporte des liens hypertextuels qui pour notre modèle de représentation peuvent être assimilés à des relations de références. Ainsi chaque élément de type Manchette réfère un article du document

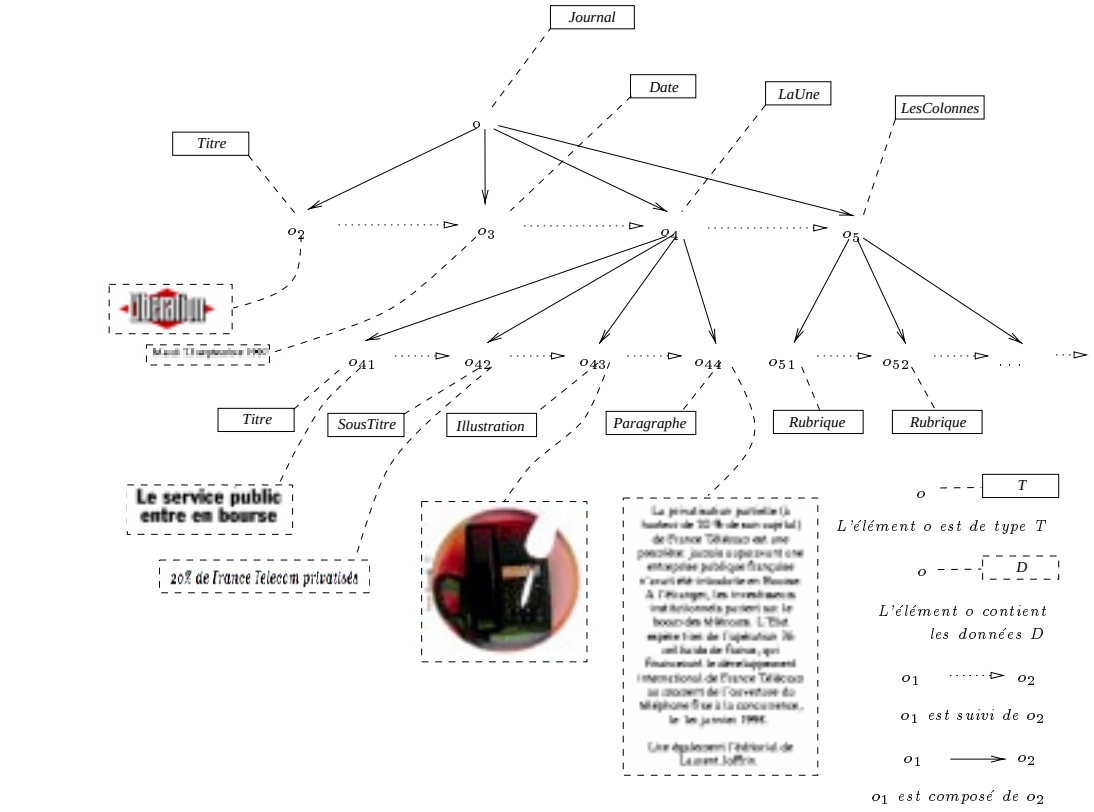



**Figure 5.3.** Arborsecence du Document Structuré

complet. Si nous ne considérons pas explicitement de liens hypertextuels dans notre modèle, la relation de référence est un moyen pour les représenter.

### 5.7.5 Conclusion

Pour conclure ces chapitres consacrés à la modélisation des documents structurés mêlant du texte et des images fixes, nous avons donné un exemple de document structuré que nous représentons à l'aide de notre modèle. Ce modèle nécessite la définition d'un type de structure comportant des types d'éléments structurels liés entre eux. Ces types proposent à la fois de représenter les abstractions de la structure logique du document structuré, la structure linguistique du texte ainsi que la structure interne des objets de l'image.

Le type de structure de notre modèle contient aussi des informations liées aux médias utilisés dans le document. Il permet de spécifier quels types d'éléments structurels acceptent quels médias. D'autre part, ce modèle propose une représentation uniforme des descripteurs de ces éléments structurels à travers les attributs. Nous proposons des attributs dont les domaines se confondent avec des langages d'indexation venant de la recherche d'information. Il est ainsi possible d'avoir des attributs classiques propres aux bases de données qui cohabitent avec des attributs donnant une représentation du contenu sémantique de chaque élément

et qui utilisent des formalismes de représentation de la connaissance plus avancés (graphes conceptuels, logiques descriptives, etc).

Ces attributs contiennent les informations qui peuvent permettre de retrouver un document, que ce soit une date, un auteur ou encore le contenu sémantique de l'élément structurel. A partir de cette représentation, chaque élément structurel décrit par un attribut est un élément retrouvable, c'est-à-dire qui peut répondre à un besoin formulé dans la requête d'un utilisateur.

# Troisième partie

# Impact de la structure du document

*Nous avons présenté dans la partie précédente un modèle de représentation des documents structurés admettant des parties textuelles et des images fixes. Dans ce modèle, l'organisation des parties des documents, appelées éléments de structure, s'articule autour de trois relations : la composition, la séquence et la référence.*

*Le modèle proposé est à la base du travail qui suit, à savoir l'impact et l'usage que nous allons faire de ces relations dans le cadre d'un système de recherche d'informations sur un fond constitué de documents structurés.*

*Pour définir l'impact de la structure sur le processus de recherche, nous identifions les lacunes actuels des approches traditionnelles et nous les caractérisons par deux notions élémentaires : la couverture des éléments de structure par les informations descriptives et les dépendances entre ces informations. Nous cherchons alors à tirer profit des propriétés des relations de structure de notre modèle pour pallier à ces lacunes. Il s'agit là de l'usage que nous faisons de la structure.*

*Cette étude nous amène à définir dans le chapitre 6 un processus d'indexation spécifique aux documents structurés qui peut être vu comme la prise en compte de mécanismes de propagation et de combinaison des informations descriptives des éléments de structure.*

*Dans le chapitre 7, nous montrons comment nous tirons bénéfice de ce processus lors de l'interrogation d'une base constituée de documents structurés.*

*Enfin, dans le chapitre 8, nous cherchons à valider notre approche par le développement de l'application my Personal Daily News. Cette application permet d'interroger une base de quotidiens d'informations dont l'indexation et l'interrogation reprennent les mécanismes développés dans les chapitre 6 et 7.*

# Chapitre 6

# Indexation structurelle

"Le désordre c'est l'ordre vu diffèrement."

HYREN 283 av.JC

Définir l'impact de la structure du document sur le processus de recherche d'informations, c'est connaître les propriétés induites par les relations qui organisent cette structure (relation de composition, relation de séquence et relation de référence) afin de les utiliser lors du processus d'indexation dans le but d'obtenir des index plus expressifs.

Il s'agit dans ce chapitre de préciser comment utiliser de l'information disponible dans le document afin d'enrichir le processus d'indexation.

La vision du processus d'indexation est alors changée puisqu'il sera guidé par la structure du document. Nous nommons ce processus *l'indexation structurelle*. Nous avons indiqué dans le modèle de document structuré que les attributs correspondent à certaines formes d'indexation, en particulier l'attribut de contenu, et qu'ils sont associés aux éléments de structure qui représentent les parties de document. Nous montrons sur un exemple comment mettre en évidence les propriétés des attributs qui sont induites par la structure et nous introduisons la notion de *portée des attributs* pour expliciter ces propriétés. La portée d'un attribut caractérise un ensemble d'éléments concernés par cet attribut. C'est à travers la notion de portée que nous allons propager les informations contenues dans les attributs et c'est donc la portée qui guidera la propagation de l'information dans la structure considérée.

Nous présentons plus en détail les différents types de portée provenant des trois relations de structure issues du modèle de document. Puis nous nous attardons sur la résolution des conflits issus de l'application des portées d'un même attribut au sein d'un document structuré.

## 6.1 Problématique générale

Notre problématique générale concerne l'indexation des documents structurés, et donc la représentation des informations décrivant ce type de documents.

Notre objectif n'est pas la description d'un langage d'indexation mais la mise en place

d'une technique d'indexation qui puisse accueillir les caractéristiques des documents structurés afin d'utiliser la liaison entre la structure et les informations exprimées par le document : contenu sémantique et/ou attributs externes. Nous donnons tout d'abord un exemple qui nous permet de présenter certaines lacunes des systèmes traditionnels et la problématique générale à laquelle nous sommes confrontés. Par la suite, nous présentons la notion d'attribut que nous adoptons dans notre modèle de représentation du document structuré. Il s'agit d'une vision élargie et enrichie de l'attribut par rapport à l'approche traditionnellement rencontrée dans les normes de représentation des documents structurés ou dans les SGBD.

### 6.1.1 Exemple Introductif: un document SGML

A travers cet exemple, nous montrons l'intérêt des attributs dans les documents structurés mais aussi leur limite actuelle due principalement à la non prise en compte des caractéristiques structurelles lors de la définition de ces attributs.

Nous allons considérer un type de document structuré décrit par une DTD SGML donné partiellement dans la figure 6.1. Nous présentons cet exemple avec la syntaxe SGML car elle présente l'avantage d'être facilement compréhensible pour décrire les types d'éléments structurels et les informations descriptives données sous la forme d'attributs au niveau de chacun de ces types.

Il s'agit ici de documents de type livre comportant un titre (type titreLivre) et un ou plusieurs chapitres (type Chapitre) qui sont eux-mêmes composés d'un titre (type titreChapitre) et d'une ou plusieurs sections (type Section).

```
1. <! DOCTYPE Livre [
2. <! ELEMENT Livre - - (titreLivre,Chapitre+)>
3. <! ELEMENT titreLivre - - (#PCDATA)>
4. <! ELEMENT Chapitre - - (titreChapitre,Section+)>
5. <! ATTLIST Chapitre
        auteur NMTOKEN #REQUIRED
        date NMTOKEN #REQUIRED
        status (final | brouillon) brouillon #REQUIRED
        numCh NMTOKEN #REQUIRED
6. <! ELEMENT titreChapitre - - (#PCDATA)>
7. <! ELEMENT Section - - (titreSection,Paragraphe+)>
... ] >
```

**Figure 6.1.** DTD d'un document de type livre

Dans la figure 6.1 (ligne 5) le type structurel chapitre possède quatre attributs : auteur, date, status et numCh. Ces attributs permettent de caractériser les éléments de structure de type chapitre. Pour chaque chapitre d'un document qui se conforme à ce DTD, nous connaîtrons le ou les auteurs, la date de création, l'état du chapitre et le numéro du chapitre dans le document complet.

### 6.1.2 Application de notre modèle à l'exemple

Nous transposons une partie de la DTD de la figure 6.1 dans le modèle de représentation que nous avons défini. Nous donnons ici quelques-uns des éléments du modèle de document, à savoir l'ensemble des types structurels, les relations entre ces types, l'ensemble des noms d'attributs ainsi que les attributs associés à chacun de ces types conformément à la DTD.

L'ensemble des types structurels est défini par les éléments de l'ensemble $TYPE_{ST}$ :

$$TYPE_{ST} = \{ \quad Livre, titreLivre, Chapitre, titreChapitre, Section, \\ titreSection, Paragraphe, \dots \}$$

La relation de composition, notée $\preceq_{tcomp}$, sur les types d'éléments structurels de l'ensemble $TYPE_{ST}$ détermine quels types d'éléments structurels peuvent se composer ($t_1 \preceq_{tcomp} t_2$ signifie qu'un élément de type $t_2$ peut être un composant d'un élément de type $t_1$) :

$$\begin{array}{ll} Livre \preceq_{tcomp} titreLivre & Livre \preceq_{tcomp} Chapitre \\ Chapitre \preceq_{tcomp} titreChapitre & Chapitre \preceq_{tcomp} Section \\ Section \preceq_{tcomp} titreSection & Section \preceq_{tcomp} Paragraphe \end{array}$$

La relation de séquence, notée $\preceq_{tseq}$, sur les types d'éléments structurels $TYPE_{ST}$ détermine quels types d'éléments structurels peuvent se succéder dans la séquence ($t_1 \preceq_{tseq} t_2$ signifie qu'un élément de type $t_2$ peut être le successeur séquentiel d'un élément de type $t_1$) :

$$\begin{array}{ll} titreLivre \preceq_{tseq} Chapitre & Chapitre \preceq_{tseq} Chapitre \\ titreChapitre \preceq_{tseq} Section & Section \preceq_{tseq} Section \\ titreSection \preceq_{tseq} Paragraphe & Paragraphe \preceq_{tseq} Paragraphe \end{array}$$

Les attributs dont les noms sont regroupés dans l'ensemble $NAME$ :

$$NAME = \{auteur, date, status, numCh\}$$

Parmi les noms de l'ensemble $NAME$ nous décrivons par la relation $\Re_{ST}$ les *attributs structurels*, c'est-à-dire les attributs attachés à un type d'élément structurel. Comme l'indique la DTD de la figure 6.1, les quatre attributs de l'ensemble $NAME$ sont des attributs structurels rattachés aux éléments de type Chapitre.

$$\begin{array}{ll} \Re_{ST}(Chapitre, auteur) & \Re_{ST}(Chapitre, date) \\ \Re_{ST}(Chapitre, status) & \Re_{ST}(Chapitre, numCh) \end{array}$$

On remarque que nous ne retrouvons pas dans notre modèle l'ensemble des contraintes définies dans la DTD. Par exemple, la notion d'alternative entre deux types structurels (notée | dans SGML) qui est présente dans une DTD n'est pas représentée dans notre modèle car elle ne présente pas dans notre optique de travail une information immédiatement pertinente. Nous avons défini un modèle capable de représenter l'instance du document et non sa spécification.

Notre modèle permet donc de spécifier moins d'information qu'une DTD SGML puisque son rôle n'est pas la description de documents structurés mais un usage particulier des documents structurés : leur recherche.

### 6.1.3 Exemples de recherche

Dans les systèmes s'intéressant à la recherche de documents représentés à l'aide de SGML [BCK+94, Pep95, FFE96, Chr96, ACC+97], la notion d'attribut est utilisée telle qu'elle est définie au niveau de la DTD. Il n'est donc pas possible de retrouver directement des éléments de type Section à partir de leur auteur puisque l'attribut auteur porte uniquement sur les éléments de type chapitre. Dans la définition de type de document, il n'existe pas de spécification particulière permettant de dire si l'information représentée par l'attribut auteur peut caractériser ou non les éléments structurels qui sont liés par des relations de structure aux éléments structurels de type chapitre.

Prenons l'exemple d'une requête dans laquelle l'utilisateur spécifie qu'il cherche un paragraphe écrit par un auteur particulier. Dans les systèmes adoptant une approche traditionnelle des attributs, que ce soit des systèmes de recherche documentaire [Tha90] ou des systèmes de gestion de base de données [Chr96, ACC+97], la recherche va s'effectuer uniquement sur les éléments de type paragraphe. Or d'après la définition du type de document, ce type d'élément n'admet pas cet attribut. Soit le système ne répond pas à la requête, c'est à dire qu'il retourne un ensemble vide de réponses, soit il décide d'étendre la requête selon des règles étendant la requête en tenant compte des compositions structurelles (premier ascendant structurel d'une section admettant l'attribut requis). Dans le premier cas (ensemble vide), la réponse fournie ne peut pas satisfaire l'utilisateur : il peut exister des paragraphes qui composent un chapitre qui a été écrit par l'auteur requis. Le silence vient ici d'une non exploitation des relations de structure et d'une trop faible couverture des éléments du document par l'attribut auteur qui est uniquement défini sur les chapitres. Dans le second cas, c'est le processus de traitement de la requête qui prend en charge la dérivation et l'élargissement de la requête selon des règles pré-définies. La fonction de mise en correspondance exécute des traitements à cause d'une défaillance de l'indexation.

Etudions maintenant un second exemple montrant l'ambiguïté des attributs dans le document structuré pour les systèmes de recherche documentaire. Considérons qu'un utilisateur souhaite retrouver *l'ensemble des livres qui sont à l'état de brouillon.* Nous voyons dans la définition du type de structure des livres que seuls les éléments de type chapitre admettent un attribut status indiquant son état. Afin de traiter cette requête, plusieurs types de stratégies peuvent se présenter pour répondre :

- Un livre est à l'état de brouillon si et seulement si au moins un chapitre est à l'état de brouillon.
- Un livre est à l'état de brouillon si et seulement si tout les chapitres sont à l'état de brouillon.
- Un livre est à l'état de brouillon si et seulement si un certain pourcentage de ses chapitres sont à l'état de brouillon.
- L'attribut n'est pas défini sur le type Livre, il est donc impossible de répondre.

Ce même type d'interrogation peut survenir lorsqu'un utilisateur recherche un élément de type section ou paragraphe en fonction de son état, brouillon ou final.

Il est difficile d'exprimer jusqu'à quel point un élément structurel tel que le livre est concerné par la valeur des attributs de ses composants, en d'autres termes quelle est la portée des attributs lorsque celle-ci n'est pas explicitée. Cet exemple très simple montre que les choix sont nombreux et que la décision dépend pleinement du type de structure et de l'information contenue dans l'attribut. Le processus de recherche est donc totalement dépendant de ces décisions puisque c'est lui qui les traite.

Par ailleurs dans cet exemple se pose le problème de la combinaison des informations provenant de différentes sources : un livre comporte plusieurs chapitres et chaque chapitre admet un attribut indiquant son état. Afin de déterminer dans quel état nous devons considérer, les composants et/ou les composés de chacun de ces chapitres, il faut que nous combinions les valeurs provenant des différentes sources pour exprimer leur état, c'est-à-dire pour donner une valeur à cet attribut.

Nous avons montré à travers cet exemple certains problèmes récurrents à la recherche des documents structurés. La majeure partie de ces difficultés réside dans la non atomicité du document structuré qui disperse l'information et la rend donc plus difficilement accessible.

Nous allons maintenant envisager une solution pour faire face à ces problèmes en nous intéressant particulièrement au processus de traitement du document qui est préalable au processus de recherche, à savoir le processus d'indexation. Pour cela, nous revenons sur les états du document au cours de ce processus.

## 6.2 Les états du document structuré

Un point fondamental qui, au premier abord, différencie une base de données d'un système de recherche d'informations, est la phase préliminaire de traitement des données. Une base de donnée, une fois modélisée, est utilisée telle quelle alors qu'un système de recherche d'informations nécessite un traitement des documents communément appelé *indexation*. Ce traitement restitue sous une forme donnée, conforme à un modèle de documents, les informations contenues dans chaque document.

Dans notre approche, les informations à propos du document ou de ces parties et les informations décrivant le document ou ces parties sont contenues dans des attributs attachés aux éléments de structure. Il nous reste à définir quelle forme nous allons donner au processus d'indexation afin de rendre l'information, et donc les documents, plus accessible.

Nous reprenons tout d'abord la définition initiale de l'attribut en explicitant les lacunes de ceux-ci dans les systèmes existant et en donnant leurs caractéristiques liées à la structure des documents. Nous montrons ainsi la nécessité d'une extension de cette notion d'attribut dans le document structuré. Par la suite, nous revenons sur le processus d'indexation et les différents états dans lesquels nous pouvons considérer les documents. En introduisant les notions de *couverture* et de *dépendance*, nous signifions le besoin d'un nouvel état qui offrirait une meilleure adéquation entre le document structuré et les requêtes en intégrant une extension à la notion d'attribut.

### 6.2.1 L'attribut : nécessité d'une extension

La notion générale d'attribut se définit comme une caractéristique ou une propriété qui est propre à quelqu'un ou à quelque chose. Dans le domaine des bases de données (modèle relationnel ou à objets), un attribut permet de représenter une caractéristique partagée par un ensemble d'individus (ou d'éléments).

On remarque que dans les bases de données, un attribut est explicitement déclaré sur une relation ou sur une classe d'objets et sa valuation représente une information explicite qui caractérise un élément d'une relation ou bien un objet. De même, dans les normes de représentation des documents structurés tels que SGML [Bur94] ou ODA [Ass91], la notion d'attribut permet de caractériser les éléments de structure. Les attributs permettent de caractériser chaque noeud d'une arborescence structurelle selon une définition fournie par le type de structure (DTD pour SGML). Cependant dans ces normes de représentation des documents structurés ou dans les systèmes gérant des documents structurés [Pep95], si les attributs sont explicitement déclarés sur des éléments de structure, il reste une part d'information implicite puisqu'il faut déterminer pour chaque attribut quels sont les éléments de structure qui sont réellement concernés par l'information contenue dans cet attribut.

Les exemples de recherche que nous avons donnés précédemment (section 6.1.3) montrent bien l'ambiguïté des attributs dans le document structuré. Cet ambiguïté réside dans le manque d'information sur la *qualification* même de chaque attribut.

D'une part nous souhaitons conserver la notion d'attributs sur les éléments de structure du document. Ces attributs permettent de décrire les caractéristiques particulières de chaque partie de document et restent donc utiles pour la recherche. Cependant nous voulons étendre cette notion afin de ne plus être confrontés lors de l'interrogation à des ambiguïtés sur les éléments concernés par une information, c'est-à-dire accroître la puissance du processus de recherche. Nous souhaitons définir lors de l'indexation structurelle quels éléments sont décrits par quelles informations et comment les informations contenues dans les attributs se combinent à l'intérieur du document structuré.

Nous voulons remédier à ces problèmes en ne considérant plus seulement un attribut mais aussi les propriétés structurelles de l'attribut afin de mieux couvrir le document et de résoudre au niveau du modèle de représentation des problèmes liés à la représentation de l'information.

Nous proposons donc de qualifier les attributs lors du processus d'indexation structurelle, c'est-à-dire en amont du processus de recherche. Nous adoptons donc une *indexation statique* que nous pouvons opposer à une *indexation dynamique* telle que celle de Paradis dans [Par96]. Dans une indexation dynamique, c'est la requête qui détermine quelles informations vont être considérées et comment elles vont se comporter. Nous préférons définir en amont ces comportements.

### 6.2.2 De l'indexation à l'interrogation

Nous revenons ici sur l'indexation et son importance vis-à-vis de l'interrogation qui est la finalité de tout système de recherche d'informations. Le processus d'interrogation, et donc le modèle de correspondance sous-jacent, repose complètement sur le résultat de l'indexation et la forme de celle-ci. Dans la figure 6.2, nous montrons que différentes formes d'indexation im-

pliquent différentes formes de correspondance si nous voulons aboutir à un résultat équivalent en sortie.

Dans cette figure, l'état initial du document est un document sur lequel aucune analyse particulière n'a été appliquée. Il s'agit donc d'un document comportant uniquement des descriptions externes assimilables à des attributs classiques de type base de données. Dans l'état final, nous présentons deux types de documents qui comportent chacun des informations supplémentaires par rapport à l'état initial. Ces informations, issues d'une phase d'indexation, ne sont pas similaires et peuvent être plus ou moins riches selon les caractéristiques de chacune des indexations.

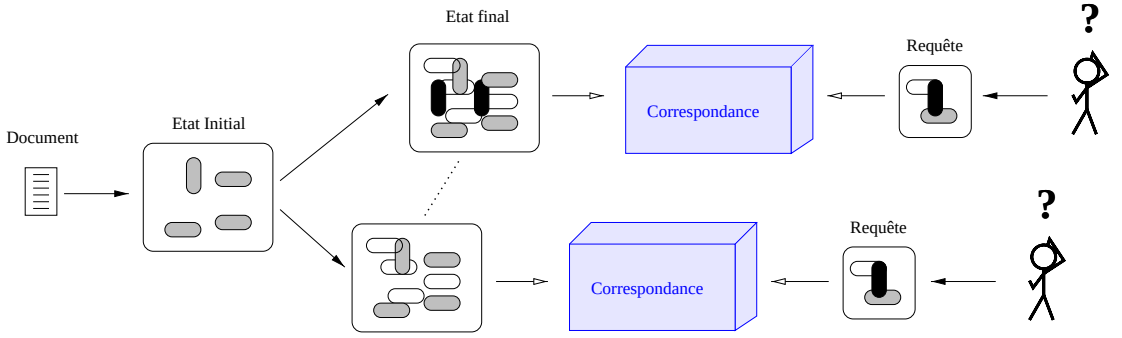



**Figure 6.2.** Dépendance entre indexation et correspondance

Il ne s'agit pas pour nous de définir l'indexation la plus riche possible, mais de nous situer vis-à-vis des indexations que nous connaissons actuellement sur des documents "plats" et d'adapter celle-ci à des documents structurés. Sachant que la structure des documents introduit une nouvelle complexité, il est préférable de calculer préalablement, c'est-à-dire lors de l'indexation, les nouveaux éléments introduits par la structure. Parmi ces éléments, nous retrouvons les dépendances entre les descriptions des éléments de structure au sein du document, c'est-à-dire la *provenance des informations* et *leur circulation dans la structure du document*.

Nous pouvons citer en exemple des travaux s'étant intéressés à définir des stratégies d'indexation qui par la suite guidaient la recherche au sein du document structuré. Ainsi Lee & al. [LYYB96] proposent une indexation basée sur des ensembles de mots clés. Les feuilles du document sont décrites par un ensemble de mots clés et ceux-ci remontent vers le noeud racine du document en s'agrégeant pour fournir pour chaque noeud de l'arborescence un index. A partir de cette stratégie d'indexation, différentes sortes de fichiers inverses sont établies. Des propriétés des fichiers inverses, c'est-à-dire quelles informations sont conservées et précalculées dans ceux-ci, vont dépendre les réponses mais aussi les espaces requis pour les index et les temps d'accès. Une autre approche basée sur le modèle logique de recherche d'information et proposée par Kheirbek et Chiaramella dans [KC95, Khe95] démontre la possibilité de décrire une stratégie de recherche dans les documents structurés dépendant complètement des propriétés fournies par l'indexation. Cette stratégie de recherche nommée "Fetch and Browse", repose entièrement sur le fait que l'indexation fournit une implication logique entre la description d'un composé et celle de son composant. Cette stratégie permet de retrouver

l'élément de structure le plus petit qui répond à une requête.

Il nous semble donc primordial lors de notre processus d'indexation de fournir en résultat de ce processus non seulement une bonne description du contenu mais surtout d'établir les relations entre les descriptions des éléments de structure afin de pouvoir guider le processus de recherche.

Au niveau de la recherche sur des documents structurés et des caractéristiques propres à cette recherche, à savoir retrouver l'élément de structure et non plus le document complet qui répond à une requête, nous introduisons deux notions qui permettent de décrire l'objectif du processus d'indexation de tels documents : la *couverture* et la *dépendance*.

**La couverture** des éléments structurels du document par les attributs a pour but d'assurer une meilleure adéquation entre l'ensemble des informations qui décrivent le document et l'ensemble des éléments de structure qui représentent le document structuré.

*Une couverture optimale se caractérise par un état du document structuré dans lequel chaque élément structurel accueille tout attribut qui contient de l'information le concernant.*

Considérons un document conforme à la DTD présentée dans la figure 6.1. Dire que ce document admet une couverture optimale signifie que seuls les éléments de type Chapitre sont concernés par l'attribut auteur. Ce postulat n'est pas défendable puisque cet attribut concerne notamment le document complet : le livre. Il en va de même pour la plupart des autres attributs au sein d'un document structuré. Il faut donc que chaque information concernant un ou plusieurs éléments de structure soit représentée explicitement par un attribut sur ces éléments de structure.

**La dépendance** des valeurs des attributs dans un document structuré s'exprime par des dépendances explicites entre informations de même nature signifiant clairement qu'il s'agit d'une même source d'information qui est représentée au niveau des différents éléments de structure.

*La dépendance des valeurs des attributs dans un document structuré est réalisée si les dépendances entre les valeurs des attributs sont explicitées.*

Nous avons défini les notions de *couverture* et de *dépendance* comme des objectifs de l'indexation des documents structurés. Une couverture optimale offre la possibilité de retrouver un maximum d'éléments structurels d'après leurs descriptions puisque celles-ci accueilleront chaque information relative à l'élément. Cette couverture minimise les dérivations et traitements de la phase de correspondance. La dépendance des valeurs des attributs dans le document structuré indique d'une part, que dans ce document certains éléments sont des sources d'information et d'autre part, qu'elle reflète l'organisation des informations au sein du document.

Considérons maintenant un document pour lequel la couverture n'est pas optimale et dont la dépendance entre les valeurs des attributs n'est pas exprimée dans l'état final de représentation. Lors du traitement d'une requête, soit le système va faillir par manque de connaissance sur la portée d'un attribut au sein du document, soit il va devoir, au cours du processus de correspondance, dériver cette connaissance. L'indexation n'ayant pas pris

en charge les notions élémentaires que nous avons définies, c'est le processus de mise en correspondance du système qui va être confronté à ces lacunes.

Nous proposons donc de reporter cette phase de dérivation des connaissances dans un processus complémentaire du processus d'indexation classique. Nous l'appellerons le processus d'*indexation structurelle*. Il aura en charge de rapprocher l'état du document final d'une couverture optimale pour celui-ci en explicitant les dépendances entre les valeurs des attributs. L'explicitation de ces dépendances sera alors un outil pour guider la recherche dans le document structuré. La figure 6.3 reflète l'objectif de ce processus par rapport à un état final du document structuré.

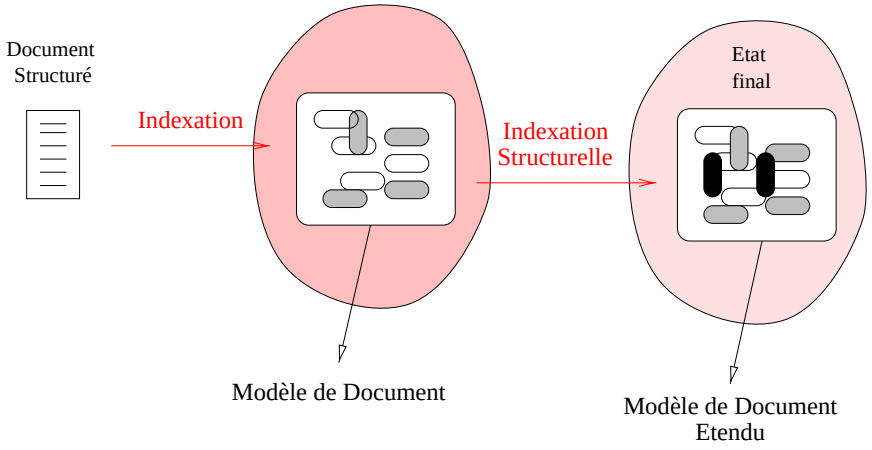



**Figure 6.3.** L'objectif d'une indexation des documents structurés

Cet état final réalise une couverture optimale et donne une vision explicite des dépendances entre les valeurs des attributs de chaque élément de structure.

Nous avons introduit figure 6.3 la notion de *modèle de document étendu*, c'est-à-dire un modèle de document auquel doit se conformer le résultat de l'indexation structurelle. L'indexation structurelle modifie la couverture du document en associant à des éléments de structure des descripteurs qui n'existaient pas auparavant sur ces éléments. Ce processus provoque donc des incohérences entre le modèle de document initial et le résultat qu'il fournit. Nous devons donc construire un nouveau modèle de document en fonction des paramètres de l'indexation structurelle.

Dans ce modèle de document étendu, les éléments nouveaux qui proviennent spécifiquement de l'indexation structurelle seront spécifiés. Nous voyons par la suite comment ce modèle de document étendu est obtenu.

## 6.3 Portée de l'attribut

Deux notions essentielles nous guident dorénavant pour définir la phase d'indexation structurelle des documents structurés : la *couverture* du document par les attributs et la *dépendance*

des valeurs des attributs.

Dans ce but, nous introduisons une nouvelle caractéristique propre aux attributs des documents structurés : la portée de l'attribut. Cette portée reflète l'extension qui permet de combler les lacunes que nous avons présentées en 6.2.1. De plus, cette nouvelle caractéristique des attributs a pour but d'obtenir une meilleure couverture du document par les attributs ainsi qu'une explicitation des dépendances entre valeurs d'attributs au sein du document structuré. Ces deux points sont les éléments que nous mettons en évidence lors d'un processus d'indexation structurelle.

Nous définissons tout d'abord la portée de l'attribut, puis nous introduisons une définition formelle de cette portée dans les documents structurés avant de l'appliquer aux relations de structure de notre modèle de représentation du document structuré.

### 6.3.1 Une propriété structurelle de l'attribut: sa portée

Nous avons vu à travers le court exemple donné précédemment que l'interrogation des documents structurés serait plus puissante en exploitant la portée des attributs dans les documents structurés. Ces attributs sont définis pour un noeud de l'arborescence structurelle sans tenir compte de l'environnement de ce noeud. La non-prise en compte de leur environnement structurel introduit des ambiguïtés et donc des lacunes au niveau de l'interrogation. Nous proposons une définition de l'attribut dans le document structuré qui lève ces ambiguïtés en définissant plus exactement quelle information est liée à l'attribut et quels éléments du document sont concernés par cette information.

Notre objectif est d'expliciter les informations décrivant les éléments de structure car nous considérons qu'il s'agit d'une phase primordiale pour faciliter la recherche dans les documents structurés. Le principal problème auquel nous sommes confrontés lors de la recherche sur des documents structurés concerne le manque de flexibilité des systèmes d'interrogation dû pour une grande part à un trop grand nombre d'informations implicites.

Afin d'utiliser les propriétés des relations de structure lors de la recherche, nous introduisons la notion de *portée d'un attribut*. Cette notion de portée doit être vue comme un moyen de rendre explicite un maximum d'informations contenues dans le document et de les rendre accessibles à tous les niveaux du document pour lesquels elles sont vérifiées. Cette approche tend donc vers la couverture et la dépendance des attributs des documents structurés.

Nous rappelons que nous considérons un état initial du document structuré représenté par un ensemble d'éléments structurels organisé en arborescence. Certains de ces éléments de structure sont décrits par des attributs qui proviennent soit de la définition du type de structure (attributs structurels), soit de la représentation du contenu de l'élément (attribut de contenu). Lorsqu'un attribut $\alpha$ décrit un élément de structure dans l'état initial du document, nous dirons que *l'attribut $\alpha$ est attaché à un élément de structure.*

L'utilisation de la portée d'un attribut modifie l'état du document structuré indexé, i.e. l'ensemble d'éléments structurels et les attributs attachés à ces éléments. On obtient un document structuré représenté non seulement par l'ensemble des éléments structurels et les attributs qui y sont attachés, mais aussi par des *attributs assignés à des éléments structurels* selon les indications de la portée des attributs. Les attributs assignés sont des attributs provenant des attributs initiaux et rendent explicites les informations implicites de document

structuré. L'introduction de la portée des attributs dans le processus d'indexation du document structuré nous permet d'augmenter la couverture des éléments structurels du document par les attributs et nous allons voir avec la définition plus précise de la portée que nous cherchons aussi à exprimer les dépendances entre les valeurs d'attributs à travers la portée.

### 6.3.2 Définition d'une portée

Nous donnons une définition de la portée d'un attribut $\alpha$ pour un élément de structure $o$ d'un document $D$.

**Définition 3 (Portée d'un attribut)** *La portée d'un attribut $\alpha$ décrivant un élément structurel $o$ d'un document $D$ est décrite par le septuplet $\mathcal{P}^o_{\alpha,rel}$. Ce septuplet détermine l'ensemble des éléments structurels, noté $EP^o_{\alpha,rel}$, qui sont décrits par ce même attribut $\alpha$ et qui partagent la même source d'information (l'élément structurel $o$).*

En décrivant une portée pour un attribut $\alpha$ d'un élément de structure $o$, nous indiquons que cet attribut ne concerne pas uniquement l'élément de structure $o$ mais qu'il décrit aussi des éléments liés à l'élément $o$ par une relation de structure. La définition de la portée correspond à une meilleure couverture du document par les attributs puisque chaque élément structurel de l'ensemble $EP^o_{\alpha,rel}$ va désormais accueillir un attribut $\alpha$.

Par ailleurs, nous indiquons la dépendance entre les valeurs des attributs $\alpha$ : les éléments structurels de l'ensemble $EP^o_{\alpha,rel}$ vont admettre un attribut $\alpha$ dont la valeur provient d'une même source d'information, l'élément structurel $o$. Nous retrouvons ici la notion de dépendance dans les valeurs des attributs et la définition d'éléments structurels en tant que sources d'informations.

Nous donnons ci-dessous (6.1), les composants qui forment le septuplet $\mathcal{P}^o_{\alpha,rel}$ :

$$\mathcal{P}^o_{\alpha,rel} = (\alpha, o, rel, cat, v_\alpha, f_\alpha, cond) \tag{6.1}$$

**L'attribut** $\alpha$ est un nom d'attribut pris dans l'ensemble des noms d'attribut $NAME$, défini pour un type de structure de documents pris dans $ST$ via la relation $\mathfrak{R}_{ST}$ ou bien défini pour un type de média pris dans $M$ via la relation $\mathfrak{R}_M$.

**L'élément structurel o** appartient à l'ensemble $OS$ des éléments structurels qui décrivent un document $D$ se conformant à un type de structure de $ST$.

**La relation rel** est une relation de structure. Nous l'appliquons aux relations de structure définies dans notre modèle de document structuré : la relation de composition, $\prec_{comp}$, la relation de séquence $\prec_{seq}$ et la relation de référence $\prec_{ref}$.

L'ensemble qui décrit la portée de l'attribut $\alpha$ pour l'élément de structure $o$ est établi en suivant cette relation, c'est-à-dire que les éléments de l'ensemble $EP^o_{\alpha,rel}$ sont liés directement ou indirectement par cette relation.

**La valeur $\mathbf{v}_\alpha$** est la valeur initiale de l'attribut $\alpha$, c'est-à-dire la valeur source qui va être partagée par les attributs $\alpha$ des éléments structurels de la portée, $EP^o_{\alpha,rel}$.

$$v_\alpha \in domain(\alpha)$$

**la fonction $f_\alpha$** est une fonction de propagation des valeurs de l'attribut $\alpha$ permettant d'assigner une valeur aux attributs $\alpha$ des éléments de la portée. Les domaines de définition des attributs sont assimilés à des langages et cette fonction est donc différente selon chaque type de langage.

$$f_\alpha : domain(\alpha) \rightarrow domain(\alpha)$$

**La catégorie cat** est une catégorie de propagation prise dans l'ensemble $\{succ, pred, static\}$. Elle indique dans quel sens la relation *rel* doit être suivie afin de construire l'ensemble décrivant la portée [1].

1. *succ*: l'élément $o$ est considéré comme la source de la relation et les éléments successeurs selon *rel* de l'élément $o$ sont des candidats potentiels de la portée définie sur $o$.

$$cat = succ \Rightarrow \forall o_i \in EP^o_{\alpha,rel},\ o_i \in Next_{rel}(o)$$

2. *pred*: l'élément $o$ est considéré comme la cible de la relation et les éléments prédécesseurs selon *rel* de l'élément $o$ sont des candidats potentiels de la portée définie sur $o$.

$$cat = pred \Rightarrow \forall o_i \in EP^o_{\alpha,rel},\ o_i \in Prev_{rel}(o)$$

3. *static*: la portée définie à partir de l'élément $o$ ne concerne pas les éléments liés à $o$ selon la relation *rel*. Cette catégorie nous ramène à des attributs classiques de type base de données qui ne prennent pas en compte les caractéristiques structurelles du document.

**Les conditions cond** expriment l'ensemble de conditions que doivent remplir les éléments structurels pour appartenir à la portée. Nous appelons les conditions spécifiées dans *cond* des *conditions d'appartenance à la portée.*

Hormis la fonction $f_\alpha$ et la valeur initiale $v_\alpha$ qui sont utilisées pour la propagation des valeurs au sein de la portée, les autres informations du septuplet sont indispensables pour la construction de l'ensemble $EP^o_{\alpha,rel}$, c'est-à-dire pour la construction de la couverture.

Nous voyons maintenant quelles conditions d'appartenance les éléments de cet ensemble peuvent vérifier.

### a) Les conditions

Parmi les conditions d'appartenance, certaines sont indépendantes du domaine d'application alors que d'autres en dépendent. Nous donnons quelques exemples de condition d'appartenance indépendantes du domaine, dépendantes du domaine et mêlant les deux :

1. Si on veut indiquer que la distance de l'élément $o_i$ par rapport à l'élément $o$ selon la relation *rel* doit être inférieure à n, on écrit : $distance_{rel}(o, o_i) < n$. Il s'agit d'une

1. Nous rappelons que l'ensemble des prédécesseurs d'un élément selon la relation *rel* est donné par $Prev_{rel}(o)$ et l'ensemble des successeurs par $Next_{rel}(o)$

condition indépendante du domaine d'application car elle peut être appliquée sur tout type de documents structurés.

La distance par rapport à l'élément $o$ se calcule en fonction du nombre de fois où la relation $rel$ est traversée pour atteindre $o$.

$$distance_{rel}(o, o_i) = n \Leftrightarrow |Prev_{rel}(o_i) \Leftrightarrow Prev_{rel}(o)| = n$$

Une condition indiquant que le niveau de profondeur est égal à 1 avec la relation de composition et la catégorie $succ$ signifie que la portée de l'attribut $\alpha$ pour l'élément $o$ est constituée de l'ensemble des descendants structurels directs de $o$.

2. Si on veut que les éléments $o_i$ ne soient pas des éléments structurels de type résumé, on écrit : $type_{str}(o_i) \neq$ résumé. Il s'agit d'une condition dépendante du domaine d'application car relative à un type d'élément structurel.

3. Pour spécifier que les éléments $o_i$ ne doivent pas appartenir à un ensemble d'éléments structurels spécifié par $C$, on écrit : $o_i \notin C$. Il s'agit d'une condition dépendante du domaine d'application et qui est principalement guidée par le contexte du document.

4. Pour spécifier que les éléments structurels $o_i$ ne doivent pas être de type résumé et que la distance à l'élément $o$ des éléments $o_i$ doit être inférieure à n, on écrit : $type_{str}(o_i) \neq$ résumé $\wedge\, distance_{rel}(o, o_i) < n$.

   Dans ce cas, les deux conditions réunies vont fournir l'ensemble des successeurs (ou prédécesseurs) de l'élément $o$ qui sont à une distance inférieure à n et qui sont de type résumé. Il s'agit ici d'une condition d'appartenance mixant une condition dépendante du domaine et une condition indépendante du domaine.

Nous donnons la notation générale suivante avec une fonction notée $cond$ ($cond : OS \rightarrow Boolean$) et qui retourne un booléen indiquant si un élément structurel vérifie ou non les conditions de la portée :

$\forall o_k \in OS$,
**Si** $cond(o_k) = vrai$ **Alors** $o_k$ est un candidat potentiel à l'ensemble $EP^o_{\alpha,rel}$
**Si** $cond(o_k) = faux$ **Alors** $o_k \notin EP^o_{\alpha,rel}$

### b) Construction de $EP^o_{\alpha,rel}$

Nous résumons ici les conditions que doit vérifier un élément $o_k$ pour appartenir à l'ensemble $EP^o_{\alpha,rel}$ d'après les informations fournies dans $\mathcal{P}_{\alpha,rel}$. Nous considérons les trois types de catégorie, $succ$, $pred$ et $static$, indépendamment des relations de structure.

**Catégorie succ** : l'ensemble $EP^o_{\alpha,rel}$ se définit à partir des informations suivantes :

$$\mathcal{P}^o_{\alpha,rel} = (\alpha, \mathbf{o}, \mathbf{rel}, \mathbf{succ}, v_\alpha, f_\alpha, \mathbf{cond})$$

et reprend l'ensemble des successeurs de $o$ selon la relation $rel$ vérifiant les conditions exprimées dans les conditions d'appartenance :

$$EP^{o}_{\alpha,rel} = \{o_k \mid o_k \in Next_{rel}(o) \text{ et } cond(o_k) = vrai\}$$

**Catégorie pred** : l'ensemble $EP^{o}_{\alpha,rel}$ se définit à partir des informations suivantes :

$$\mathcal{P}^{o}_{\alpha,rel} = (\alpha, \mathbf{o}, \mathbf{rel}, \mathbf{pred}, v_\alpha, f_\alpha, \mathbf{cond})$$

et reprend l'ensemble des prédécesseurs de $o$ selon la relation $rel$ en vérifiant les conditions exprimées dans les conditions d'appartenance :

$$EP^{o}_{\alpha,rel} = \{o_k \mid o_k \in Prev_{rel}(o) \text{ et } cond(o_k) = vrai\}$$

**Catégorie static** : l'ensemble $EP^{o}_{\alpha,rel}$ se définit à partir des informations suivantes :

$$\mathcal{P}^{o}_{\alpha,rel} = (\alpha, \mathbf{o}, \mathbf{rel}, \mathbf{static}, v_\alpha, f_\alpha, \mathbf{cond})$$

et indique que la portée est vide, c'est-à-dire qu'aucun autre élément de structure que $o$ n'est concerné par l'information contenue dans $\alpha$. Les informations sur la relation de structure, la fonction d'accès, la fonction de propagation et les conditions d'appartenance ne sont donc pas nécessaires puisqu'elles restent inutilisées.

$$EP^{o}_{\alpha,rel} = \emptyset$$

### c) La fonction de propagation : $f_\alpha$

La fonction $f_\alpha$ propage la valeur initiale de la portée, notée $v_\alpha$, vers tous les éléments de la portée en suivant la relation de structure qui supporte cette portée. Les éléments de l'ensemble $EP^{o}_{\alpha,rel}$ ne sont pas obligatoirement liés directement par la relation $rel$ mais ils sont liés par une relation déduite de $rel$ que nous notons $rel_\alpha$. Cette relation reprend les mêmes propriétés que la relation $rel$ mais elle tient compte des conditions d'appartenance exprimées dans $\mathcal{P}^{o}_{\alpha,rel}$. Les éléments mis en relation par $rel_\alpha$ vérifient tous les conditions d'appartenance.

$$rel_\alpha(o, o_k) \Leftrightarrow \begin{cases} o_k \in Next_{rel}(o) \\ \wedge \\ cond(o_k) = vrai \\ \wedge \\ \nexists\, o_i \in Next_{rel}(o) \text{ tel que } rel_\alpha(o, o_i) \text{ et } rel_\alpha(o_i, o_k) \end{cases}$$

La relation $rel_\alpha$ n'est pas transitive.

La fonction de propagation prend en entrée un élément du domaine de l'attribut $\alpha$ pour donner en résultat un autre élément de ce domaine.

Pour déterminer la valeur de l'attribut $\alpha$ d'un élément $o_k$ de l'ensemble $EP^o_{\alpha,rel}$, nous devons nous référer à la valeur source de la portée ($v_\alpha$) ainsi qu'à la distance séparant l'élément $o_k$ de l'élément source de la portée selon la relation $rel_\alpha$. Nous notons $distance_{rel_\alpha}$ la fonction qui fournit la distance entre deux éléments selon cette relation ($distance_{rel_\alpha} : OS \times OS \rightarrow \mathbb{N}$). La valeur de l'attribut $\alpha$ assignée à l'élément $o_k$ suit donc la formule suivante :

$$\begin{aligned} o_k.\alpha &= f^n_\alpha(v_\alpha) \text{ avec } distance_{rel_\alpha}(o, o_k) = n \\ &= f_\alpha \circ \ldots \circ f_\alpha(v_\alpha) \end{aligned}$$

Considérons les deux catégories *pred* et *succ* et montrons l'application de la fonction de propagation sur les éléments en fonction de leur distance à $o$ selon $rel_\alpha$.

- $cat = pred \Rightarrow Prev_{rel_\alpha}(o) = P^o_{\alpha,rel}$ : la relation se parcourt en partant de l'élément $o$ et en allant vers les prédécesseurs de $o$ selon $rel_\alpha$. Par définition, et pour toute relation de structure, il existe un unique prédécesseur direct pour chaque élément $o$.

  La fonction $f_\alpha$ s'applique alors de la manière suivante :

  - Soit l'élément structurel $o_1$ prédécesseur direct de $o$, $rel_\alpha(o_1, o)$, alors on a

    $$o_1.\alpha = f_\alpha(v_\alpha)$$

  - Soit l'élément structurel $o_2$ prédécesseur de $o$ et prédécesseur direct de $o_1$, $rel_\alpha(o_2, o_1)$, alors on a

    $$o_2.\alpha = f^2_\alpha(v_\alpha) = f_\alpha(o_1.\alpha)$$

  - Pour tout élément structurel $o_n$ prédécesseur de $o$ et prédécesseur direct de $o_{n-1}$, $rel_\alpha(o_n, o_{n-1})$, alors on a

    $$o_n.\alpha = f^n_\alpha(v_\alpha) = f_\alpha(o_{n-1}.\alpha)$$

- $cat = succ \Rightarrow Next_{rel_\alpha}(o) = P^o_{\alpha,rel}$

  Par rapport au cas précédent, il est important de noter que pour la relation de composition un élément peut admettre plusieurs successeurs directs. Dans ce cas, la fonction $f_\alpha$ s'applique de la manière suivante :

  - Pour tout élément structurel $o_{1i}$ successeur direct de $o$, $rel_\alpha(o, o_{1i})$, alors on a

    $$o_{1i}.\alpha = f_\alpha(v_\alpha)$$

  - Pour tout élément structurel $o_{2i}$ successeur de $o$, et successeur direct de $o_{1i}$, $rel_\alpha(o_{1i}, o_{2i})$, alors on a

    $$o_{2i}.\alpha = f^2_\alpha(v_\alpha) = f_\alpha(o_{1i}.\alpha)$$

  - Pour tout élément structurel $o_n$ successeur de $o$ et successeur direct de $o_{n-1}$,$rel_\alpha(o_{n-1}, o_n)$, alors on a

    $$o_n.\alpha = f^n_\alpha(v_\alpha) = f_\alpha(o_{n-1}.\alpha)$$

Par ailleurs, l'application de cette fonction de propagation engendre une relation, notée $R_\alpha$, sur les valeurs générées. Cette relation est définie sur le domaine de l'attribut $\alpha$ :

$$R_\alpha \subset domain(\alpha) \times domain(\alpha)$$

Cet aspect nous intéresse particulièrement pour les stratégies de recherche que nous allons proposer.

$$v_j = f_\alpha(v_i) \Rightarrow R_\alpha(v_i, v_j)$$

La relation $R_\alpha$ est transitive, c'est-à-dire pour $v_i$, $v_j$ et $v_k$ appartenant au domaine de l'attribut $\alpha$, si $R_\alpha(v_i, v_j)$ et $R_\alpha(v_j, v_k)$ alors $R_\alpha(v_i, v_k)$.

Cette relation admet sa propre sémantique relative au domaine et aux caractéristiques de l'attribut $\alpha$ et elle sera interprétée lors de l'interrogation pour guider la stratégie de recherche.

Les exemples que nous donnons à cet instant restent très génériques. Nous nous appuyons sur des domaines de définition tels que les entiers ($\mathbb{N}$), des ensembles de mots-clés (langage $\mathfrak{L}$) ou pour l'exemple qui suit sur la notion d'implication du modèle logique de recherche d'informations. Nous notons ici $\rightarrow_\alpha$ l'implication logique pour un attribut $\alpha$.

La relation $R_\alpha$ entre les valeurs $v_i$ et $v_j$ exprime une implication. Dans le modèle logique de recherche d'informations, l'implication notée $d \rightarrow q$ signifie que le document représenté par $d$ répond à la requête représentée par $q$. Autrement dit, $d$ reprend les informations contenues dans $q$ et peut en contenir d'autres.

$$R_\alpha(v_i, v_j) \Leftrightarrow \begin{cases} v_i \rightarrow_\alpha v_j \\ ou \\ v_j \rightarrow_\alpha v_i \end{cases}$$

Pour les entiers, nous donnons trois opérateurs de comparaison qui déterminent trois dépendances possibles. Une fonction de propagation correspondant à l'identité fournira la dépendance $v_j = v_i$.

$$domain(\alpha) = \mathbb{N} \text{ et } R_\alpha(v_i, v_j) \Rightarrow \begin{cases} v_j = v_i \\ ou \\ v_j < v_i \\ ou \\ v_j > v_i \end{cases}$$

Sur le langage $\mathfrak{L}$ assimilant le domaine de l'attribut $\alpha$ à un ensemble de mots-clés, nous considérons là encore trois cas possibles en utilisant les opérateurs d'inclusion et l'égalité sur les ensembles. Si la fonction réduit l'ensemble de mots-clés représenté par $v_i$ alors nous aurons : $v_j \subset v_i$.

$$domain(\alpha) = \mathfrak{L} \text{ et } R_\alpha(v_i, v_j) \Rightarrow \begin{cases} v_j \subset v_i \\ ou \\ v_j = v_i \\ ou \\ v_i \subset v_j \end{cases}$$

### 6.3.3 Définition des portées dans un document

Nous avons défini la portée d'un attribut à partir d'un élément en liant celle-ci aux relations de structure du document. Nous nous intéressons maintenant à l'ensemble des portées de ce même attribut que nous pouvons rencontrer au sein du document structuré. Dans un même document, les portées d'un attribut utilisent les mêmes informations, c'est-à-dire un même septuplet à l'élément $o$ près. Afin de déterminer la couverture complète du document par l'attribut $\alpha$, nous allons dorénavant nous intéresser à la composition de ces portées au sein d'un document.

Avant toute chose, il faut noter qu'aucune contrainte particulière n'a pour l'instant été imposée à la définition des portées d'un attribut mis à part les conditions d'appartenance. Nous considérons donc un *état initial* du document structuré dans lequel les portées sont définies selon les informations données par un ensemble de septuplet $\mathcal{P}^{o}_{\alpha,rel}$. Pour chaque élément $o$, un ensemble noté $EP^{o}_{\alpha,rel}$, indique les éléments structurels qui appartiennent à cette portée et qui reçoivent donc de l'information provenant de l'élément $o$.

Cependant, le fait de n'avoir imposé aucune contrainte signifie qu'un élément structurel peut appartenir à deux portées, $EP^{o_i}_{\alpha,rel}$ et $EP^{o_j}_{\alpha,rel}$, de deux éléments distincts, $o_i$ et $o_j$. Cela signifierait que cet élément reçoit de l'information de l'élément $o_i$ et aussi de l'élément $o_j$. Or, nous ne pouvons considérer plus d'un attribut $\alpha$ sur un même élément, nous devons donc instaurer une fonction de combinaison qui va permettre de combiner les informations provenant d'éléments distincts. Cette fonction est notée $combine_\alpha$.

Nous définissons donc le triplet suivant pour déterminer la couverture et la dépendance au sein d'un document structuré à l'aide des portées :

$$(OS_\alpha, \mathcal{P}_{\alpha,rel}, combine_\alpha) \tag{6.2}$$

**L'ensemble $\mathbf{OS}_\alpha$** : l'ensemble des éléments structurels du document qui admettent une portée pour l'attribut $\alpha$.

$$OS_\alpha \subseteq OS$$

**L'ensemble des définitions de portées $\mathcal{P}_{\alpha,\mathbf{rel}}$** : ensemble des définitions $\mathcal{P}^{o_i}_{\alpha,rel}$ décrivant les ensembles $EP^{o_i}_{\alpha,rel}$ pour tout élément $o_i$ de $OS_\alpha$.

$$\forall o_i \in OS_\alpha,\ \exists\, \mathcal{P}^{o_i}_{\alpha,rel} \text{ tel que } \mathcal{P}^{o_i}_{\alpha,rel} \subset \mathcal{P}_{\alpha,rel}$$

**La fonction de combinaison $\mathbf{combine}_\alpha$** : une fonction de combinaison prend en entrée deux valeurs de l'attribut $\alpha$ et donne en résultat une nouvelle valeur pour cet attribut. Une telle fonction se définit comme suit :

$$combine_\alpha : domain(\alpha) \times domain(\alpha) \rightarrow domain(\alpha)$$

Nous introduisons la relation $C_\alpha$ qui indique les dépendances entre les valeurs des attributs. Cette relation est définie sur le domaine de l'attribut $\alpha$ :

$$C_\alpha \subset domain(\alpha) \times domain(\alpha)$$

Cette relation est transitive, c'est-à-dire que pour $a$, $b$ et $c$ appartenant au domaine de l'attribut $\alpha$ nous avons :

si $C_\alpha(a,b)$ et $C_\alpha(b,c)$ alors $C_\alpha(a,c)$

1. **filtrage** : l'une des deux valeurs données en entrée à la fonction, $b$, est ignorée de telle manière que le résultat, $c$, dépend uniquement de l'autre valeur, $C_\alpha(a,c)$.

$$combine_\alpha(a,b) = c \Rightarrow \begin{cases} C_\alpha(b,c) \\ \textbf{et} \\ \neg C_\alpha(a,c) \end{cases}$$

2. **mixage** : les deux valeurs données en entrée à la fonction, $a$ et $b$, sont combinées de telle manière que le résultat, $c$, peut s'exprimer en fonction de ces deux valeurs, $C_\alpha(a,c)$ et $C_\alpha(b,c)$.

$$combine_\alpha(a,b) = c \Rightarrow \begin{cases} C_\alpha(b,c) \\ \textbf{et} \\ C_\alpha(a,c) \end{cases}$$

Si nous considérons que $v_\alpha$ est une valeur quelconque non nulle de l'attribut $\alpha$ et que $NULL$ est la valeur nulle pour cet attribut $\alpha$, alors l'application de la fonction de combinaison sur ces deux valeurs sera : $combine_\alpha(v_\alpha, NULL) = v_\alpha$. La valeur nulle signifie que l'attribut $\alpha$ n'admet pas de valeur mais que cet attribut existe.

La relation $C_\alpha$ doit être compatible avec la relation engendrée par la fonction de propagation, $R_\alpha$. Cette compatibilité s'exprime par la propriété suivante :

si $R_\alpha(a,b)$ et $C_\alpha(b,c)$ alors $C_\alpha(a,c)$

Prenons quelques exemples pour montrer tout l'intérêt de cette propriété. Considérons les domaines de définition suivant : 1) les entiers, $\mathbb{N}$, avec une fonction de propagation qui est l'identité et une fonction de combinaison qui additionne les valeurs; 2) un ensemble de mots clés avec comme fonction de propagation l'identité et comme fonction de combinaison l'union des valeurs; 3) les graphes conceptuels avec comme fonction de propagation la copie et comme fonction de combinaison la jointure.

1. Pour les entiers, la relation issue de la fonction de propagation notée ici $R_\mathbb{N}$, est alors l'égalité, $=$, et la relation issue de la fonction de combinaison $C_\mathbb{N}$ est inférieur ou égal, $\leq$.

   $a, b, c \in \mathbb{N}$
   $R_\mathbb{N}(a,b) \Leftrightarrow a = b$ et $C_\mathbb{N}(b,c) \Leftrightarrow b \leq c$ implique $C_\mathbb{N}(a,c) \Leftrightarrow a \leq c$

2. Pour l'ensemble de mots clés, la relation issue de la fonction de propagation notée ici $R_E$, est alors l'égalité, $=$, et la relation issue de la fonction de combinaison $C_E$ est l'inclusion, $\subseteq$.

   $a, b, c \in E$
   $R_E(a,b) \Leftrightarrow a = b$ et $C_E(b,c) \Leftrightarrow b \subseteq c$ implique $C_E(a,c) \Leftrightarrow a \subseteq c$

3. Pour les graphes conceptuels, la relation issue de la fonction de propagation notée ici $R_{GC}$, est alors l'égalité, $=$ , et la relation issue de la fonction de combinaison $C_{GC}$ est l'implication, $\rightarrow$. La description de ces opérateurs peut être trouvée dans [OAP97].

$$a, b, c \in GC$$
$$R_{GC}(a,b) \Leftrightarrow a = b \text{ et } C_GC(b,c) \Leftrightarrow b \rightarrow c \text{ implique } C_{GC}(a,c) \Leftrightarrow a \rightarrow c$$

Nous verrons par la suite toute l'utilité de cette propriété lors de l'interrogation.

Pour un attribut $\alpha$ et une relation $rel$ nous considérons l'ensemble des portées données par $\mathcal{P}_{\alpha,rel}$ ainsi que la fonction de combinaison $combine_\alpha$ afin de résoudre les conflits et de fournir un document structuré pour lequel les contraintes sur les portées seront vérifiées.

*Contraintes sur l'existence des portées*

Pour que les portées d'un attribut puissent prendre tout leur sens, il faut que les éléments structurels du document vérifient certaines contraintes liées à ces portées. Tout d'abord, pour chaque élément structurel, il doit exister une unique valeur de l'attribut $\alpha$, et donc un élément structurel est soit dans l'ensemble $OS_\alpha$, soit dans un ensemble décrivant une portée, $EP^{o_i}_{\alpha,rel}$. Nous donnons les contraintes suivantes à vérifier :

**Contrainte 6.3.3.1** Un élément structurel de $OS_\alpha$ n'appartient pas à un ensemble décrivant une portée, $EP^{o_i}_{\alpha,rel}$.

$$\forall o_i, o_j \in OS_\alpha,\ o_i \notin EP^{o_j}_{\alpha,rel}$$

**Contrainte 6.3.3.2** Un élément structurel n'appartenant pas à $OS_\alpha$ appartient à une et une seule portée, $EP^{o_i}_{\alpha,rel}$.

$$\forall o_i, o_j \in OS_\alpha,\ o_i \neq o_j, \forall o \in EP^{o_i}_{\alpha,rel},\ o \notin EP^{o_j}_{\alpha,rel}$$

L'introduction de ces deux contraintes (6.3.3.1) et (6.3.3.2) permet de vérifier qu'un élément structurel admettra un unique attribut $\alpha$. Cependant, la vérification de ces contraintes nécessite une phase de résolution de conflits des portées afin d'avoir des portées pour un attribut $\alpha$ qui n'admettent pas d'éléments communs. Nous voyons plus en détail cette phase de résolution de conflits dans les sections 6.8, 6.9 et 6.10.

Avant de préciser la résolution des conflits en fonction des relations structurelles, ce qui représente une part importante de l'application des portées, nous introduisons une nouvelle contrainte afin de réduire la complexité de ces conflits.

**Contrainte 6.3.3.3** Pour un type de document structuré donné, les portées définies sur l'attribut $\alpha$ ne peuvent s'établir à la fois selon la relation séquence et selon la relation de composition.

$$(OS_\alpha, \mathcal{P}_{\alpha,rel}, combine_\alpha) \Rightarrow \begin{cases} rel = comp \\ \textbf{ou} \\ rel = seq \end{cases}$$

Cette contrainte supprime toute possibilité de définition d'une portée d'un même attribut $\alpha$ selon la relation de séquence et selon la relation de composition au sein d'un même document. Nous considérons que la sémantique de ces deux relations est suffisamment différente pour ne pas autoriser la définition de portées d'un attribut selon les deux relations.

Par contre, nous autorisons la définition simultanée d'une portée selon l'une de ces deux relations et selon une relation de référence. Comme nous allons le voir dans la suite, les portées selon les relations de référence permettent de transférer de l'information du contenu du document vers les attributs qui le décrivent. Il ne s'agit donc pas d'une propagation d'attribut d'un élément vers l'attribut d'un autre élément comme pour les relations de séquence et de composition.

### 6.3.4 Synthèse

A travers les définitions des portées des attributs, nous avons décrit le corps de l'indexation structurelle. En effet, nous avions posé comme objectif pour cette indexation d'augmenter la couverture du document par les attributs et de rendre explicite les dépendances entre les valeurs d'un même attribut au sein du document.

D'une part, les portées des attributs garantissent une meilleure couverture du document par les attributs puisque ceux-ci vont pouvoir se propager en suivant les relations de structure du document. D'autre part, les portées des attributs fournissent aussi des relations entre les valeurs des attributs puisque nous appliquons la fonction de propagation au sein d'une portée et la fonction de combinaison entre les portées d'un même attribut au sein d'un document. Nous rendons donc explicite les dépendances entre les valeurs d'un même attribut à l'intérieur d'une portée mais aussi entre les portées d'un même attribut au sein d'un document.

Le document final résultant du processus d'indexation structurelle contient donc des informations supplémentaires par rapport au document résultant d'un processus d'indexation classique. Les apports s'expriment en terme de couverture du document et en terme de dépendances des valeurs. Une meilleure couverture permet d'atteindre des éléments de structure qui ne seraient pas atteints naturellement avec un document n'incluant pas la portée. L'explicitation des dépendances fournit un guide pour la recherche, c'est-à-dire qu'elle permet de déterminer la stratégie de recherche pour accéder aux éléments de structure qui répondent.

Comme le montre la figure 6.4, le processus d'indexation structurelle repose sur l'intégration des portées des attributs afin de nous rapprocher de nos objectifs d'indexation. Le résultat de ce processus est une représentation du document conforme à un modèle de document étendu.

Il nous reste à déterminer la combinaison des portées au sein du document, la construction du modèle de document étendu ainsi que le déroulement de ce processus. Nous allons voir dans la section qui suit quelques exemples de portées relativement aux trois relations de structure du document structuré. Nous enchaînons ensuite sur une classification des attributs selon leur portée. A partir de cette classification, nous établissons le déroulement du processus d'indexation structurelle ainsi que le passage au modèle de document étendu. Enfin nous décrivons l'application des portées dans un document structuré.

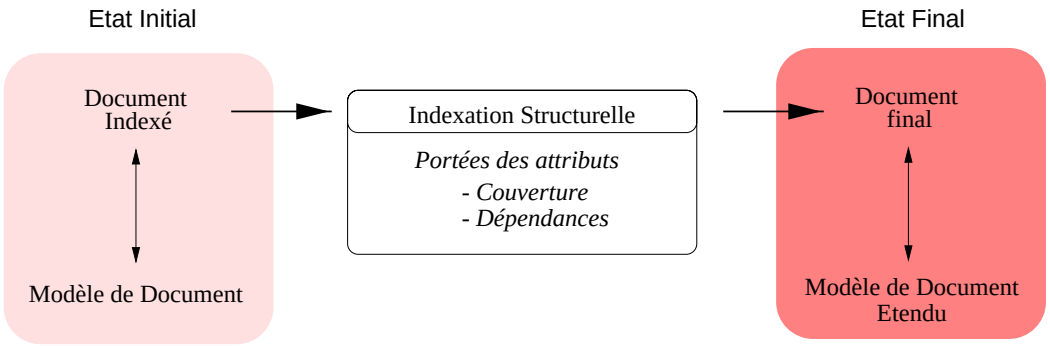



**Figure 6.4.** Processus d'Indexation Structurelle

## 6.4 Usage des relations de structure

Nous avons défini la notion de portée d'un point de vue formel dans un document structuré. Nous allons donner quelques utilisations des relations de structure pour définir des portées dans des documents. Nous proposons tout d'abord des définitions de portée avec les relations de composition et de séquence, celles-ci étant similaires. Puis nous approfondissons la définition des portées avec la relation de référence.

### 6.4.1 Usage des relations de composition et/ou de séquence

La composition, notée $\prec_{comp}$, dans le document structuré exprime l'agrégation d'éléments structurels, c'est-à-dire la participation au sein d'un même élément structurel de plusieurs autres éléments structurels. La séquence, notée $\prec_{seq}$, exprime l'enchaînement des éléments structurels.

Pour définir complètement la portée d'un attribut $\alpha$ attaché à un élément $o$, il nous faut spécifier le quadruplet $(cat, v_\alpha, f_\alpha, cond)$, c'est-à-dire définir une catégorie, un accès à la valeur initiale de $\alpha$, une fonction de propagation et un ensemble de conditions d'appartenance. L'accès à la valeur initiale de $\alpha$ sera donné par la fonction $value_\alpha$ qui fournit la valeur de l'attribut $\alpha$ assignée à l'élément $o$ : $value_\alpha(o)$ [2].

Nous allons considérer deux exemples d'usage de chaque relation de structure. Tout d'abord, l'usage de la relation de composition pour construire une portée admettant l'ensemble des ascendants structurels, puis nous utilisons la relation de séquence pour construire une portée constituée de l'ensemble des successeurs séquentiels.

**Exemple 6.4.1.1 (Ensemble des ascendants structurels)**
*La portée est constituée de* **l'ensemble des ascendants structurels** *de $o$ : l'attribut $\alpha$ attaché à $o$ est assigné à tous les éléments structurels $o_i$ qui sont des ascendants structurels de $o$ avec une valeur provenant de celle de l'élément $o$.*

2. Nous utilisons indifféremment la notation $value_\alpha(o)$ et la notation $o.\alpha$ pour atteindre la valeur d'un attribut $\alpha$ d'un élément structurel $o$

Les informations définissant cette portée comportent la relation de composition, la catégorie *pred* sans imposer de conditions d'appartenance. La portée se définit donc de la manière suivante :

$$\mathcal{P}^{o}_{\alpha,comp} = (\alpha, o, \prec_{comp}, pred, value_{\alpha}(o), f_{\alpha}, vrai)$$

Elle permet de construire l'ensemble[3] $EP^{o}_{\alpha,comp}$ des éléments de structure concernés par l'attribut $\alpha$ de l'élément $o$: $EP^{o}_{\alpha,comp} = Prev_{comp}(o)$

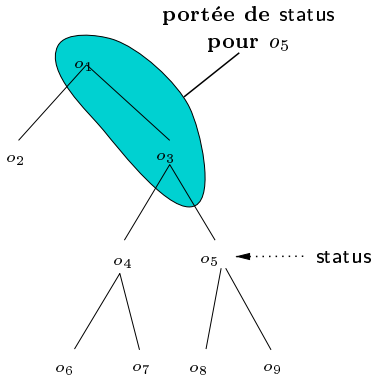



**Figure 6.5.** Portée incluant l'ensemble des ascendants structurels

**Exemple 6.4.1.2 (Ensemble des successeurs séquentiels)**
*La portée est constituée de* **l'ensemble des successeurs séquentiels** *de l'élément $o$. L'attribut $\alpha$ attaché à $o$ est assigné à tous les éléments structurels qui sont des successeurs séquentiels de $o$ avec une valeur provenant de celle attribuée à $\alpha$ de $o$.*

Les informations définissant cette portée comportent la relation de séquence, la catégorie *succ* sans imposer, là encore, de conditions d'appartenance. La portée se définit donc de la manière suivante :

$$\mathcal{P}^{o}_{\alpha,seq} = (\alpha, o, \prec_{seq}, succ, value_{\alpha}(o), f_{\alpha}, vrai)$$

Elle permet de construire l'ensemble $EP^{o}_{\alpha,seq}$ des éléments de structure concernés par l'attribut $\alpha$ de l'élément $o$: $EP^{o}_{\alpha,seq} = Next_{seq}(o)$.

Nous donnons ici un exemple avec un attribut correspondant à la numérotation des paragraphes, notée $nPa$, qui porte donc uniquement sur les éléments structurels de type paragraphe comme le spécifie la condition de la portée ci-dessous : $type_{str}(o_k) =$ paragraphe.

$$\mathcal{P}^{o}_{nPa,seq} = (nPa, o, \prec_{seq}, pred, value_{nPa}(o), f_{nPa}, (type_{str}(o_k) = paragraphe))$$

3. Nous rappelons que les fonctions $Prev_{rel}$ et $Next_{rel}$ donnent respectivement l'ensemble des ascendants structurels et l'ensemble des descendants d'un élément structurel selon la relation $\prec_{rel}$.

Des attributs décrivant la position d'un élément dans la structure peuvent aussi admettre une portée liée à la relation de séquence. Ces attributs expriment par exemple que l'élément $o_i$ est le nième élément. Dans ce cas, l'attribut ne dépend pas explicitement d'un type de structure. Toutefois seule la relation de séquence permet de calculer leur position.

### 6.4.2 Usage de la relation de référence

Les relations de référence dans le document structuré, et particulièrement dans notre modèle expriment les dépendances transversales à l'arborescence structurelle entre les éléments de structure. Nous pouvons différencier deux types de relation : les relations internes et les relations externes. Pour une relation interne, la cible et la source de la relation appartiennent au même document alors que pour une relation externe, elles appartiennent à deux documents différents. Nous ne différencierons pas les traitements des portées définies à partir de relations de références internes et externes car ils sont similaires.

Nous allons surtout nous intéresser à la définition des portées selon les relations de référence et aux choix que nous faisons quant à leur utilisation. Il faut tout d'abord préciser que les relations de référence que nous considérons expriment une dépendance entre deux éléments structurels ou plus exactement une dépendance entre le contenu de l'élément source ($o$) et l'élément cible ($o_j$). En effet, les informations contenues dans l'élément source précisent le contenu de l'élément cible et peuvent donc être représentées par un attribut assigné à cet élément cible. Cette assignation rend explicite, dans la représentation de l'élément structurel cible, l'information formulée au niveau de l'élément source.

Considérons l'exemple suivant dans lequel la relation de référence transmet un attribut *auteur* :

**Exemple 6.4.2.1 (Relation de Référence)**
Echenoz est l'auteur du chapitre 5.

Dans cet exemple que nous schématisons dans la figure 6.6, nous identifions dans le contenu de l'élément source, $o$, une relation de référence qui concerne un attribut auteur et qui a pour cible l'élément de structure correspondant au chapitre 5, $o_k$. La valeur de l'attribut auteur, "Echenoz", est elle aussi dans le contenu de l'élément source.

Chacune de ces informations est importante. Nous notons tout d'abord la relation liant les deux éléments : $o \prec_{ref} o_k$. Toutefois cette relation admet la caractéristique particulière de transmettre un attribut auteur. Nous allons donc dorénavant noter cette relation en indiquant cette caractéristique : $o \prec_{ref}^{auteur} o_k$; les deux éléments sont liés par une relation qui concerne les auteurs de l'élément cible.

Pour résumer, nous avons donc trois éléments essentiels pour définir une portée selon la relation de référence :

**L'élément structurel source $o_s$** : il s'agit d'un élément de structure défini au sein d'un document structuré $D$. L'élément de structure $o_s$ est le point de départ, la source, de la relation de référence.

**L'élément structurel cible $o_c$** : il s'agit d'un élément de structure défini au sein d'un document structuré $D'$. La relation de référence qui part de l'élément source $o_s$ atteint

l'élément cible $o_c$.

S'il s'agit d'une *relation interne*, le document structuré $D'$ est le même que $D$. Par contre, s'il s'agit d'une *relation externe*, les documents $D$ et $D'$ sont différents.

**La relation de référence $\prec^{\alpha}_{\mathbf{ref}}$** : il s'agit de la relation liant l'élément source $o_s$ à l'élément cible $o_c$. Cette relation est orientée et se définit de la manière suivante:

$$\prec^{\alpha}_{ref} \subset OS_D \times OS_{D'}$$

Les ensembles $OS_D$ et $OS_{D'}$ sont respectivement les ensembles des éléments structurels des document $D$ et $D'$. Cette relation de référence lie deux éléments structurels d'un même document structuré (relation interne au document) ou appartenant à deux documents d'une base de documents structurés (relation externe au document). En introduisant une portée définie par des relations de référence, nous interprétons cette liaison comme un transfert d'information du contenu de l'élément source $o_s$ vers un attribut $\alpha$ assigné à l'élément cible $o_c$.

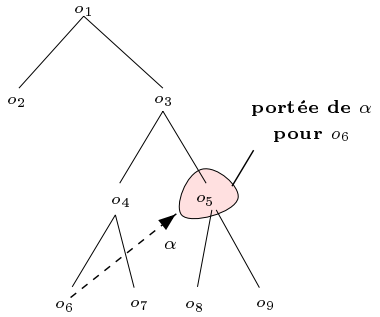



**Figure 6.6.** Portée définie à partir d'une relation interne

L'exemple de relation de référence ci-dessous permet de décrire une portée et donc un ensemble $EP^{o}_{auteur,ref}$.

$$EP^{o}_{auteur,ref} = \{o_k \mid o \prec^{auteur}_{ref} o_k\}$$

Il nous reste à déterminer la valeur de l'attribut transmis. Cette valeur doit être extraite du contenu de l'élément source comme sont extraits le nom de l'attribut et l'élément cible de la relation. L'accès à la valeur de l'attribut se fait par l'intermédiaire de la fonction $extract_{\alpha}$.

$$extract_{\alpha} : OS \rightarrow domain(\alpha)$$

La valeur extraite par l'intermédiaire de la fonction $extract_{\alpha}$ appartient au domaine de l'attribut $\alpha$ ($extract_{\alpha}(o) \in domain(\alpha)$), et permet donc de fournir une valeur pour l'attribut $\alpha$ de l'élément cible. L'application de la fonction $extract_{\alpha}$ sur l'exemple 6.4.2.1 fournit l'information suivante:

$$extract_{auteur}(o) = \{\text{“Echenoz”}\} \text{ avec } \{\text{“}Echenoz\text{”}\} \in domain(auteur)$$

Nous reprenons donc les trois informations suivantes afin d'expliciter les portées pour les relations de référence :

**Le nom de l'attribut** $\alpha$ $(\alpha \in NAME)$ : cet attribut correspond au type de l'information qui est transmise via la relation de référence. La portée selon la relation de référence sera définie sur cet attribut $\alpha$.

**La valeur de l'attribut** $\mathbf{v}_\alpha$ $(v_\alpha \in domain(\alpha))$ : cette valeur est extraite du contenu et doit correspondre au domaine de définition de l'attribut $\alpha$. Il s'agit de l'information transmise par l'intermédiaire de la relation de référence.

$$extract_\alpha(o_s) = v_\alpha$$

**La fonction de propagation** : nous nous limitons à une fonction de propagation correspondant à l'identité et que nous notons $Id_\alpha$. Ainsi la valeur envoyée à l'élément cible sera :

$$Id_\alpha(extract_\alpha(o_s)) = extract_\alpha(o_s)$$

La valeur de l'attribut extraite du contenu de l'élément source correspond à la valeur transmise à l'élément cible, la fonction de propagation est donc une fonction identité.

Nous pouvons maintenant définir complètement une portée selon une relation de référence en utilisant les informations décrites ici : une relation de référence transmettant un attribut $\alpha$, une fonction d'accès à la valeur de l'attribut $\alpha$ notée $extract_\alpha$, une fonction de propagation correspondant à l'identité notée $Id_\alpha$ et une condition d'appartenance explicitant que seul l'élément cible de la relation est pris en compte.

$$\mathcal{P}^{o_s}_{\alpha,ref} = (\alpha, o_s, \prec^\alpha_{ref}, succ, extract_\alpha(o), Id_\alpha, distance(o, o_k) < 2)$$

Par rapport aux portées que nous avons étudiées précédemment, celles-ci se différencient principalement par le fait que l'attribut n'est pas défini sur l'élément source mais qu'il est défini par la relation. De plus, la portée se limite à un ensemble comprenant un unique élément.

Ce type de portée est intéressant afin d'établir des flux d'information entre des éléments structurels de nature différente. Par exemple, dans des parties textuelles d'un document, il n'est pas rare de trouver des références vers une image avec une description de cette image. Les portées sur les relations de référence nous permettent de transmettre ce type d'information entre les médias et donc d'enrichir l'indexation des médias.

## 6.5 Synthèse et Classification des portées

Dans notre étude, nous avons considéré les trois types de relations qui sont à la base de notre modèle de représentation du document structuré et nous avons montré que chacun de ces types de relations peut être utilisé pour définir une portée. Chaque type de relation structurelle

engendre un type de portée particulier admettant sa propre sémantique et répercutant pour chaque attribut le sens de la relation de structure au niveau des attributs.

Nous établissons une classification des attributs qui dépend de leur type de portée :

**Portée vide** : ce type de portée n'est lié à aucune relation particulière puisque l'attribut est dit *statique* et que l'information contenue ne concerne aucun des éléments structurels voisins de $o$.

**Attribut Statique** (AS) : l'attribut ne se propage pas.

Les portées au sein du document se définissent par le triplet suivant pour lequel aucun fonction de combinaison n'est spécifiée puisqu'elle n'est pas utile[4].

$$(OS_\alpha, \mathcal{P}_{\alpha,*}, *)$$

Chaque élément de $\mathcal{P}_{\alpha,comp}$ se définit alors de la manière suivante pour un élément $o$ de $OS_\alpha$. Dans cette définition, nous ne spécifions pas de relation de structure particulière, ni de fonction d'accés, ni de fonction de propagation, ni de condition d'appartenance.

$$\mathcal{P}^o_{\alpha,*} = (\alpha, o, *, \mathbf{static}, *, *, *)$$

**Portée liée à la relation de composition structurelle** : ce type de portée, soutenu par la relation de composition, met en évidence l'aggrégation des informations dans le document structuré au niveau des attributs. Ce type de portée explicite aussi la dissémination des informations dans les différentes parties du document structuré.

De manière simplificatrice, les informations s'aggrègent en remontant vers la racine du document et elles se disséminent en descendant vers les feuilles. Ce type de portée reflète l'organisation des informations au sein d'une arborescence et elle est comparable à tout outil élaboré en plusieurs composants. Chaque composant indépendamment des autres comporte des fonctionnalités de base qui lui sont propres. L'assemblage de l'ensemble des composants lie ces fonctionnalités de base et offre alors des fonctionnalités globales. Le comportement des informations selon la relation de composition est assez similaire. Chaque élément de structure comporte des informations spécifiques qui aggrégées les unes aux autres forment les informations du document structuré complet.

Les portées d'un attribut $\alpha$ selon la relation de composition, au sein d'un document structuré, se définissent par le triplet suivant :

$$(OS_\alpha, \mathcal{P}_{\alpha,comp}, combine_\alpha)$$

Nous avons identifié deux classes d'attributs basées sur la relation de composition. Cette classification est dirigée par le sens de propagation des informations. Nous donnons pour chaque classe la définition d'une portée de $\mathcal{P}_{\alpha,comp}$ pour un élément structurel $o$ de $OS_\alpha$.

**Attribut de composition Ascendant** (AcA) : l'attribut se propage vers le ou les ascendants structurels.

$$\mathcal{P}^o_{\alpha,comp} = (\alpha, o, \prec_{\mathbf{comp}}, \mathbf{pred}, \mathbf{value}_\alpha(\mathbf{o}), f_\alpha, cond)$$

4. La notation * signifie qu'il s'agit d'une information qui n'a pas besoin d'être précisée puisqu'elle va rester inutilsée

**Attribut de composition Descendant** (AcD) : l'attribut se propage vers le ou les descendants structurels.

$$\mathcal{P}^{o}_{\alpha,comp} = (\alpha, o, \prec_{\mathbf{comp}}, \mathbf{succ}, \mathbf{value}_{\alpha}(\mathbf{o}), f_{\alpha}, cond)$$

**Portée liée à la relation de séquence** : ce type de portée, reposant sur la relation de séquence, exprime les dépendances sémantiques liées à l'ordonnancement séquentiel. Elle permet de retrouver des attributs indiquant le positionnement des éléments de structure au sein du document structuré. Par ailleurs, selon le type de structure des documents, et donc le domaine d'application, ce type de portée peut s'avérer utile pour propager des informations; exemple de la gestion des versions.

Les portées d'un attribut $\alpha$ selon la relation de séquence, au sein d'un document structuré, se définissent par le triplet suivant :

$$(OS_{\alpha}, \mathcal{P}_{\alpha,seq}, combine_{\alpha})$$

La relation de séquence fournit deux classes d'attributs qui dépendent là encore du sens de propagation des informations dans leurs portées. Nous donnons pour chaque classe la définition d'une portée de $\mathcal{P}_{\alpha,seq}$ pour un élément structurel $o$ de $OS_{\alpha}$.

**Attribut séquentiel Arrière** (AsB) : l'attribut se propage vers le ou les prédecesseurs séquentiels.

$$\mathcal{P}^{o}_{\alpha,seq} = (\alpha, o, \prec_{\mathbf{seq}}, \mathbf{pred}, \mathbf{value}_{\alpha}(\mathbf{o}), f_{\alpha}, cond)$$

**Attribut séquentiel Avant** (AsF) : l'attribut se propage vers le ou les successeurs séquentiels.

$$\mathcal{P}^{o}_{\alpha,seq} = (\alpha, o, \prec_{\mathbf{seq}}, \mathbf{succ}, \mathbf{value}_{\alpha}(\mathbf{o}), f_{\alpha}, cond)$$

**Portée liée aux relations de références** : ce type de portée, reposant sur les relations de référence, permet d'incorporer dans la représentation des informations provenant du contenu des documents. L'incorporation de ces informations est reflétée par la création d'attributs et leur valuation basée sur le contenu de l'élément source.

$$(OS_{\alpha}, \mathcal{P}_{\alpha,ref}, combine_{\alpha})$$

Dans l'ensemble $OS_{\alpha}$ se trouvent les éléments sources de la relation de référence qui n'admettent pas obligatoirement une valeur pour l'attribut $\alpha$ considéré.

Nous considérons une unique classe d'attributs pour les relations de référence.

**Attribut de référence** (AR) : la portée de ce type d'attributs correspond à l'élément cible d'une relation de référence $\prec^{\alpha}_{ref}$.
Pour un élément $o$ de $OS_{\alpha}$, chaque portée appartenant à $\mathcal{P}_{\alpha,ref}$ se définit de la manière suivante :

$$\mathcal{P}^{o}_{\alpha,ref} = (\alpha, \mathbf{o}, \prec_{\alpha}, \mathbf{succ}, \mathbf{extract}_{\alpha}(\mathbf{o}), \mathbf{Id}_{\alpha}, \mathbf{distance}(\mathbf{o}, \mathbf{o_k}) < \mathbf{2})$$

L'approche proposée ici pour décrire la portée des attributs au sein du document structuré reste très ouverte [FMB97]. Pour un type de document structuré donné, elle permet de définir une *structure d'indexation* [FM96] propre à chaque attribut. Nous sommes restés ici indépendant d'un domaine d'application mais la notion de structure d'indexation reste une notion incontournable dans la recherche des documents structurés. La structure d'indexation représente un sous-ensemble de la structure complète du document, c'est-à-dire l'ensemble des éléments structurels susceptibles d'être intéressants en tant que réponse à une requête. Ce choix de conserver un type d'élément structurel dans la structure d'indexation peut par exemple dépendre du type de l'élément, de son contenu ou encore de sa taille.

Dans le cadre de l'indexation d'encyclopédies, il est plus que probable que l'utilisateur ne souhaite pas retrouver une encyclopédie complète qui, bien qu'elle contienne de l'information susceptible de répondre à son besoin, est d'une telle taille qu'il devra *fouiller* l'encyclopédie complète et effectuer sa propre recherche. Ce type d'élément doit donc être écarté, comme d'autres, des réponses possibles. Ceci est possible grâce à la structure d'indexation.

Nous avons donné dans l'application PRIME-GC [FM96, MMFB97, BFM$^+$97, BFMM97] une structure d'indexation aux documents médicaux que nous avons indexés. Ainsi la structure complète des documents contient des séries d'images médicales composées d'images qui n'étaient pas toutes pertinentes puisque certaines ne contenaient pas d'informations intéressantes ou n'étaient pas "lisibles". La structure d'indexation contient donc uniquement les images médicales qui laissaient apparaître des informations intéressantes c'est-à-dire des informations qui nous permettent de discriminer le contenu de ces images des autres.

Grâce à ce que nous avons présenté, chaque attribut peut dorénavant admettre sa propre structure d'indexation décrite par l'intermédiaire de la relation $rel_\alpha$ au sein de l'ensemble des éléments structurels d'un document structuré ($OS_\alpha$). Cet ensemble est complété par les éléments des portées de cet attribut. La structure d'indexation d'un attribut correspond à l'ensemble des éléments d'un document qui sont concernés par cet attribut, c'est-à-dire qui admettent une valeur pour cet attribut. C'est aussi la structure d'indexation d'un attribut qui va définir quels éléments sont interreogeables par cet attribut. Si un élément appartient à la structure d'indexation d'un attribut $\alpha$ alors il pourra être retrouvé par une requête contenant un critère sur $\alpha$.

## 6.6 Processus Complet d'indexation structurelle

Le processus complet d'indexation structurelle a deux objectifs prioritaires: donner une représentation du document structuré avec la meilleure couverture possible de ses éléments structurels par les attributs et fournir explicitement les relations entre les valeurs d'attributs dans cette représentation.

Afin d'aboutir à cet état de représentation, nous avons introduit la notion de portée d'un attribut, ce qui nous permet de connaître dorénavant *quels sont les éléments structurels qui sont concernés par une information.*

L'introduction des informations relatives aux portées des attributs, à savoir déterminer dans quelle classe se situe chaque attribut, nous conduit à considérer de nouvelles données pour le processus d'indexation structurelle. Dans un processus d'indexation classique de re-

cherche d'information, les données que nous possédons en entrée sont les documents et le modèle de document. Nous faisons ici abstraction des connaissances relatives au domaine ou à une connaissance particulière [Ber88] qui autorisent une extraction plus fine du contenu des documents. En sortie du processus d'indexation, nous avons une représentation du document se conformant au modèle de document étendu.

Ici nous introduisons en entrée du processus des éléments qui étendent les attributs initialement définis au sein du modèle de document. Cette définition est donnée par l'intermédiaire de deux relations : la relation $\Re_{ST}$ pour les attributs liés à un type structurel et la relation $\Re_M$ pour les attributs liés à un média. Avec l'introduction des portées, nous voulons que les informations se propagent dans le document structuré sans que le modèle de document initial ne nous limite dans ce cheminement. Comme nous le montrons plus précisément sur l'exemple de la section 6.7, pour ne pas limiter la propagation des informations et pour conserver en sortie du processus d'indexation une représentation du document conforme à un modèle, il est impératif d'étendre le modèle de document structuré.

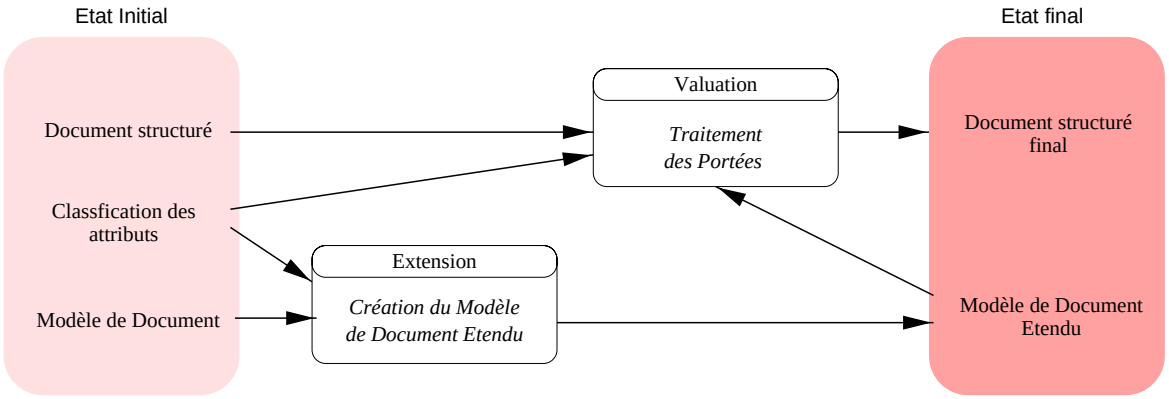



**Figure 6.7.** Processus Complet d'indexation structurelle

La figure 6.7 décrit de manière schématique le processus d'indexation structurelle et l'ordonnancement des composantes de ce processus. Nous distinguons deux composantes : l'*extension* et la *valuation*. L'extension correspond à la création du modèle de document étendu alors que la valuation est l'application sur les documents structurés des portées des attributs, c'est-à-dire la propagation des informations.

Il est important de noter que la figure 6.7 comporte un axe temporel qui indique l'enchaînement des deux composantes. Cet axe indique qu'il faut tout d'abord étendre le modèle de document en utilisant la classification des attributs et le modèle de document initial. Ensuite, nous nous intéressons aux instances, c'est-à-dire aux documents structurés, en propageant les informations et en valuant les nouveaux attributs. Cette phase de valuation considère en entrée les documents structurés, la classification des attributs et le modèle de document étendu afin d'aboutir à une représentation des documents structurés conforme au modèle étendu et remplissant au mieux les objectifs fixés.

## 6.7 Modèle de document étendu

Dans son état initial, le document structuré est décrit par des attributs rattachés à des types d'éléments structurels (par la relation $\Re_{ST}$) ou à des médias (par la relation $\Re_M$) conformément aux spécifications du modèle de document. Lorsque nous augmentons la couverture en introduisant les portées, nous allons attacher des attributs à des éléments structurels d'un type structurel $t$ ou d'un média $m$ n'admettant pas ces attributs dans le modèle de document initial. Ceci signifie que l'introduction et l'application des portées des attributs ne conservent pas la cohérence avec le modèle de document initial.

De manière à conserver un cadre de représentation et une connaissance sur la description finale des documents, nous proposons d'étendre le modèle de document initial. Cette extension a pour objectif de fournir une structure d'accueil au sein du document pour les attributs propagés via les portées. La structure d'accueil correspond à l'autorisation d'attacher des attributs au travers de l'extension des relations $\Re_{ST}$ et $\Re_M$. En effet, dans le modèle initial, ces deux relations sont trop restrictives pour autoriser la propagation des attributs et des valeurs d'attributs. En étendant les deux relations en fonction de la définition des portées (relation de structure, catégorie et conditions d'appartenance), la propagation devient possible et le résultat de l'application des portées sera cohérent avec le *modèle de document étendu*. Dans ce modèle, les relations $\Re'_{ST}$ et $\Re'_M$ remplacent les relations $\Re_{ST}$ et $\Re_M$.

Nous indiquons tout d'abord au travers d'un exemple le besoin de cette extension du modèle. Nous considérons alors d'une part les classes d'attributs qui nécessitent une extension du modèle et celles pour lesquelles cela n'est pas nécessaire.

### 6.7.1 Un exemple d'extension

Afin de mieux comprendre cette logique d'extension du modèle de document, nous considérons l'exemple des documents de type livre que nous avons décrits dans la section 6.1.

Nous considérons l'attribut status comme un attribut de composition descendant (AcD), c'est-à-dire un attribut se propageant selon la relation de composition d'un élément vers ses successeurs. Cet attribut est uniquement défini dans le modèle de document initial sur les éléments de type structurel chapitre.

$$\Re_{ST}(Chapitre, Status)$$
$$et$$
$$\mathcal{P}^o_{status,comp} = (status, o, \prec_{comp}, succ, value_{status}(o), Id_{status}, vrai)$$

D'après la définition de la portée de cet attribut, les éléments structurels qui sont des descendants d'un élément structurel de type chapitre peuvent admettre un attribut status. Les types structurels des éléments qui peuvent être des descendants d'un élément de type chapitre sont donnés par la relation sur les types structurels : $\preceq_{tcomp}$. Cette relation ne donne que les descendants directs et il nous faut l'ensemble des descendants du type chapitre. Il nous faut donc considérer la fonction qui fournit cet ensemble : $Next_{tcomp}$. En effet, pour le type

chapitre :

$$Chapitre \preceq_{tcomp} titreChapitre,\ Chapitre \preceq_{tcomp} Section$$
$$Section \preceq_{tcomp} titreSection,\ Section \preceq_{tcomp} Paragraphe$$

d'où $Next_{tcomp}(Chapitre) = \{titreChapitre, Section, titreSection, Paragraphe\}$

La portée définie pour l'attribut status ne comporte pas de conditions restreignant l'appartenance d'un élément à l'ensemble décrivant la portée, nous pouvons étendre la relation $\mathfrak{R}_{ST}$ de manière à ce que tout les éléments de $Next_{tcomp}(Chapitre)$ admettent cet attribut dans le modèle de document étendu :

$$\mathfrak{R}_{ST}(Chapitre, Status) \text{ et } \mathcal{P}_{Status,comp} \Rightarrow \begin{cases} \mathfrak{R}'_{ST}(TitreChapitre, Status) \\ \mathfrak{R}'_{ST}(Chapitre, Status) \\ \mathfrak{R}'_{ST}(Section, Status) \\ \mathfrak{R}'_{ST}(Paragraphe, Status) \end{cases}$$

De façon générale, l'extension du modèle de document pour l'intégration des portées des attributs nécessite la prise en compte de plusieurs informations :

- Tout d'abord la relation de structure qui est concernée par la portée de l'attribut afin de déterminer la relation sur les types structurels que nous devons considérer.
- Ensuite, il faut considérer la catégorie indiquant le sens de parcours de la relation de structure qui donne là aussi le sens de parcours de la relation sur les types structurels.
- Enfin, nous devons considérer les conditions d'appartenance afin de délimiter l'ensemble des types qui peuvent admettre, via les portées, l'attribut.

Nous sommes ici dans le cas d'un attribut structurel, c'est-à-dire un attribut défini pour un ou plusieurs types d'éléments de structure, et qui nécessite donc l'extension de $\mathfrak{R}_{ST}$. Dans le cas d'attributs de contenu, c'est-à-dire des attributs définis par rapport à un média, c'est la relation $\mathfrak{R}_M$ qui est concernée.

Dans la suite nous donnons les éléments qui permettent d'étendre le modèle en séparant clairement deux types de portée : celles auxquelles nous accordons une extension et celles qui ne nécessitent pas d'extension. Nous nous intéressons donc tout d'abord aux portées définies selon les relations de composition et de séquence pour lesquelles nous étendons le modèle de document. Ensuite nous discutons des portées définies selon la relation de référence et du choix d'étendre ou de ne pas étendre pour ce type de portée.

### 6.7.2 Les attributs nécessitant une extension

Nous considérons ici les extensions du modèle de document pour les portées définies selon les relations de composition et de séquence. La première étape consiste à déterminer quelles sont les informations que nous possédons et dans quel but nous allons les utiliser. Pour cela, nous considérons séparément les attributs structurels et les attributs de contenu puisque les premiers jouent sur les types structurels alors que les seconds jouent sur les médias.

### a) Les attributs de structure

Dans les spécifications des portées des attributs structurels, nous sommes particulièrement intéressés par la relation de structure, la catégorie et les conditions d'appartenance.

- la relation de structure de la portée détermine la relation sur les types à utiliser : soit $\preceq_{tseq}$ pour la relation de séquence, soit $\preceq_{tcomp}$ pour la relation de composition.
- la catégorie de la portée détermine dans quel sens la relation sur les types doit être suivie. Si $cat = pred$ nous considérons une fonction accédant aux types prédécesseurs, et si $cat = succ$ nous considérons une fonction accédant aux types successeurs.
- les conditions d'appartenance délimitent les types qui nous intéressent. Nous ne pouvons ici considérer que des conditions indépendantes du domaine (par exemple sur les types de structure ou la distance) puisque nous ne tenons pas compte des instances.

La première propriété à vérifier pour la construction de la relation $\mathfrak{R}'_{ST}$ concerne l'inclusion de $\mathfrak{R}_{ST}$ : $\mathfrak{R}_{ST} \subseteq \mathfrak{R}'_{ST}$. Ceci signifie que toute relation de $\mathfrak{R}_{ST}$ se retrouve dans $\mathfrak{R}'_{ST}$ ($\mathfrak{R}_{ST} \subset \mathfrak{R}'_{ST}$) :

$\forall t \in TYPE_{ST},\ \forall \alpha \in NAME$ Si $\mathfrak{R}_{ST}(t, \alpha)$ Alors $\mathfrak{R}'_{ST}(t, \alpha)$

Par la suite les règles de construction de la relation $\mathfrak{R}'_{ST}$ reposent sur les éléments suivants :

Soit $\mathcal{P}^{o}_{\alpha,rel} = (\alpha, o, rel, cat, value_\alpha, f_\alpha, cond)$
et $\forall t \in TYPE_{ST},\ \mathfrak{R}_{ST}(t, \alpha)$
Selon $(rel, cat)$ :
  $(rel, cat) = (\prec_{comp}, succ)$ :
    $\forall t' \in Next_{tcomp}(t)$ et $cond_t(t') = vrai \Rightarrow \mathfrak{R}'_{ST}(t', \alpha)$
  $(rel, cat) = (\prec_{seq}, succ)$ :
    $\forall t' \in Next_{tseq}(t)$ et $cond_t(t') = vrai \Rightarrow \mathfrak{R}'_{ST}(t', \alpha)$
  $(rel, cat) = (\prec_{comp}, pred)$ :
    $\forall t' \in Prev_{tcomp}(t)$ et $cond_t(t') = vrai \Rightarrow \mathfrak{R}'_{ST}(t', \alpha)$
  $(rel, cat) = (\prec_{seq}, pred)$ :
    $\forall t' \in Prev_{tseq}(t)$ et $cond_t(t') = vrai \Rightarrow \mathfrak{R}'_{ST}(t', \alpha)$

La fonction $cond_t$ que nous avons introduite nous permet de gérer les conditions d'appartenance indépendantes du domaine, c'est-à-dire sur les types structurels et les distances. Cette fonction est une reformulation sur les types de la fonction sur les éléments structurels donnés dans la définition de la portée section 6.3.2, page 147.

- S'il s'agit d'une condition sur les types, par exemple $type_{str}(o_k) \neq t_1$, nous avons :

  $cond_t(t') = vrai$ ssi $t' \neq t_1$

– S'il s'agit d'une condition sur les distances, $distance_{rel}(o, o_k) < n$, où $rel$ est soit la composition soit la séquence et où l'élément $o$ est de type $t$, $type_{str}(o) = t$, nous avons :

$$cond_t(t') = vrai \text{ ssi } |Prev_{trel}(t') \Leftrightarrow Prev_{trel}(t)| < n$$

où la fonction $Prev_{trel}$ est $Prev_{tcomp}$ si $rel$ correspond à la composition et $Prev_{tseq}$ est $Prev_{tseq}$ si $rel$ correspond à la séquence. Ces fonctions permettent d'accéder respectivement à l'ensemble des types prédécesseurs et successeurs selon la relation de structure.

En adoptant ces choix, c'est-à-dire en restant au niveau des types structurels et en ne considérant pas les instances, nous donnons une relation $\Re'_{ST}$ plus englobante que si nous considérions les instances. Considérons par exemple que le type $t$ puisse être composé d'un ou plusieurs autres éléments de type $t'$ et qu'il admette un attribut de composition descendant $\alpha$ avec une condition limitant la distance à un élément :

$$t \preceq_{tcomp} t' \text{ et } \Re_{ST}(t, \alpha)$$

Soient les éléments $o$, $o_1$ et $o_2$ tels que $type_{str}(o) = type_{str}(o_1) = t$ et $type_{str}(o_2) = t'$. Soit la relation de composition les liant : $o \prec_{comp} o_1$ et $o_1 \prec_{comp} o_2$. L'attribut $\alpha$ est attaché à l'élément $o$.

D'après la définition que nous avons donnée de la portée, seul l'élément $o_1$ va recevoir l'attribut $\alpha$ puisque l'élément $o_2$ est trop éloigné de $o$. Or nos règles de construction vont fournir une relation $\Re'_{ST}(t', \alpha)$ qui ici ne sera pas utilisée. Nous montrons ceci dans la figure 6.8 où la flèche en pointillé entre $\alpha$ et l'élément $o$ correspond à la relation originale $\Re_{ST}$ alors que les flèches pleines entre $\alpha$ et les éléments $o_1$ et $o_2$ proviennent de la relation $\Re'_{ST}$.

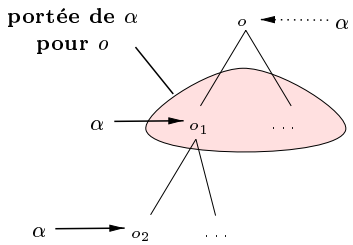



**Figure 6.8.** La relation étendue des attributs structurels fournit un sur-ensemble

Nos règles de construction de la relation $\Re'_{ST}$ fournissent un sur-ensemble des contraintes que doivent respecter le document au niveau des instances.

### b) Les attributs de contenu

La relation $\Re_M$ définissant les attributs de contenu s'exprime en fonction des médias. Nous imposons que chacun des attributs de contenu soit un attribut de composition ascendant afin

de n'avoir à indexer, et donc à valuer, que les éléments monomédias qui sont les feuilles de la structure. Le contenu de ces attributs se propage par la suite vers la racine du document (comportement d'un attribut de composition ascendant).

Par ailleurs, nous avons une contrainte sur les compositions de média au niveau des types structurels qui dit que si un type structurel $t$ est un super-type de $t'$ alors il reprend les médias de $t'$ :

$$\forall m \text{ tel que } \mathfrak{T}(t', m) \text{ et } t \preceq_{tcomp} t' \Rightarrow \mathfrak{T}(t, m)$$

Cette propriété ajoutée au fait que ce sont les éléments monomédias feuilles de structure qui sont indexés, nous permet de vérifier que l'extension de la relation $\mathfrak{R}_M$ est inutile puisque nous ne rajoutons pas d'attributs pour les médias. Nous ne faisons que propager les valeurs des attributs des éléments monomédias feuilles de structure.

### 6.7.3 Les attributs de référence

Les relations de référence et par extension, les attributs de référence, comportent des informations supplémentaires ou complémentaires par rapport à l'indexation classique. Il s'agit d'ajouter des informations qui apparaissent dans le contenu d'un élément de structure du document et qui réfèrent un autre élément de structure du document. Ce type d'ajout a montré son intérêt dans [Par96] pour l'indexation de documents textuels ou encore dans [HSD97] afin d'aider à la recherche d'images fixes dans un environnement hypertextuel.

Nous considérons les relations de référence comme des éléments qui peuvent nous apporter de l'information supplémentaire pour décrire le document et donc pour le retrouver. Cependant, la portée selon la relation de référence pourrait introduire des incohérences. Une relation de référence peut par exemple avoir pour cible un élément structurel de type $t$ et lui transmettre, via sa portée, un attribut de nom $a$ alors que la relation $\mathfrak{R}_{ST}(t, a)$ n'existe pas.

Nous considérons les relations du modèle de document étendu $\mathfrak{R}'_{ST}$ et $\mathfrak{R}'_M$ et vérifions que ce modèle autorise l'application d'une portée utilisant une relation de référence, c'est-à-dire que l'élément structurel cible de la relation peut admettre cet attribut.

Ceci peut se transformer selon les deux règles suivantes :

**Règle 6.7.3.1 (Attribut structurel de référence)**
*L'attribut structurel de nom $a$, ainsi que sa valeur, est transmis vers un élément structurel de type $t$ via une portée définie selon la relation de référence si et seulement si la relation $\mathfrak{R}'_{ST}(t, a)$ existe.*

$$EP^o_{\alpha,ref} = \{o_i\} \textbf{ ssi } \mathfrak{R}'_{ST}(type_{str}(o_i), \alpha)$$

**Règle 6.7.3.2 (Attribut de contenu de référence)**
*L'attribut de contenu de nom $b$, ainsi que sa valeur, est transmis vers un élément structurel de média $m$ via une portée définie selon la relation de référence si et seulement si la relation $\mathfrak{R}'_M(m, b)$ existe.*

$$EP^{o}_{\beta,ref} = \{o_j\} \textbf{ ssi } \mathfrak{R}'_M(media(o_j), \beta)$$

Les portées décrites selon des relations de référence qui ne vérifient pas l'une de ces deux règles, 6.7.3.1 et 6.7.3.2, seront ignorées. Ne pas poser cette contrainte signifie que c'est le contenu du document qui aurait guidé la création des attributs dans le modèle de document étendu. Il nous semble préférable de conserver un modèle de document étendu indépendant du contenu des documents.

## 6.8 Résolution de conflits sur les portées

Le modèle de document étendu intègre la portée des attributs. Nous allons décrire maintenant comment se déroule la phase de résolution de conflits et nous séparons les conflits liés à la relation de référence des conflits liés aux autres relations (relation de séquence et relation de composition).

### 6.8.1 Récapitulatif

La résolution de conflits correspond à la phase de valuation des attributs du document structuré. Nous avons en entrée de cette phase un document structuré et des informations indiquant la classification de chaque attribut de ce document. Cependant les contraintes décrites au niveau du document (page 155) ne sont pas encore vérifiées. Cette phase va consister à rétablir systématiquement des portées vérifiant ces contraintes.

Dans cette phase de résolution de conflits, nous distinguons sur la figure 6.9 les trois types de portées liées aux trois relations structures du document. Cette figure schématise l'ordonnancement des traitements.

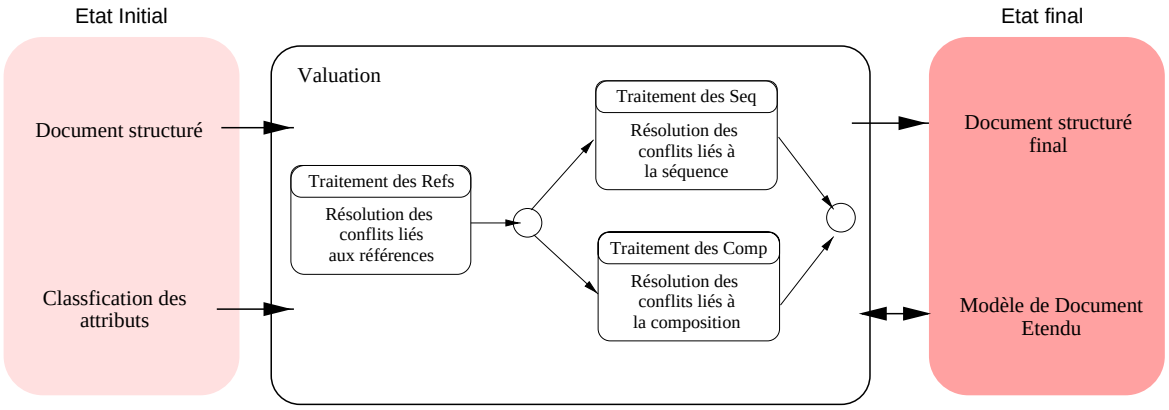



**Figure 6.9.** Le traitement des portées

En premier lieu, les portées définies selon une relation de référence sont traitées. En effet, si par exemple, l'attribut auteur dans un document est un attribut de composition ascendant

et qu'une relation de référence génère un attribut auteur sur un élément alors l'attribut généré est un attribut de composition ascendant.

Nous rappelons que la portée d'attribut ne peut être à la fois définie selon la relation de composition et selon la relation de séquence. Ces portées sont donc indépendantes et peuvent donc être traitées dans un ordre indifférent sans altérer le résultat de ce traitement.

Nous avons défini la notion de portée d'un attribut permettant d'expliciter les dépendances entre les valeurs d'attribut du document structuré. Il est essentiel de vérifier que chaque élément appartenant à un ensemble décrivant une portée admette une unique valeur pour l'attribut concerné.

Les deux règles qui suivent sont des rappels des contraintes énoncées page 155 et elles ne concernent que la relation de composition et la relation de séquence.

Nous devons vérifier la contrainte 6.3.3.1 (de la page 155) pour qu'un élément source n'appartienne pas à un ensemble décrivant une portée :

$$\forall o_i, o_j \in OS_\alpha,\ o_i \neq o_j\ :\ o_i \notin EP^{o_j}_{\alpha,rel}$$

Nous devons de plus vérifier la contrainte 6.3.3.2 (de la page 155) pour que les ensembles décrivant les portées soient disjoints deux à deux :

$$\forall o_i, o_j \in OS_\alpha,\ o_i \neq o_j\ :\ EP^{o_i}_{\alpha,rel} \cap EP^{o_j}_{\alpha,rel} = \emptyset$$

Nous allons tout d'abord traiter les conflits liés aux relations de référence puis nous passerons aux conflits liés aux relations de composition et de séquence.

## 6.9 Conflits liés à la relation de référence

Les conflits liés à la relation de référence peuvent avoir différentes origines donc plusieurs sortes de résolutions.

- *L'attribut transmis sur le type de l'élément cible de la relation de référence dans le modèle de document étendu n'est pas défini au niveau de l'élément cible.* Nous ne transmettons pas cet attribut et ignorons sa portée car sinon cela signifierait que nous transgressons les règles établies par le modèle de document étendu.
- *L'existence préalable d'une valeur pour l'attribut transmis sur l'élément cible de la relation de référence.* Il nous faut donc combiner la valeur de l'attribut de l'élément cible avec celle transmise (fonction $combine_\alpha$).

Le premier type de conflit conduit à ignorer la portée afin d'assurer la cohérence avec le modèle de document étendu.

Par contre, pour le second type de conflit, nous montrons comment se déroule la transmission de la valeur et la combinaison de celle-ci avec la valeur de l'attribut de l'élément cible de la relation de référence.

Soit l'élément cible, noté $o_c$, l'attribut $\alpha$ et l'élément source de la relation de référence, noté $o_s$ :

$$o_c.\alpha = v_\alpha \text{ et } extract_\alpha(o_s) = v'_\alpha$$

L'application de la fonction de combinaison nous mène aux trois cas suivants, valeur nulle, filtrage et combinaison. Nous utilisons la relation de référence suivante : Echenoz est l'auteur du chapitre 5 (décrite dans la section 6.4.2, page 159). Soit $o_5$ l'élément de structure représentant le chapitre 5 et $o$ l'élément de structure représentant l'élément source.

1. *valeur nulle* : la valeur $v_\alpha$ de l'attribut $\alpha$ de l'élément cible est nulle : $v_\alpha = NULL$. Nous considérons que cette valeur nulle existe pour tout domaine d'attribut et qu'elle correspond à un attribut non valué. L'attribut $\alpha$ de cet élément cible prend donc la valeur transmise par la relation de référence.

   $$o_c.\alpha = combine_\alpha(NULL, v'_\alpha) = v'_\alpha$$

   **Exemple 6.9.0.1**
   *Le chapitre 5 admet un attribut* auteur *qui n'est pas valué (égal à NULL) :* $o_5.auteur = NULL$. *La valeur transmise par la relation de référence est donc reçue pour valuer l'attribut* auteur *:*

   $$o_5.auteur = combine_{auteur}(NULL, "Echenoz") = "Echenoz"$$

   Il s'agit ici de valuer un attribut qui existait dans le modèle de document étendu mais qui n'admettait pas de valeur dans ce document. L'application de cette portée représente ici un ajout d'information matérialisé par la valuation de l'attribut.

2. *filtrage* : la valeur transmise par la relation de référence est ignorée de telle manière que le résultat de la combinaison soit toujours égal à la valeur $v_\alpha$ de l'attribut $\alpha$ de l'élément cible.

   $$o_c.\alpha = combine_\alpha(v_\alpha, v'_\alpha) = v_\alpha$$

   **Exemple 6.9.0.2**
   *Le chapitre 5 admet déjà un attribut* auteur *qui est valué :* $o_5.auteur = "Nothomb"$. *La valeur transmise par la relation de référence n'est pas prise en compte pour ne pas modifier la valeur initiale de l'attribut :*

   $$o_5.auteur = combine_{auteur}("Nothomb", "Echenoz") = "Nothomb"$$

   L'application de la portée ne présente pas d'ajout d'information puisque la valeur transmise est ignorée. Il s'agit à travers ce cas de montrer que nous avons une plus grande confiance dans les valeurs originales des attributs que dans les valeurs transmises par les relations de référence. Par ailleurs, il nous est apparu indispensable d'intégrer ce cas de figure afin de pouvoir gérer la transmission d'information sur des attributs monovalués. Si par exemple l'attribut auteur dans l'exemple précédent avait été une chaîne de caractères plutôt qu'un ensemble de chaînes, nous n'aurions pas pu déterminer une fonction de combinaison ainsi qu'une relation de dépendance cohérente. Avec le filtrage, nous signifions que la valeur transmise est ignorée et nous résolvons ainsi ce problème.

3. *mixage* : les deux valeurs, $v_\alpha$ et $v'_\alpha$ se combinent de telle manière que ces deux valeurs apparaissent dans le résultat de la combinaison.

$$o_c.\alpha = combine_\alpha(v_\alpha, v'_\alpha) = v$$

**Exemple 6.9.0.3**
*Nous considérons pour ce cas, que le domaine de l'attribut* auteur *est un ensemble de termes. Le chapitre 5 admet déjà un attribut* auteur *qui est valué :* $o_5.auteur = \{"Nothomb"\}$. *La valeur transmise par la relation de référence est combinée à la valeur initiale de l'attribut (ici par un opérateur d'union) :*

$$o_5.auteur = combine_{auteur}(\{"Nothomb"\}, \{"Echenoz"\}) = \{"Nothomb", "Echenoz"\}$$

Il s'agit ici de donner une nouvelle valeur à un attribut qui était déjà valué. Cette nouvelle valeur est construite à partir d'un mixage de la précédente avec celle qui a été transmise par la portée.

## 6.10 Conflits liés aux autres relations

Nous utilisons par la suite les notations $rel$ et $\prec_{rel}$ pour dénoter indifféremment les relations de composition et de séquence.

Nous avons vu en section 6.5 que les portées sont regroupées en fonction de la catégorie de la portée : *succ* ou *pred*. Nous décrivons les conflits liés à ces deux relations en séparant ces deux catégories.

Les informations données dans les équations 6.3 décrivent la couverture du document par $\alpha$ ainsi que les fonctions de propagation et de combinaison permettant de définir la dépendance des valeurs :

$$\begin{gathered} \forall o_i \in OS_\alpha \ : \mathcal{P}^{o_i}_{\alpha,rel} = (\alpha, o_i, \prec_{rel}, cat, value_\alpha(o), f_\alpha, cond) \\ et \\ (OS_\alpha, \mathcal{P}_{\alpha,rel}, combine_\alpha) \end{gathered} \tag{6.3}$$

Ces portées sont modifiées afin d'atteindre un état dans lequel les contraintes 6.3.3.1 et 6.3.3.2 décrites en section 6.3.3 (page 153) sont vérifiées. Ces contraintes expriment la disjonction des ensembles décrivant les portées. Lorsqu'une contrainte n'est pas respectée, cela signifie qu'un élément structurel reçoit plusieurs valeurs pour un même attribut, et nous déterminons les ensembles décrivant les portées qui ne les respectent pas.

### 6.10.1 Ensembles caractéristiques

Considérons un attribut $\alpha$ et deux éléments distincts $o_i$ et $o_j$ admettant cet attribut (i.e, $o_i, o_j \in OS_\alpha$) et sur lesquels sont définis des portées selon la relation $rel$ indépendamment de la catégorie. Les informations données en section 6.4 décrivent les ensembles $EP^{o_i}_{\alpha,rel}$ et $EP^{o_j}_{\alpha,rel}$.

$$\text{Soit } o_i, o_j \in OS_\alpha,\ o_i \neq o_j \begin{cases} \mathcal{P}^{o_i}_{\alpha,rel} = (\alpha, o_i, \prec_{rel}, cat, value_\alpha(o_i), f_\alpha, cond_i) \\ et \\ \mathcal{P}^{o_j}_{\alpha,rel} = (\alpha, o_j, \prec_{rel}, cat, value_\alpha(o_j), f_\alpha, cond_j) \end{cases} \tag{6.4}$$

Ces ensembles ne vérifient pas obligatoirement les contraintes énoncées précédemment. Nous définissons trois ensembles intervenant dans le processus de résolution des conflits.

- **Portée commune à $o_i$ et $o_j$ notée $p_{(i,j)}$**

  Ensemble des éléments de structure de $OS$ qui appartiennent à la fois à la portée de l'élément $o_i$ et à celle de l'élément $o_j$.

  $$p_{(i,j)} = EP^{o_i}_{\alpha,rel} \cap EP^{o_j}_{\alpha,rel}$$

- **Successeurs spécifiques à $o_i$, ensemble noté $p_i$**

  Ensemble des éléments de structure de $OS$ qui appartiennent à la portée de l'élément $o_i$ et qui n'appartiennent pas à celle de l'élément $o_j$.

  $$p_i = EP^{o_i}_{\alpha,rel} \Leftrightarrow EP^{o_j}_{\alpha,rel}$$

- **Successeurs spécifiques à $o_j$, ensemble noté $p_j$**

  Ensemble des éléments de structure de $OS$ qui appartiennent à la portée de l'élément $o_j$ et qui n'appartiennent pas à celle de l'élément $o_i$.

  $$p_j = EP^{o_j}_{\alpha,rel} \Leftrightarrow EP^{o_i}_{\alpha,rel}$$

Ces ensembles sont disjoints deux à deux et ils fournissent la même couverture du document que les deux ensembles initiaux qui décrivent les portées $EP^{o_j}_{\alpha,rel}$ et $EP^{o_i}_{\alpha,rel}$.

### 6.10.2 Conflits liés à la catégorie successeur

D'après les informations données dans la formule 6.4 et avec la catégorie *succ*, les ensembles $EP^{o_i}_{\alpha,rel}$ et $EP^{o_j}_{\alpha,rel}$ prennent la forme suivante :

$$EP^{o_i}_{\alpha,rel} = \{o_k \mid o_k \in Next_{rel}(o_i) \wedge cond_i(o_k) = vrai\}$$
$$et$$
$$EP^{o_j}_{\alpha,rel} = \{o_k \mid o_k \in Next_{rel}(o_j) \wedge cond_j(o_k) = vrai\}$$

Deux cas peuvent se produire pour un attribut admettant une portée avec la catégorie *succ* selon la relation *rel*.

**Imbrication des portées** : la portée de l'attribut $\alpha$ de l'élément $o_j$ est un sous-ensemble de la portée de l'élément $o_i$

$$EP^{o_j}_{\alpha,rel} \subset EP^{o_i}_{\alpha,rel} \tag{6.5}$$

**Disjonction des portées** : l'intersection entre les portées des deux éléments $o_i$ et $o_j$ est vide.

$$EP^{o_j}_{\alpha,rel} \cap EP^{o_i}_{\alpha,rel} = \emptyset \tag{6.6}$$

### a) Imbrication de portées

L'élément $o_j$ est un descendant structurel de $o_i$ : $o_j \in Next_{rel}(o_i)$.

Nous considérons deux stratégies et utilisons le document de la figure 6.10 comme exemple. Nous constatons que l'ensemble des *successeurs spécifiques à l'élément $o_j$ est vide* ($p_j = \emptyset$).

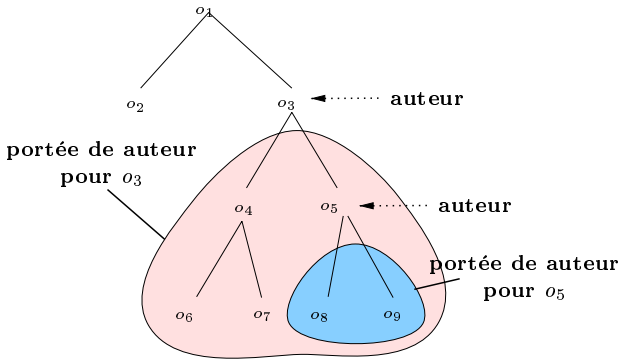



**Figure 6.10.** Combinaison de Portées d'attributs descendants

Nous présentons un cas général et un cas particulier pour résoudre les conflits de cette catégorie.

1. **Cas général**

   Nous redéfinissons les septuplets des portées des éléments $o_i$ et $o_j$ de manière à introduire une condition sur la portée de l'élément $o_i$ afin que $o_j$ et ses successeurs selon la relation $rel$ n'appartiennent plus à la portée de $o_i$. La valeur initiale de l'attribut $\alpha$ de l'élément successeur, $o_j$, est dorénavant $combine_\alpha(v_j, f_\alpha(o_k.\alpha))$ avec $v_j = value_\alpha(o_j)$.

   Les portées de l'attribut $\alpha$ pour $o_i$ et $o_j$ s'expriment alors de la manière suivante :

$$\begin{aligned} &\mathcal{P}^{o_j}_{\alpha,rel} = (\alpha, o_j, \prec_{comp}, succ, combine_\alpha(v_j, f_\alpha(o_k.\alpha)), f_\alpha, cond_j) \\ &et \\ &\mathcal{P}^{o_i}_{\alpha,rel} = (\alpha, o_i, \prec_{comp}, succ, value_\alpha(o_i), f_\alpha, cond'_i) \end{aligned} \tag{6.7}$$

avec $v_j = value_\alpha(o_j)$
et $o_k \in EP^{o_i}_{\alpha,rel}$ et $o_k \prec_{rel} o_j$
tel que $o_k.\alpha = f^m_\alpha(o_i.\alpha)$ avec $distance_\alpha(o_i, o_k) = m$
d'où $R_\alpha(o_i.\alpha, o_k.\alpha)$

et

$$cond'_i = cond_i \wedge o_c \in p_{(i,j)}$$

Les ensembles décrivant les portées sont alors disjoints et correspondent aux ensembles caractéristiques suivants :

$$EP^{o_j}_{\alpha,rel} = p_{(i,j)} \quad et \quad EP^{o_i}_{\alpha,rel} = p_i$$

Deux cas de figure se présentent pour l'application de la fonction de combinaison $combine_\alpha$. Soit cette fonction combine les valeurs, c'est-à-dire qu'elle crée une réelle dépendance entre les valeurs (mixage), soit elle ignore l'une des deux valeurs afin de conserver l'indépendance des valeurs (filtrage).

(a) **Indépendance des valeurs - filtrage**
Si les valeurs sont initialement indépendantes dans les deux portées, elles conservent cette indépendance :

$$o_j.\alpha = combine_\alpha(v_j, f_\alpha(o_k.\alpha)) \Rightarrow \begin{cases} C_\alpha(v_j, o_j.\alpha) \\ et \\ \neg C_\alpha(f_\alpha(o_k.\alpha), o_j.\alpha) \end{cases}$$

La valeur assignée à l'attribut $\alpha$ de $o_j$ par l'intermédiaire de la fonction de combinaison ne dépend pas de la valeur attachée à l'élément racine de la portée de $o_i$. Ceci signifie que la relation $C_\alpha(f_\alpha(o_k.\alpha), o_j.\alpha)$ n'existe pas.
Ce type de combinaison correspond à une *surcharge de la portée de l'élément le plus haut ($o_i$) par la portée de l'élément successeur ($o_j$)*. Dans l'exemple suivant, la valeur provenant de la portée de $o_i$ (par $o_k$) est ignorée par l'élément $o_j$ et par conséquent par tous les éléments de sa portée. La fonction $combine_\alpha$ a la propriété de filtrer la valeur provenant de $o_k$ et de conserver uniquement celle provenant de $o_j$.

**Exemple 6.10.2.1**
*Soit dans la figure 6.10 un attribut* auteur *descendant défini sur $o_3$ et sur $o_5$. Nous considérons des portées définies avec une fonction de propagation $f_{auteur}$ correspondant à l'identité : $f_{auteur}(v) = v_1 \Rightarrow v = v_1$. Au sein d'un ensemble $EP^o_{auteur,comp}$, les attributs* auteur *des éléments structurels ont donc tous la même valeur.*

| $domain(auteur)$ | $R_{auteur}(a,b)$ | $C_{auteur}(a,b)$ |
|---|---|---|
| *Ensemble de chaînes de caractères* | $a = b$ | $a \subseteq b$ |

*Soit les portées définies pour les éléments $o_3$ et $o_5$ :*

$$\mathcal{P}^{o_3}_{auteur,comp} = (auteur, o_3, \prec_{comp}, succ, o_3.auteur, f_{auteur}, cond_3)$$
$$et$$
$$\mathcal{P}^{o_5}_{auteur,comp} = (auteur, o_5, \prec_{comp}, succ, o_5.auteur, f_{auteur}, cond_5)$$

*Les portées initiales sont les suivantes : $EP^{o_3}_{auteur,comp} = \{o_4, o_5, o_6, o_7, o_8, o_9\}$ et $EP^{o_5}_{auteur,comp} = \{o_8, o_9\}$. Les valeurs assignés à l'attribut* auteur *pour $o_3$ et $o_5$ sont :*

$$v_3 = o_3.auteur = \{"Echenoz"\}$$
$$et$$
$$v_5 = o_5.auteur = \{"Nothomb"\}$$

*Ces portées présentent donc un conflit puisque certains éléments sont communs aux deux ensembles $EP^{o_3}_{auteur,comp}$ et $EP^{o_5}_{auteur,comp}$. En appliquant le cas général, l'ensemble décrivant la portée de l'élément $o_3$ est réduite, $EP^{o_3}_{auteur} = p_{(3,5)} = \{o_4, o_6, o_7\}$ alors que la nouvelle valeur assignée à l'élément $o_5$ avec la fonction de combinaison est : $combine_{auteur}(\{"Echenoz"\}, \{"Nothomb"\})$. Nous considérons ici un filtrage ne laissant pas passer les informations provenant de l'élément $o_3$ ; l'auteur est Echenoz. L'élément $o_5$ conserve sa valeur initiale, l'auteur est Nothomb,*

$$o_5.auteur = combine_{auteur}(v_3, v_5) \Rightarrow \begin{cases} o_5.auteur = v_5 \\ et \\ o_3.auteur \not\sqsupseteq o_5.auteur \end{cases}$$

(b) **Dépendance des valeurs - mixage**

La dépendance des valeurs signifie que l'application de la fonction de combinaison introduit une dépendance entre les valeurs qui se propagent au sein des deux portées. Cette dépendance s'exprime à l'aide de la relation $C_\alpha$ :

$$o_j.\alpha = combine_\alpha(v_j, f_\alpha(o_k.\alpha)) \Rightarrow \begin{cases} C_\alpha(v_j, o_j.\alpha) \\ et \\ C_\alpha(f_\alpha(o_k.\alpha), o_j.\alpha) \end{cases}$$

C'est surtout la dépendance explicite entre la nouvelle valeur assignée à l'élément $o_j$ et celle provenant de l'élément $o_i$ qui nous intéresse. En effet, les propriétés et contraintes sur les fonctions de combinaison et de propagation énoncées en section 6.3.2 nous permettent de vérifier la propriété suivante sur les valeurs des attributs $\alpha$ des éléments $o_i$ et $o_j$ de l'ensemble $OS_\alpha$ : $C_\alpha(o_i.\alpha, o_j.\alpha)$.

Nous donnons un exemple dans lequel *la valeur provenant de l'attribut $\alpha$ de l'élément prédécesseur ($o_i$) s'agrège avec la valeur de l'attribut $\alpha$ de l'élément successeur ($o_j$) pour fournir la nouvelle valeur de $\alpha$ pour $o_j$.*

**Exemple 6.10.2.2**

*Considérons le même exemple que précédemment (exemple 6.10.2.1). Les ensembles*

*définissant les portées restent inchangés, mais ici, la fonction de combinaison réalise un mixage des valeurs.*

$$\mathcal{P}^{o_3}_{auteur,comp} = (auteur, o_3, \prec_{comp}, succ, o_3.auteur, f_{auteur}, cond_3)$$
*et*
$$\mathcal{P}^{o_5}_{auteur,comp} = (auteur, o_5, \prec_{comp}, succ, combine_{auteur}(v_3, v_5), f_{auteur}, cond'_5)$$

*Cependant la valeur assignée pour l'élément $o_5$ est modifiée pour prendre en considération la valeur provenant de $o_3$. Cette modification intervient au niveau de la fonction de combinaison $combine_{auteur}$ qui ne filtre plus mais agrège les informations :*

$$\begin{aligned} o_5.auteur &= combine_{auteur}(v_3, v_5) \\ &= v_3 \cup v_5 = \{"Echenoz", "Nothomb"\} \end{aligned}$$

*Sur cet exemple on obtient donc $o_5.auteur = \{"Echenoz", "Nothomb"\}$. Ceci s'interprète comme la combinaison au niveau de $o_5$ des valeurs provenant de $o_3$ et de celle assignée initialement à $o_5$. L'auteur de $o_3$ est Echenoz et par conséquent Echenoz est l'auteur de chacun des composants de $o_3$, donc de $o_5$. Par définition de la portée, tous les descendants structurels de $o_5$ admettront un attribut* auteur *avec une valeur provenant de celle de $o_5$. Ici nous vérifions donc la propriété suivante :*

$$o_5.auteur = combine_{auteur}(v_3, v_5) \Rightarrow \begin{cases} v_5 \subseteq o_5.auteur \\ et \\ o_3.auteur \subseteq o_5.auteur \end{cases}$$

*c'est-à-dire :*

$$o_3.auteur = v_3 = \{"Echenoz"\} \subseteq o_5.auteur = \{"Echenoz", "Nothomb"\}$$

2. **Cas particulier : la portée de l'élément successeur ($o_j$) est absorbée par la portée de l'élément prédécesseur ($o_i$)**

   Pour que ce cas particulier existe il faut que les trois conditions suivantes soient remplies : 1) il existe un élément $o_k$ dans la portée de l'élément $o_i$ tel que 2) l'élément $o_k$ soit le prédécesseur direct de l'élément $o_j$ et 3) l'application de la fonction de propagation $f_\alpha$ sur la valeur de l'attribut $\alpha$ de $o_k$ soit égale à la valeur de l'attribut $\alpha$ de $o_j$.

   **Si** $o_k \in EP^{o_i}_{\alpha,rel}$ et $o_k \prec_{rel} o_j$ et $f_\alpha(o_k.\alpha) = o_j.\alpha$
   **Alors** la portée de $o_j$ est absorbée par celle de $o_i$

Il ne réside donc qu'une unique portée définie sur l'élément $o_i$, celle définie sur l'élément $o_j$ a été absorbée. Dans l'état initial nous avions deux portées distinctes, mais leurs valeurs étant compatibles, nous supprimons la portée de l'élément le plus bas, ici l'élément $o_j$, et nous conservons uniquement la portée de l'élément le plus haut, $o_i$. Nous

réduisons donc l'ensemble des éléments sources en retirant l'élément $o_j$ de l'ensemble $OS_\alpha$.

**Exemple 6.10.2.3**
*Si nous reprenons toujours le même exemple avec la relation de composition, mais en considérant dorénavant que la valeur de l'attribut* auteur *de l'élément $o_5$ est la même que celle de l'élément $o_3$. La fonction de propagation correspond toujours à l'identité. Nous avons alors :*

$$o_5.auteur = \{"Echenoz"\} \text{ et } f_{auteur}(o_3.auteur) = \{"Echenoz"\}$$

*La portée définie sur l'élément $o_5$ peut donc être supprimée, en sachant que la portée de l'élément $o_3$ garantit que nous ne perdons pas d'information.*

### b) Disjonction de portées

Dans ce cas l'intersection entre les ensembles décrivant les portées des deux éléments $o_i$ et $o_j$ est vide. Les trois contraintes que nous avons énoncées sur la validité des portées sont ici toutes les trois vérifiées. Nous rappelons qu'il s'agit de l'équation 6.6 : $EP^{o_i}_{\alpha,rel} \cap EP^{o_j}_{\alpha,rel} = \emptyset$.

Les éléments $o_i$ et $o_j$ ne sont pas en relation dans l'arborescence structurelle, que ce soit par la relation de composition ou par la relation de séquence, c'est-à-dire que $o_i \notin Next_{rel}(o_j)$ et $o_i \notin Prev_{rel}(o_j)$ comme nous le montrons dans la figure 6.11 avec les attributs auteur des éléments $o_4$ et $o_5$ pour la relation de composition.

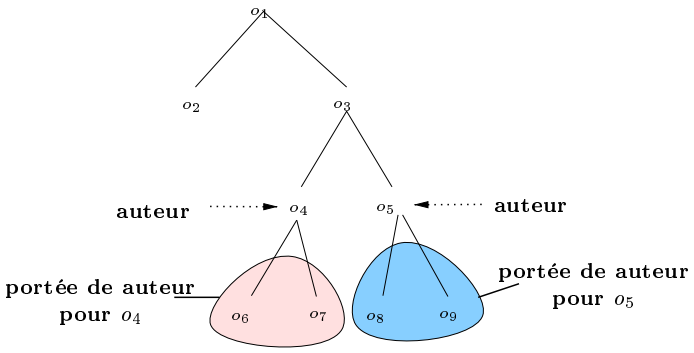



**Figure 6.11.** Portées d'attributs disjointes

Les attributs $\alpha$ de $o_i$ et de $o_j$ demeurent indépendants et les valeurs qui leurs sont assignées pour $\alpha$ sont elles aussi indépendantes. Nous reprenons donc les portées spécifiques pour les ensembles décrivant les portées puisqu'elles sont valides vis-à-vis des contraintes sur les portées.

$$EP^{o_j}_{\alpha,rel} = p_j \quad et \quad EP^{o_i}_{\alpha,rel} = p_i$$

**Exemple 6.10.2.4**
*Dans la figure 6.11, nous donnons un exemple d'une telle configuration pour la relation de composition. Les attributs* auteur *sont définis sur les éléments $o_4$ et $o_5$. Les ensembles définissant la portée de ces attributs sont disjoints car $o_4$ et $o_5$ ne sont pas parents structurels et donc les contraintes sont vérifiées. Les portées peuvent être conservées telles qu'elles étaient dans leur état initial.*

### 6.10.3 Conflits liés à la catégorie prédécesseur

Considérons un attribut $\alpha$ et deux éléments distincts $o_i$ et $o_j$ admettant cet attribut et sur lesquels sont définies des portées avec la catégorie *pred* selon la relation $rel$. Les ensembles $EP^{o_i}_{\alpha,rel}$ et $EP^{o_j}_{\alpha,rel}$ prennent la forme suivante:

$$EP^{o_i}_{\alpha,rel} = \{o_k \mid o_k \in Prev_{rel}(o_i) \wedge cond_i(o_k) = vrai\}$$
$$et$$
$$EP^{o_j}_{\alpha,rel} = \{o_k \mid o_k \in Prev_{rel}(o_j) \wedge cond_j(o_k) = vrai\}$$

Nous allons décrire les différents états que nous pouvons rencontrer pour un attribut admettant une portée avec la catégorie *pred* selon la relation $rel$ comme nous l'avons fait pour la catégorie *succ*. Nous identifions deux cas de figure pour lesquels nous devons résoudre des conflits:

**Inclusion de portées** : la portée de l'attribut $\alpha$ de l'élément $o_j$ est un sous-ensemble de la portée de l'élément $o_i$

$$EP^{o_j}_{\alpha,rel} \subset EP^{o_i}_{\alpha,rel} \tag{6.8}$$

**Non inclusion de portées** : l'intersection entre les portées des deux éléments $o_i$ et $o_j$ n'est pas vide et l'une des deux portées n'est pas incluse dans l'autre.

$$EP^{o_j}_{\alpha,rel} \cap EP^{o_i}_{\alpha,rel} \neq \emptyset \text{ et } EP^{o_i}_{\alpha,rel} \not\subset EP^{o_j}_{\alpha,rel} \text{ et } EP^{o_j}_{\alpha,rel} \not\subset EP^{o_i}_{\alpha,rel} \tag{6.9}$$

#### a) Inclusion de portées

L'ensemble décrivant la portée de l'attribut $\alpha$ de l'élément prédécesseur est un sous-ensemble de l'ensemble décrivant la portée de l'élément successeur. Compte-tenu de la définition de l'attribut $\alpha$, l'élément structurel $o_i$ est un prédécesseur structurel de l'élément $o_j$ : $o_i \in Prev_{rel}(o_j)$. L'ensemble des prédécesseurs spécifiques à l'élément $o_i$, noté $p_i$, est vide. En effet, l'élément $o_i$ est un prédécesseur structurel de l'élément $o_j$, et sachant qu'il existe un unique prédécesseur structurel direct par élément et que $o_i$ et $o_j$ partagent les mêmes prédécesseurs, il ne peut pas exister d'éléments structurels qui soient uniquement des prédécesseurs de $o_i$ sans être des prédécesseurs de $o_j$. L'ensemble $p_i$ est donc vide.

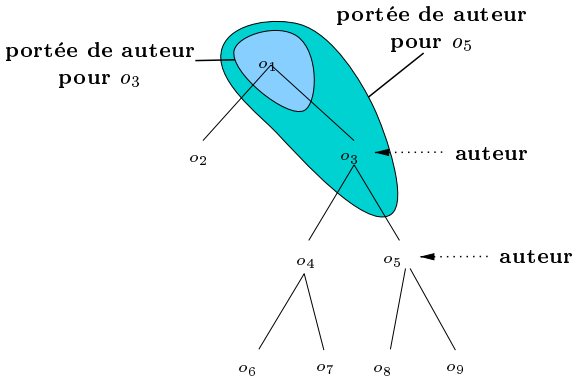



**Figure 6.12.** Combinaison de Portées d'attributs ascendants

Comme pour la catégorie *succ* nous allons considérer deux stratégies élémentaires pour résoudre les conflits et nous utilisons le document de la figure 6.12 comme exemple. Soit un cas général et un cas particulier que nous présentons par la suite.

1. **Cas général**

   Dans le cas général, lorsque la portée d'un élément $o_i$ est incluse dans la portée d'un autre élément $o_j$, la portée de l'élément $o_j$ est réduite en terme d'ensemble alors que celle de l'élément $o_i$ voit sa valeur initiale modifiée: $value_\alpha(o)$ devient $combine_\alpha(v_i, f_\alpha(o_k.\alpha))$. Les portées des deux éléments s'expriment donc après la résolution du conflit de la manière suivante:

$$\begin{aligned} &\mathcal{P}^{o_i}_{\alpha,rel} = (\alpha, o_i, \prec_{rel}, pred, combine_\alpha(v_i, f_\alpha(o_k.\alpha)), f_\alpha, cond_i) \\ &et \\ &\mathcal{P}^{o_j}_{\alpha,rel} = (\alpha, o_j, \prec_{rel}, pred, value_\alpha(o_j), f_\alpha, cond'_j) \end{aligned} \tag{6.10}$$

   où

$$\begin{aligned} &v_i = o_i.\alpha \\ &\text{et } o_k \in EP^{o_j}_{\alpha,rel} \quad \text{et } o_i \prec_{rel} o_k \\ &\qquad\qquad \text{tel que } o_k.\alpha = f^m_\alpha(o_j.\alpha) \text{ avec } distance_\alpha(o_j, o_k) = m \\ &\qquad\qquad \text{d'où } R_\alpha(o_j.\alpha, o_k.\alpha) \end{aligned}$$

   L'élément $o_k$ est l'élément de l'ensemble décrivant la portée de l'élément $o_j$ le plus proche selon la relation $rel$ de l'élément $o_i$ et c'est la valeur de l'attribut $\alpha$ de $o_k$ qui est transmise et combinée avec celle attachée à $o_i$ pour fournir une nouvelle valeur. Et,

$$EP^{o_i}_{\alpha,rel} = \{o_c \mid o_c \in p_{(i,j)} \text{ et } cond(o_c) = vrai\}$$

signifie que tout élément appartenant à l'ensemble décrivant la portée de l'élément $o_i$ appartient à la portée commune notée $p_{(i,j)}$ et vérifie les conditions d'appartenance. Et,

$$EP^{o_j}_{\alpha,rel} = \{o_c \mid o_c \in p_j \text{ et } cond'_j(o_c) = vrai\}$$

signifie que tout élément appartenant à l'ensemble décrivant la portée de l'élément $o_j$ appartient à l'ensemble des prédécesseurs de $o_j$ notée $p_j$.

Nous montrons maintenant comment se déroule la combinaison des valeurs selon que nous appliquions un filtrage (indépendance des valeurs) ou un mixage des valeurs (dépendance des valeurs).

(a) **Indépendance des valeurs - filtrage**

L'indépendance des valeurs signifie que nous ne lions pas les valeurs des portées définies à partir des deux éléments $o_i$ et $o_j$. La fonction de combinaison agit ici comme un filtre en ignorant la valeur de l'élément $o_k$ défini précédemment dans 6.10. Si, dans l'état initial, il n'existe pas de dépendance explicite entre la valeur de l'attribut $\alpha$ de $o_i$ et celle de $o_j$, alors l'indépendance entre les valeurs est conservée et les propriétés suivantes sont vérifiées.

$$o_i.\alpha = combine_\alpha(v_i, f_\alpha(o_k.\alpha)) \Rightarrow \begin{cases} C_\alpha(v_i, o_i.\alpha) \\ et \\ \neg C_\alpha(f_\alpha(o_k.\alpha), o_i.\alpha) \end{cases}$$

**Exemple 6.10.3.1**

*Soit un attribut de composition ascendant* auteur *défini sur les éléments $o_3$ et $o_5$ du document donné dans la figure 6.12. La fonction $f_{auteur}$ correspond à l'identité. Le domaine et les relations sont alors les suivantes :*

| $domain(auteur)$ | $R_{auteur}(a,b)$ | $C_{auteur}(a,b)$ |
|---|---|---|
| *Ensemble de chaînes de caractères* | $a = b$ | $a \subseteq b$ |

*Les valeurs assignés à cet attribut pour les éléments structurels $o_3$ et $o_5$ sont $v_3 = o_3.auteur = \{"Echenoz"\}$ et $v_5 = o_5.auteur = \{"Nothomb"\}$.*

$$\mathcal{P}^{o_3}_{auteur,comp} = (auteur, o_3, \prec_{comp}, pred, combine_{auteur}(v_3, v_5), f_{auteur}, cond_3)$$
*et*
$$\mathcal{P}^{o_5}_{auteur,comp} = (auteur, o_5, \prec_{comp}, pred, v_5, f_{auteur}, cond_5)$$

*Les portées initiales respectives des éléments structurels $o_3$ et $o_5$ sont :*

$$EP^{o_3}_{auteur,comp} = \{o_1\} \quad et \quad EP^{o_5}_{auteur,comp} = \{o_1, o_3\}$$

*Nous nous situons dans le cas général et nous obtenons une nouvelle portée pour $o_5$ correspondant à l'ensemble des prédécesseurs spécifiques à l'élément $o_5$. Cet ensemble que nous avons noté $p_5$ est ici vide.*

*En appliquant la stratégie de résolution avec un filtrage, l'attribut* auteur *de l'élément $o_3$ ne tient pas compte de la valeur de l'attribut* auteur *de l'élément $o_5$,*

*c'est-à-dire que la fonction de combinaison $combine_{auteur}$ filtre les informations provenant de l'élément le plus bas ($o_5$).*

$$o_3.auteur = combine_{auteur}(v_3, v_5) = v_3 = \{"Echenoz"\} \Rightarrow \begin{cases} o_3.auteur = v_3 \\ et \\ o_3.auteur \not\supseteq v_5 \end{cases}$$

*En utilisant cette stratégie, nous n'incorporons pas les informations provenant des éléments les plus bas de la hiérarchie. Ces informations sont masquées par les informations des éléments qui sont plus haut dans la hiérarchie. Nous obtenons finalement les portées suivantes :*

$$\begin{aligned} \mathcal{P}^{o_3}_{auteur,comp} = \; & (auteur, o_3, \prec_{comp}, pred, v_3, f_{auteur}, cond_3) \\ & et \\ \mathcal{P}^{o_5}_{auteur,comp} = \; & (auteur, o_5, \prec_{comp}, pred, v_5, f_{auteur}, cond_5 \wedge o_k \in p_5) \\ & où\ p_5 = \emptyset \end{aligned}$$

Le filtrage conserve l'indépendance des valeurs et la résolution de conflit produit donc uniquement des modifications au niveau des ensembles décrivant les portées.

(b) **Dépendance des valeurs - mixage**

Comme pour la catégorie *succ*, la dépendance des valeurs s'exprime à travers la relation $C_\alpha$ qui est générée entre les données et le résultat de la fonction de combinaison.

$$o_i.\alpha = combine_\alpha(v_i, f_\alpha(o_k.\alpha)) \Rightarrow \begin{cases} C_\alpha(v_i, o_i.\alpha) \\ et \\ C_\alpha(f_\alpha(o_k.\alpha), o_i.\alpha) \end{cases}$$

Là encore, la relation entre la valeur assignée à l'attribut $\alpha$ de l'élément $o_i$ et celle provenant de $\alpha$ de l'élément $o_j$ (par l'intermédiaire de l'élément $o_k$) nous permet de vérifier la propriété suivante :

$$C_\alpha(o_j.\alpha, o_i.\alpha)$$

**Exemple 6.10.3.2**

*En reprenant toujours l'exemple de document de la figure 6.12 et en appliquant cette stratégie à l'attribut* auteur*, la portée de l'élément $o_5$ correspond aux successeurs spécifiques à l'élément $o_5$ : $p_5 = \emptyset$. Par ailleurs, la valeur pour l'attribut* auteur *de $o_3$ intègre la valeur provenant de l'attribut* auteur *de l'élément $o_5$.*

*Si nous considérons que la fonction de combinaison des valeurs est une agrégation alors nous obtenons la valeur suivante pour l'élément $o_3$ et sa portée :*

$$\begin{aligned} o_3.auteur &= combine_{auteur}(v_3, v_5) \\ &= v_3 \cup v_5 = \{"Echenoz", "Nothomb"\} \end{aligned}$$

*Le nom d'auteur "Echenoz" provenant de l'attribut de l'élément $o_5$ est pris en compte par les attributs* auteur *des éléments qui sont des ascendants structurels de $o_5$, via la portée ascendante spécifique à $o_5$, mais aussi pour les ascendants structurels de $o_3$ via la portée ascendante commune. Nous pouvons vérifier que la propriété suivante est vérifiée :*

$$o_3.auteur = combine_{auteur}(v_3, v_5) \Rightarrow \begin{cases} v_5 \subseteq o_3.auteur \\ et \\ o_3.auteur \subseteq o_5.auteur \end{cases}$$

2. **Cas Particulier : la portée de l'élément prédécesseur ($o_i$) est absorbée par la portée de l'élément successeur ($o_j$).**

   Pour qu'une portée décrite sur un élément soit absorbée par une autre portée décrite sur un autre élément, la condition suivante doit être remplie : soit un élément structurel $o_k$ appartenant à l'ensemble décrivant la portée de $\alpha$ à partir de l'élément $o_j$, $o_k \in P^{o_j}_{\alpha,rel}$, cet élément $o_k$ est le successeur structurel selon la relation $rel$ de l'élément $o_i$, $o_i \prec_{rel} o_k$ et l'application de la fonction de propagation $f_\alpha$ sur la valeur de $\alpha$ assignée à $o_k$ est égale à la valeur de $\alpha$ de $o_i$ : $f_\alpha(o_k.\alpha) = o_i.\alpha$.

   **Si** $o_k \in EP^{o_j}_{\alpha,rel}$ et $o_i \prec_{rel} o_k$ et $f_\alpha(o_k.\alpha) = o_i.\alpha$
   **Alors** la portée de $o_i$ est absorbée par celle de $o_j$

   L'absorption d'une portée signifie qu'à partir de deux portées décrites sur deux éléments différents nous construisons une unique portée définie sur un des deux éléments de telle manière qu'il n'y ait pas de perte d'information. Cette reformulation correspond à une simplification des portées.

   Ce cas particulier se produit lorsqu'il existe déjà au sein d'un document structuré une dépendance des valeurs des attributs. Nous réduisons alors les portées afin de simplifier et de ne conserver qu'un minimum de définitions de portées pertinentes.

### b) Non inclusion de portées

L'intersection entre les ensembles décrivant les portées des deux éléments $o_i$ et $o_j$ est non vide et ces ensembles ne sont pas inclus l'un dans l'autre (6.9, page 181).

Ce cas n'existe qu'avec la relation de composition. Les propriétés de la relation de séquence impliquent que si l'intersection entre les deux ensembles est non vide, un ensemble est inclus dans l'autre. Les deux éléments vérifient alors la propriété suivante : $o_j \notin Next_{comp}(o_i)$ et $o_j \notin Prev_{comp}(o_i)$; les éléments structurels $o_i$ et $o_j$ ne sont pas liés par la relation de composition. Ils admettent uniquement des parents communs comme le montre la figure 6.13.

D'un état initial dans lequel nous avions deux portées, celles des éléments $o_i$ et $o_j$, nous aboutissons à un état dans lequel nous avons dorénavant trois portées, celles des éléments $o_i$ et $o_j$ que nous restreignons d'un point de vue ensembliste et celles de l'élément $o_k$ qui est le plus petit élément commun des portées initiales des éléments $o_i$ et $o_j$. L'élément est donc rajouté dans l'ensemble $OS_\alpha$ puisqu'il admet maintenant une portée. Il prend pour valeur

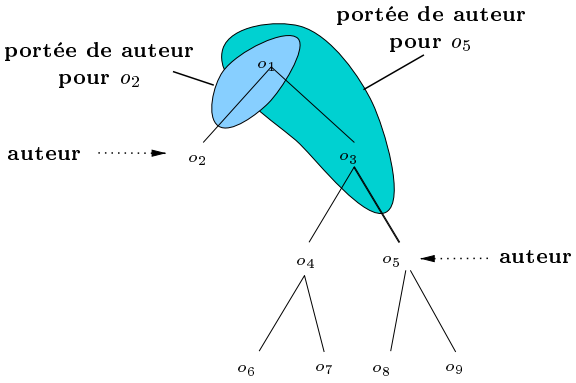



**Figure 6.13.** Intersection non vide sans inclusion entre des portées ascendantes

initiale une combinaison des valeurs provenant des éléments les plus hauts des ensembles $p_j$ et $p_i$ représentant les prédécesseurs spécifiques aux éléments $o_i$ et $o_j$. Nous obtenons donc les trois portées suivantes pour résoudre ce type de conflit :

$$(\alpha, o_j, \prec_{comp}, pred, value_\alpha(o_j), f_\alpha, o_c \in p_j)$$
$$et$$
$$(\alpha, o_i, \prec_{comp}, pred, value_\alpha(o_i), f_\alpha, o_c \in p_i)$$
$$et$$
$$(\alpha, o_k, \prec_{comp}, pred, combine_\alpha(f_\alpha(v_{ir}), f_\alpha(v_{jr})), f_\alpha, o_c \in p_{(i,j)} \wedge o_c \neq o_k)$$

L'élément structurel $o_k$ est ajouté dans l'ensemble des éléments sources de l'attribut $\alpha$ puisqu'une nouvelle valeur de cet attribut est construite.

$$v_{ir} = o_{ir}.\alpha = f_\alpha^n(o_i.\alpha) \text{ avec } o_{ir} \in p_i \text{ et } n = distance_\alpha(o_i, o_{ir})$$
$$ou$$
$$v_{ir} = o_i.\alpha \text{ si } EP^{o_i}_{\alpha,comp} = \emptyset$$

et

$$v_{jr} = o_{jr}.\alpha = f_\alpha^m(o_j.\alpha) \text{ avec } o_{jr} \in p_j \text{ et } m = distance_\alpha(o_j, o_{jr})$$
$$ou$$
$$v_{jr} = o_j.\alpha \text{ si } EP^{o_j}_{\alpha,comp} = \emptyset$$

La propriété suivante est alors vérifiée :

$$o_k.\alpha = combine_\alpha(f_\alpha(v_{ir}), f_\alpha(v_{jr})) \Rightarrow \begin{cases} C_\alpha(f_\alpha(v_{ir}), o_k.\alpha) \\ et \\ C_\alpha(f_\alpha(v_{jr}), o_k.\alpha) \end{cases}$$

De plus, d'après les propriétés de la fonction de propagation, nous avons $R_\alpha(f_\alpha(v_{ir}), o_i.\alpha)$ et $R_\alpha(f_\alpha(v_{jr}), o_j.\alpha)$, nous vérifions donc la propriété suivante :

$$o_k.\alpha = combine_\alpha(f_\alpha(v_{ir}), f_\alpha(v_{jr})) \Rightarrow \begin{cases} C_\alpha(o_i.\alpha, o_k.\alpha) \\ et \\ C_\alpha(o_j.\alpha, o_k.\alpha) \end{cases}$$

Cette propriété signifie que la valeur assignée à l'attribut $\alpha$ de l'élément structurel $o_k$ est en relation avec les valeurs attachées aux attributs $\alpha$ des éléments structurels $o_i$ et $o_j$.

**Exemple 6.10.3.3**
*Sur notre exemple de la figure 6.13, la combinaison pour l'attribut* auteur *donnée par la fonction* $combine_{auteur}$ *correspond à une agrégation de valeurs. La fonction de propagation* $f_\alpha$ *est l'identité.*

*La définition initiale des portées sur les éléments structurels* $o_2$ *et* $o_5$ *est donnée par :*

$$\mathcal{P}^{o_2}_{auteur,comp} = (auteur, o_2, \prec_{comp}, pred, o_2.auteur, f_{auteur}, cond_2)$$
$$et$$
$$\mathcal{P}^{o_5}_{auteur,comp} = (auteur, o_5, \prec_{comp}, pred, o_5.auteur, f_{auteur}, cond_5)$$

*devient*

$$(auteur, o_2, \prec_{comp}, pred, o_2.auteur, f_{auteur}, o_c \in p_2)$$
$$et$$
$$(auteur, o_5, \prec_{comp}, pred, o_5.auteur, f_{auteur}, o_c \in p_5)$$
$$et$$
$$(auteur, o_1, \prec_{comp}, pred, combine_{auteur}(o_2.auteur, o_3.auteur), f_{auteur}, o_c \in p_{(2,5)} \wedge o_c \neq o_1)$$

*Avec* $o_2.auteur = \{"Darrieussecq"\}$ *et* $o_5.auteur = \{"Echenoz"\}$*, nous obtenons la valeur suivante pour l'élément* $o_1$ *(élément sur lequel est définie une nouvelle portée) :*

$$\begin{aligned} o_1.auteur = \; & combine_{auteur}(o_2.auteur, o_5.auteur) \\ = \; & o_2.auteur \cup o_5.auteur = \{"Darrieussecq", "Echenoz"\} \\ & \Rightarrow \begin{cases} o_2.auteur \subseteq o_1.auteur \\ et \\ o_5.auteur \subseteq o_1.auteur \end{cases} \end{aligned}$$

### 6.10.4 Combinaison des conditions d'appartenance

Lors de la combinaison des portées, nous devons particulièrement nous intéresser aux conditions d'appartenance. Ce sont ces conditions, exprimées pour chaque élément source

d'un attribut, qui indiquent quels éléments sont concernés par l'attribut. Lors des résolutions de conflits, nous avons rencontré plusieurs cas :

**Mise à jour** d'un ensemble $EP^{o}_{\alpha,rel}$ décrivant la portée de l'un des deux éléments sources en conflit. Cette mise à jour se fait par l'ajout d'une condition dans la définition de la portée de l'élément structurel $o$.

- Cas général de la résolution de conflits de la catégorie *succ*.
- Cas général de la résolution de conflits de la catégorie *pred*.

**Fusion** de deux portées. L'une des deux portées en conflit est absorbée par l'autre portée. L'élément source de la portée absorbée est retirée de l'ensemble des éléments sources.

- Cas particulier de la résolution de conflits de la catégorie *succ*.
- Cas particulier de la résolution de conflits de la catégorie *pred*.

**Création** d'une portée. La résolution de conflit nécessite la création d'un nouvel élément source ayant comme portée l'ensemble des éléments structurels communs aux deux portées initiales.

- Résolution de conflits de la catégorie *pred* dans le cas où les portées initiales des deux éléments ne sont pas disjointes et ne sont pas incluses l'une dans l'autre.

### 6.10.5 Propriété de la fonction de combinaison

La fonction de combinaison, $combine_{\alpha}$, permet de combiner les valeurs provenant de plusieurs sources. Ces multiples sources sont ici les éléments à l'origine des portées, c'est-à-dire les éléments de $OS_{\alpha}$. L'application de cette fonction introduit aussi des dépendances entre les valeurs représentées par la relation $C_{\alpha}$.

La relation dérivée de l'application de la fonction de combinaison décrit une partie de la *dépendance* des valeurs des attributs au sein du document structuré. C'est en effet par cette fonction que nous réalisons la seconde partie de notre objectif d'indexation du document structuré : rendre explicite les dépendances entre les valeurs des attributs.

Nous allons maintenant nous intéresser aux propriétés de cette fonction de combinaison. Cette fonction est indépendante du langage d'indexation, ce qui signifie que quel que soit le domaine de l'attribut $\alpha$, cette fonction doit respecter les critères définis ci-dessous. Nous faisons ce choix pour conserver l'indépendance du modèle vis-à-vis des langages d'indexation, mais aussi parce que nous nous situons dans un contexte particulier où il existe différentes sortes d'attributs de domaines de définition différents.

Nous avons présenté précédemment les résolutions de conflits en ne considérant pour chaque conflit que deux éléments structurels sources, donc deux ensembles décrivant des portées. Toutefois, nous nous situons dans une arborescence et il nous faut dorénavant considérer que les conflits ne se limitent pas à deux éléments. Considérons l'arborescence de la figure 6.14 dans laquelle l'élément structurel $o$ est composé de n éléments structurels, $o_1, o_2, \ldots, o_n$.

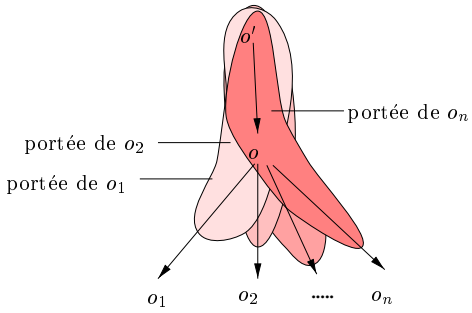



**Figure 6.14.** Arborescence N-aire

Les n éléments fils admettent un attribut de composition ascendant $\alpha$ c'est-à-dire avec une portée définie selon la relation de composition et dont la catégorie est *pred*.

*Méthode utilisant le mixage*

Si nous considérons les éléments structurels frères deux à deux, nous pouvons appliquer la résolution de conflits présentée précédemment (cas II, page 185). L'élément structurel $o$ admet un attribut $\alpha$ non valué et il est inclus dans chacune des portées de ses n éléments fils.

**Etape 1**: Soient les éléments $o_1$ et $o_2$ fils de l'élément $o$. La résolution du conflit lié à ces deux éléments fournit une valeur à l'attribut $\alpha$ de l'élément $o$ et réduit les portées respectives des éléments $o_1$ et $o_2$ à l'ensemble vide. Il s'agit ici de l'application de la fonction de combinaison en tant que *mixage des valeurs*.

$$o.\alpha = combine_\alpha(f_\alpha(o_1.\alpha), f_\alpha(o_2.\alpha))$$

Cette approche est équivalente à celle où nous considérons la valeur nulle pour l'attribut $\alpha$ de l'élément $o$ et considérons dans un premier temps un filtrage entre la valeur nulle et la valeur provenant de l'élément $o_1$ pour fournir une valeur à $\alpha$ de $o$ puis un mixage entre cette valeur et celle provenant de l'élément $o_2$ :

$$o.\alpha = combine_\alpha(f_\alpha(o_1.\alpha), NULL) = f_\alpha(o_1.\alpha)$$
*puis*
$$o.\alpha' = combine_\alpha(o.\alpha, f_\alpha(o_2.\alpha)) = combine_\alpha(f_\alpha(o_1.\alpha), f_\alpha(o_2.\alpha))$$

Nous obtenons dans les deux cas de figure la même valeur pour l'attribut $\alpha$ de l'élément structurel $o$: $combine_\alpha(f_\alpha(o_1.\alpha), f_\alpha(o_2.\alpha))$.

**Etape 2**: Dorénavant l'élément $o$ admet un attribut $\alpha$ ainsi qu'une portée décrite par l'intersection des ensembles $EP^{o_1}_{\alpha,comp}$ et $EP^{o_2}_{\alpha,comp}$. Les portées des éléments $o_1$ et $o_2$ ne connaissent plus de conflit puisque leurs portées ont été réduites. Nous avons maintenant des conflits dans lesquels l'une des portées est incluse dans l'autre: $EP^{o}_{\alpha,comp} \subset EP^{o_i}_{\alpha,comp}$ pour $i = 3, \ldots, n$.

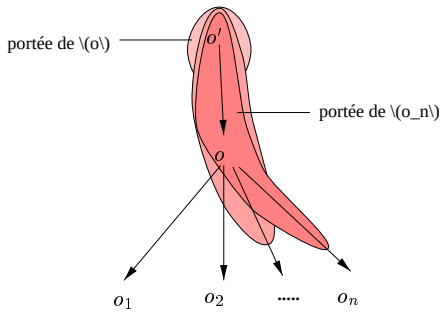



**Figure 6.15.** Arborescence N-aire

Nous appliquons la résolution décrite en page 184 qui permet de remonter les valeurs de l'attribut $\alpha$ inscrites au niveau des éléments $o_i$, $i = 3, \ldots, n$. Ainsi nous obtenons[5] :

$$
\begin{array}{ll}
i = 3 & o.\alpha^3 = combine_\alpha(o.\alpha, f_\alpha(o_3.\alpha)) \\
i = 4 & o.\alpha^4 = combine_\alpha(o.\alpha^3, f_\alpha(o_4.\alpha)) \\
\ldots & \\
i = n & o.\alpha^n = combine_\alpha(o.\alpha^{n-1}, f_\alpha(o_n.\alpha))
\end{array}
$$

Finalement, l'attribut $\alpha$ de l'élément $o$ prend la valeur $o.\alpha^{n-2}$. Il faut remarquer que nous avons ici choisi un ordre pour effectuer les combinaisons des valeurs. Afin que l'ordre de combinaison des éléments ne modifie pas le résultat final, nous imposons les deux propriétés suivantes sur la fonction de combinaison pour le *mixage* :

– Commutativité :

$$\forall a, b \in domain(\alpha),\ combine_\alpha(a, b) = combine_\alpha(b, a)$$

– Associativité :

$$
\begin{array}{l}
\forall a, b, c \in domain(\alpha), \\
combine_\alpha(a, combine_\alpha(b, c)) \quad = \quad combine_\alpha(combine_\alpha(a, b), c) \\
\qquad\qquad\qquad\qquad\qquad\quad = \quad combine_\alpha(combine_\alpha(a, c), b)
\end{array}
$$

Vérifions maintenant les propriétés engendrées entre les valeurs de l'attribut $\alpha$.

– conflit des portées des éléments $o_1$ et $o_2$ : nous combinons tout d'abord les valeurs des deux premiers éléments fils de l'élément $o$ pour donner une première valeur à l'attribut

5. la notation $o.\alpha^i$ signifie qu'il s'agit d'une valeur temporaire pour l'attribut $\alpha$ de l'élément $o$ obtenu à partir de i-1 combinaisons

$\alpha$ de $o$. L'élément structurel $o$ est aussi ajouté dans l'ensemble $OS_\alpha$ et il sera maintenant considéré comme un élément source.

$$o.\alpha = combine_\alpha(f_\alpha(o_1.\alpha), f_\alpha(o_2.\alpha)) \Rightarrow \begin{cases} C_\alpha(o_1.\alpha, o.\alpha) \\ et \\ C_\alpha(o_2.\alpha, o.\alpha) \end{cases}$$

– conflit des portées des éléments $o$ et $o_3$ : l'élément $o$ admet dorénavant un attribut $\alpha$ ainsi qu'une valeur, $o.\alpha^2$, et une portée. Le conflit se situe donc entre les portées de l'élément $o$ et celles des éléments fils de $o$.

$$\begin{aligned} o.\alpha^1 = combine_\alpha(o.\alpha, o_3.\alpha) \quad &= combine_\alpha(combine_\alpha(f_\alpha(o_1.\alpha), f_\alpha(o_2.\alpha)), o_3.\alpha) \\ &\Rightarrow \begin{cases} C_\alpha(o.\alpha, o.\alpha^1) \\ et \\ C_\alpha(o_3.\alpha, o.\alpha^1) \end{cases} \end{aligned}$$

A partir de l'existence de la relation $C_\alpha(o.\alpha, o.\alpha^1)$, la transitivité de la relation $C_\alpha$ nous donne :

$$C_\alpha(o_1.\alpha, o.\alpha^1) \text{ et } C_\alpha(o_2.\alpha, o.\alpha^1)$$

c'est-à-dire que pour tout élément $o_i$, $1 \leq i \leq 3$, nous avons : $C_\alpha(o_i.\alpha, o.\alpha^1)$.

– conflit des portées des éléments $o$ et $o_k$, $3 < k < n$ :

$$o.\alpha^{k-2} = combine_\alpha(o.\alpha^{k-3}, o_k.\alpha) \Rightarrow \begin{cases} C_\alpha(o.\alpha^{k-3}, o.\alpha^{k-2}) \\ et \\ C_\alpha(o_k.\alpha, o.\alpha^{k-2}) \end{cases}$$

A partir de l'existence de la relation $C_\alpha(o.\alpha^{k-3}, o.\alpha^{k-2})$, la transitivité de la relation $C_\alpha$ nous donne :

$$\forall o_i,\ 1 \leq i \leq k,\ C_\alpha(o_i.\alpha, o.\alpha^{k-2})$$

– conflit des portées des éléments $o$ et $o_n$ :

$$o.\alpha^{n-2} = combine_\alpha(o.\alpha^{n-3}, o_n.\alpha) \Rightarrow \begin{cases} C_\alpha(o.\alpha^{n-2}, o.\alpha^{n-3}) \\ et \\ C_\alpha(o_n.\alpha, o.\alpha^{n-3}) \end{cases}$$

Nous aboutissons donc à la conclusion suivante pour tout les éléments $o_i$ fils de l'élément $o$ :

$$\forall o_i,\ i = 1, \ldots, n\ :\ C_\alpha(o_i.\alpha, o.\alpha^{n-2})$$

Cette propriété signifie que pour tout élément $o_i$ fils de l'élément $o$, il existe la même relation de dépendance entre la valeur de l'attribut $\alpha$ de $o_i$ et celle de l'élément $o$ : $C_\alpha(o_i.\alpha, o.\alpha^{n-2})$ où $o.\alpha^{n-2}$ est la valeur de l'attribut $\alpha$ de l'élément$o$.

Les propriétés générées lors de chaque application de la fonction de combinaison sont donc toujours valable malgré l'évolution de la valeur de l'attribut $\alpha$ attachée à l'élément $o$.

L'intérêt des propriétés de *commutativité* et d'*associativité* devient ici évident. Elles nous permettent de vérifier que l'ordre de traitement des éléments fils de $o$ n'est pas contraignant puisque grâce à ces propriétés le résultat final, c'est-à-dire la valeur assignée à l'attribut $\alpha$ de $o$, est toujours identique et les propriétés liées à la relation $C_\alpha$ sont vérifiées.

*Méthode utilisant le filtrage*

Considérons maintenant la cas où l'attribut $\alpha$ de l'élément structurel est valué. c'est-à-dire non nul, et que nous appliquons un filtrage. Dans ce cas, la première résolution de conflit entre la portée de l'élément $o$ et celle d'un élément $o_i$ donne la réduction de la portée de l'élément $o_i$, l'ensemble $EP^{o_i}_{\alpha,comp}$ est vide, et la nouvelle valeur de $\alpha$ de $o$ que nous notons $o.\alpha'$ est :

$$o.\alpha' = combine_\alpha(o.\alpha, f_\alpha(o_i.\alpha)$$
$$\text{tel que } C_\alpha(o.\alpha, o.\alpha') \text{ et } \neg C_\alpha(o_i.\alpha, o.\alpha')$$

C'est-à-dire $o.\alpha' = o.\alpha$.

Quel que soit l'élément structurel $o_i$, la filtrage fournira toujours ce même résultat qui n'introduit pas de dépendance entre la valeur de l'attribut $\alpha$ de l'élément $o$ et les valeurs des attributs $\alpha$ des éléments $o_i$. La valeur de l'attribut $\alpha$ de l'élément $o$ sera inchangée et aucune relation $C_\alpha$ ne sera introduite dans l'ensemble $OS_\alpha$

## 6.11 Conclusion

Nous avons présenté un processus d'indexation pour les documents structurés. Ce processus reprend les caractéristiques premières du processus classique d'indexation des documents dans les systèmes de recherche d'information, à savoir l'extraction des connaissances et la représentation du document selon un modèle. Nous ajoutons à ce processus d'indexation une tâche spécifique aux documents structurés : l'intégration des caractéristiques structurelles des attributs.

Pour effectuer cette tâche, nous avons introduit la *portée des attributs*. Une portée se décrit comme une propriété permettant de propager les valeurs d'un attribut en suivant une relation de structure du document. L'intégration des portées des attributs au sein du document structuré permet d'une part d'augmenter la couverture du document structuré pour chaque attribut. Nous avons originalement, avant l'indexation structurelle, un ensemble d'éléments structurels ($OS^1_\alpha$) qui admettent une valeur pour un attribut $\alpha$ quelconque. Après cette indexation structurelle, nous avons un ensemble d'éléments structurels sources $OS_\alpha$ et des portées pour chaque élément structurel source $EP^o_{\alpha,rel}$. Avant l'indexation structurelle la

couverture était donnée par l'ensemble $OS^1_\alpha$. Après cette indexation, la couverture est décrite par l'ensemble des éléments structurels sources et les éléments des portées :

$$OS^1_\alpha \subseteq OS_\alpha \cup \bigcup_{o \in OS_\alpha} EP^o_{\alpha,rel}$$

Cette intégration permet d'avoir pour chaque attribut au sein du document structuré des dépendances explicites entre les valeurs présentes de cet attribut pour chaque élément structurel. Ces dépendances sont représentées par les relations $C_\alpha$, entre les éléments structurels sources ($o \in OS_\alpha$), et $R_\alpha$, entre les éléments structurels d'une portée ($o_i \in EP^o_{\alpha,rel}$).

L'introduction des portées a nécessité l'extension du modèle de document afin de garantir que le résultat de l'indexation structurelle soit conforme à un modèle de document, en l'occurrence le *modèle de document étendu*. La construction de ce modèle étendu dépend de la classification des attributs et doit donc être associé à une classe de documents partageant la même définition structurelle.

Dans [CFM96], nous avions défini trois types d'attributs : les attributs statiques, les attributs ascendants et les attributs descendants. La propagation des attributs se faisait uniquement selon la relation de composition et nous avions considéré une fonction de combinaison spécifique à chaque type d'attribut. D'autre part, nous spécifiions à part la structure d'indexation pour l'attribut de contenu sémantique. Ici, nous définissons au sein des portées la structure d'indexation et ceci pour chaque attribut et non plus seulement pour l'attribut de contenu sémantique. De plus, nous considérons deux types de propagation des valeurs d'attribut : l'une pouvant être considérée comme une simple propagation, $f_\alpha$, et l'autre autorisant la combinaison des valeurs, $combine_\alpha$.

Notre travail représente donc une généralisation et englobe la notion d'attributs dynamiques et la propagation des informations au sein du document structuré décrites dans [CK96, CFM96, Fou96, FM96]. Toutes les techniques d'indexation qui consistent à agréger les informations en les remontant des feuilles de la structure vers la racine [Khe95, CFM96, RF96, LYYB96] peuvent donc être représentées à l'aide de notre modèle. Nous pouvons ainsi reprendre au sein de notre modèle l'application PRIME [BFM+97, BFMM97, Fou97, MMFB97] en considérant un attribut de contenu sémantique ayant pour domaine un langage d'indexation basé sur le formalisme des graphes conceptuels.

Dans d'autres travaux cherchant à rapprocher les systèmes de gestion de bases de données des systèmes de recherche d'information [BR90, NBY95b, Chr96, ACC+97], l'accent est mis sur une plus grande flexibilité dans l'interrogation. Par exemple, cette flexibilité passe par l'introduction de variables d'attributs dans [BR90, Chr96, ACC+97] dans le langage de requêtes afin d'atteindre un maximum d'attributs et de limiter le besoin de connaissance de la structure des documents interrogés. Notre indexation structurelle accroît le nombre d'éléments structurels concernés par chaque attribut et permet donc d'atteindre des éléments que ces systèmes ne permettent pas d'atteindre puisqu'ils ne tiennent pas compte de la portée des attributs. De plus, nous limitons nous aussi le besoin de connaissance de la structure des documents puisque nous étendons le modèle en fonction des caractéristiques structurelles des attributs. Et il n'est donc plus utile d'avoir une requête conforme au modèle de document mais elle doit dorénavant être conforme au modèle étendu.

# Chapitre 7

# Interrogation des documents structurés

Après avoir défini un processus d'indexation intégrant l'utilisation des relations de structure du document, nous nous intéressons maintenant à l'interrogation des documents. Pour cela, nous allons décrire un modèle de correspondance des documents structurés selon les caractéristiques mises en évidence précédemment. Les éléments du modèle de correspondance sont déterminés par le processus d'indexation structurelle.

Dans un premier temps, nous caractérisons l'interrogation des documents structurés puis nous rappelons la forme du document que nous considérons, c'est-à-dire l'état de représentation provenant des traitements successifs subis par le document : indexation, extension du modèle de document, indexation structurelle. Nous voyons par la suite les stratégies de recherche que nous développons selon les propriétés de couverture et de dépendance des attributs développées avec l'indexation structurelle. Dans ces stratégies, nous évaluons, selon les types de requêtes, les éléments qui sont utiles à la résolution de la requête. Nous donnons aussi l'interprétation structurelle des opérateurs booléens disponibles pour construire une requête.

Dans ce chapitre, lorsque nous parlons d'une portée, il s'agit soit d'une portée selon la relation de séquence, soit d'une portée selon la relation de composition. Nous n'utilisons donc pas les portées définies à partir des relations de référence lors de l'interrogation car nous estimons qu'il s'agit d'un outil aidant à l'indexation qui ne génère pas de propriétés susceptibles d'être directement utilisées lors du processus de mise en correspondance des documents et de la requête.

## 7.1 Caractérisation de l'interrogation

Lorsque nous nous intéressons à des fonds constitués de documents structurés [Bur92], deux classes de requêtes sont classiquement distinguées : les requêtes structurelles et les re-

quêtes sur le contenu. La combinaison de ces deux classes est un des aspects de notre travail. La prise en compte de la structure dans les requêtes sur le contenu contribue à cette complémentarité.

Les travaux sur des langages de requêtes [BR90, Mac90, HK95, Nav95, Chr96] mettent en évidence les nouvelles fonctionnalités souhaitables pour intégrer la structure dans le processus de recherche. Ces travaux, que nous avons décrits dans le chapitre 2, montrent que l'interrogation des documents structurés peut être abordée selon différents aspects. Nous citons ici ceux qui nous semblent les plus intéressants.

Les systèmes MULTOS [BR90] et MAESTRO [Mac90] ont été parmi les premiers à proposer l'introduction d'opérateurs spécifiques aux documents structurés et une combinaison d'opérateurs structurels et de contenu. Les langages proposés adressent la spécification de caractéristiques de contenu alliées à des caractéristiques structurelles. Par la suite, Navarro [NBY96] cherche toujours à combiner ces deux caractéristiques en améliorant l'expressivité des opérateurs et en ayant pour objectif une efficacité souvent trop délaissée dans ces approches. Dans cette lignée qui consiste à ajouter des opérateurs à des langages de requêtes déclaratifs, Christophides cherche à assouplir ces langages en fournissant des opérateurs qui nécessitent une moindre connaissance de la structure [Chr96]. Ces nouvelles fonctionnalités augmentent l'expressivité des langages. Selon nous, il faut éviter les opérateurs nécessitant une connaissance exacte de la structure des documents pour pouvoir construire une requête.

Nous abordons l'interrogation des documents structurés indépendamment d'un langage de requêtes. En effet, nous préférons nous concentrer sur les stratégies de recherche liées à la découverte dans le document des parties les plus pertinentes. Il s'agit donc d'une approche plus proche de ce qui est pratiqué dans les systèmes de recherche d'informations. Nous conservons la consultation basée sur des primitives d'accès communes aux langages sur les documents structurés en les complétant par l'originalité et les spécificités de notre approche sur les attributs du document structuré.

### 7.1.1 L'espace d'information

Nous allons décrire notre approche en la comparant à une approche traditionnelle de la recherche d'informations. Dans un système traditionnel, le fonds documentaire constitue un espace d'informations constitué d'éléments de base. Ce sont des éléments de cet espace qui sont restitués comme réponses aux requêtes. Traditionnellement un élément de base est un document. En considérant des documents structurés, chaque document représente un espace d'informations au sein duquel les éléments de base sont les parties du document. Cette approche permet de restituer non plus un document complet, mais la ou les parties du document les plus proches de l'information requise. L'espace d'information décrit par le fonds documentaire est donc modifié puisque les éléments de base sont les parties de document et non plus les documents complets (figure 7.1).

Nous pouvons faire quelques remarques sur cette modification de l'espace d'informations considéré. Tout d'abord, le fait d'avoir un plus grand nombre d'éléments dans l'espace requiert de nouvelles stratégies de recherche mais surtout le système va pouvoir restituer de nouvelles réponses correspondant à des fragments de documents. Ceci permet de diminuer la granularité des réponses et de restituer des réponses moins perturbées par du bruit.

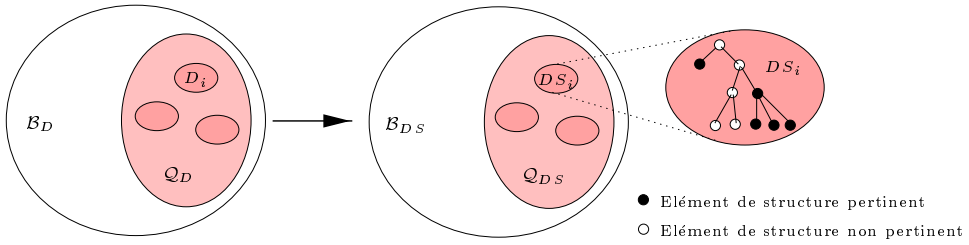



**Figure 7.1.** Modification des espaces d'information pour des fonds de documents structurés

Les systèmes classiques restituent un ensemble de documents jugés pertinents c'est-à-dire ceux étant les plus proches, selon le système, des besoins formulés par l'utilisateur dans sa requête. En introduisant la structure dans le processus global d'indexation et dans le processus d'interrogation, nous cherchons à mieux caractériser cet ensemble de documents en sélectionnant les espaces de cet ensemble qui satisfont au mieux l'utilisateur. Il s'agit donc de restituer non seulement un ensemble de documents mais surtout l'ensemble des éléments de structure qui répondent correctement.

Un document structuré est représenté par un ensemble d'éléments structurels, noté $OS$, qui correspond à l'espace d'information du document. Le sous-espace d'information qui correspond aux parties du document pertinentes à une requête correspond à un sous-ensemble de l'ensemble $OS$. C'est ce sous-espace que nous cherchons à caractériser lors de la résolution d'une requête. Sur la figure 7.1, nous représentons d'un côté un fonds documentaire traditionnel, noté $B_D$, et de l'autre côté un fonds constitué de documents structurés, noté $B_{DS}$. Lors de la résolution d'un requête $Q$ sur le fonds $B_D$, le système doit construire l'ensemble des documents qui répondent à $Q$, représenté sur la figure 7.1 par l'ensemble $Q_D$ de documents $D_i$. La résolution d'une requête $Q$ sur le fonds $B_{DS}$ nécessite la construction de l'ensemble des documents $DS_i$ qui répondent à $Q$, noté $Q_{DS}$. Cet ensemble, à la différence de l'ensemble $Q_D$, est constitué uniquement des éléments de structure qui sont pertinents et non plus de documents complets.

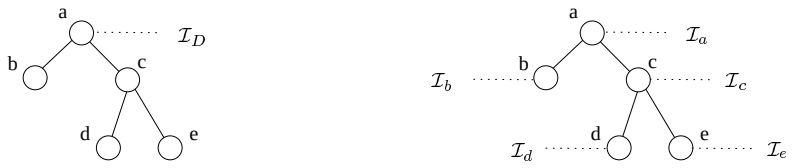



**Figure 7.2.** Une approche traditionnelle comparée à une approche dédiée à des fonds de documents structurés

Il s'agit ici d'une évolution du modèle de correspondance qui ne peut être mise en oeuvre que si le modèle de représentation des documents a lui aussi évolué. Nous donnons dans la figure 7.2 une caricature de cette évolution : une approche traditionnelle restituant un document complet nécessite une unique représentation associée au document complet, repré-

sentation que nous avons noté $\mathcal{I}_D$, alors que notre approche nécessite une représentation pour chaque élément de structure susceptible d'être une réponse à une requête: chaque composant $x$ admet une représentation $\mathcal{I}_x$.

Nous proposons pour notre part la notion de *structure d'indexation* pour chaque attribut qui décrit le document.

### 7.1.2 La structure d'indexation

De manière générale, les approches sur les documents structurés [LYYB96, RF96] considèrent que chaque noeud de l'arborescence structurelle est susceptible d'être une réponse à une requête et doit donc admettre une représentation pour décrire son contenu. Pour notre part, nous introduisons la notion de portée sur les attributs qui permet d'associer à chaque élément de structure les attributs ainsi que leurs valeurs qui les concernent. Outre que les portées permettent de mixer des informations provenant de médias différents, elles décrivent alors une structure d'indexation dépendante de chaque attribut. En effet, pour chaque attribut, nous obtenons en sortie du processus d'indexation structurelle deux ensembles de composants, notés $OS_\alpha$ et $OS^p_\alpha$. Ces deux ensembles proviennent de l'application des portées :

- $OS_\alpha$ est l'ensemble des éléments sources pour l'attribut $\alpha$, $OS_\alpha \subset OS$.
- $OS^p_\alpha$ est l'ensemble des éléments atteints par une portée définie sur un élément source pour l'attribut $\alpha$. Il s'agit de l'union des ensembles décrivant les portées de l'attribut $\alpha$ au sein du document.

$$OS^p_\alpha = \bigcup_{o \in OS_\alpha} (EP^o_{\alpha,rel})$$

Sur la figure 7.3, nous considérons un document $\mathcal{D}$ sur lequel sont définis deux attributs: $X$ et $Y$. Les composants $a$ et $c$ admettent l'attribut $X$ et le composant $a$ admet aussi l'attribut $Y$. L'état initial du document est donné par l'arborescence de gauche. Les trois arborescences de droite représentent quant à elles l'état final en considérant trois attributs: $X$, $Y$ et $I$ (l'attribut de contenu sémantique). Nous expliquons comment nous avons obtenu ces trois arborescences pour chaque attribut.

- l'attribut $X$ est un attribut de composition descendant ayant une fonction de combinaison, notée $combine_X$ qui mixe les valeurs et n'admettant pas de contraintes particulières de propagation.

  Nous considérons deux portées puisque deux éléments admettent cet attribut. La résolution du conflit au niveau de l'élément $c$ est de type *imbrication de portées*. Nous obtenons donc les informations suivantes :

  - $OS_X = \{a, c\}$ avec $a.X = X_a$ et $c.X = X'_c$. De plus, nous avons la relation suivante $C_X(X_a, X'_c)$ qui signifie la dépendance entre les deux valeurs.
  - $EP^a_{X,comp} = \{b\}$ et $EP^c_{X,comp} = \{d, e\}$

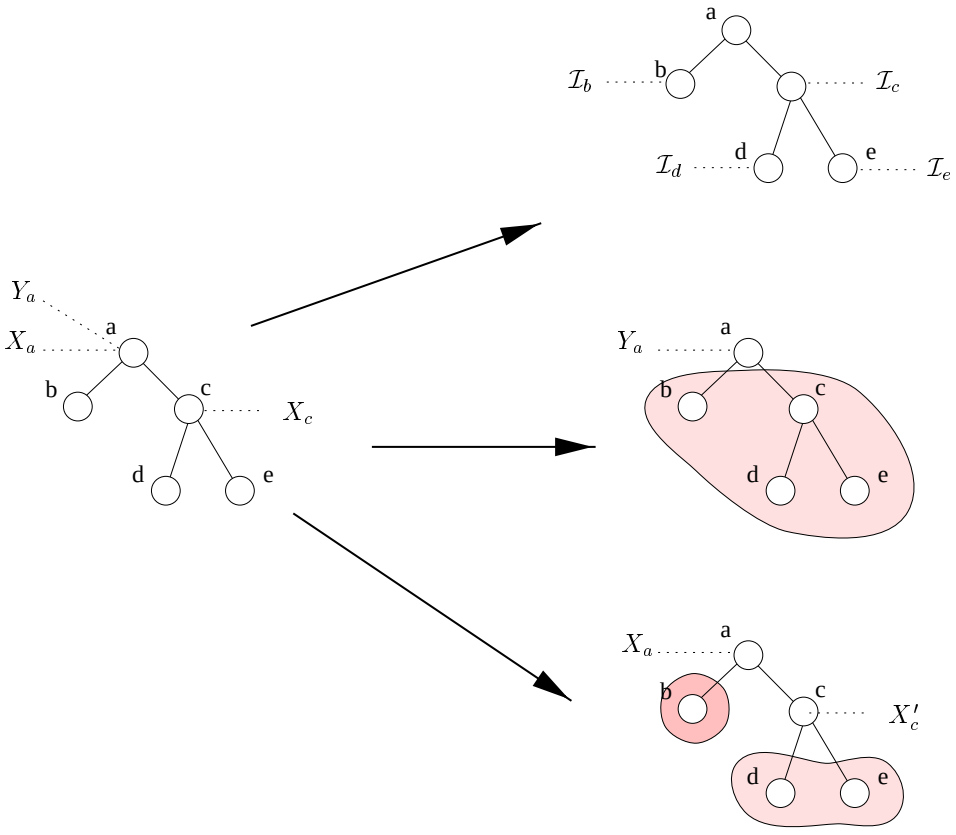



**Figure 7.3.** Notre approche dédiée à des fonds de documents structurés

Si le domaine de l'attribut $X$ est un ensemble de mots-clés, la relation $C_X$ correspond à l'inclusion, c'est-à-dire que nous avons $X_a \subset X'_c$.

- l'attribut $Y$ est un attribut de composition descendant admettant une contrainte de propagation: l'élément racine n'est pas indexé.

  La portée définie à partir de l'élément $a$ reprend tous les éléments du document $\mathcal{D}$:

  - $OS_Y = \{a\}$ avec $a.Y = Y_a$
  - $EP^a_{Y,comp} = \{b, c, d, e\}$

- l'attribut $I$ est un attribut de composition ascendant. Nous avons dit qu'il s'agissait de l'attribut de contenu sémantique. Nous posons donc comme hypothèse qu'un processus d'indexation classique a indexé les éléments feuilles de la structure, les éléments $b$, $d$ et $e$ reçoivent respectivement les valeurs $I_b$, $I_d$ et $I_e$. Après propagation nous obtenons alors :

  - $OS_I = \{b, c, d, e\}$ avec $b.I = I_b$, $c.I = I_c$, $d.I = I_d$ et $e.I = I_e$. De plus, nous avons la relation suivante $C_I(I_c, I_d)$ et $C_I(I_c, I_e)$ qui signifient la dépendance entre les valeurs.

– $\forall x \in OS_I$ nous avons $EP^x_{I,comp} = \emptyset$

Considérons que le domaine de cet attribut est basé sur le formalisme des graphes conceptuels, alors la relation $C_I$ correspond à une relation de spécialisation/généralisation, qui peut être interprétée par une implication logique. Nous avons alors les relations suivantes : $I_c \rightarrow I_d$ et $I_c \rightarrow I_e$.

Nous voyons à travers cet exemple que chaque attribut possède sa propre structure d'indexation et que des relations lient les valeurs de ces attributs selon les propriétés engendrées par l'application des portées.

### 7.1.3 Modèle de correspondance

Le modèle de correspondance, c'est-à-dire le processus de recherche, repose sur l'utilisation des ensembles issus des portées tout en assurant la cohérence des réponses. Il doit ainsi éviter l'apparition de redondance dans les réponses qui se caractérise par la présence d'un élément de structure et de l'un de ses parents dans l'ensemble des réponses, mais nous verrons où se situe ce traitement par la suite.

Nous établissons deux règles afin de spécifier les choix que nous faisons dans les réponses que nous restituons à un utilisateur.

**Règle 7.1.3.1**
*Pour un document structuré et pour une requête donnée, nous restituons un ensemble d'éléments structurels pertinents. Cet ensemble prend la forme d'une forêt d'arborescences.*

Nous montrons l'application de cette règle sur la figure 7.1. Les éléments structurels noirs du document $DS_i$ sont les éléments qui répondent à une requête. Ces éléments forment la réponse qui doit être restituée à l'utilisateur sous la forme d'une forêt d'arborescences.

**Règle 7.1.3.2**
*Pour un document structuré et pour une requête donnée, nous privilégions les éléments structurels qui sont "source d'information".*

Nous rappelons que les éléments sources d'information sont les éléments à partir desquels sont décrites des portées. Nous allons les différencier des autres éléments structurels lors du processus de correspondance et ils seront privilégiés dans les réponses. Cela signifie que nous allons en premier lieu parcourir l'ensemble des éléments sources avant de consulter l'ensemble des éléments appartenant aux portées définies sur ces éléments sources. De plus le parcours de l'ensemble des éléments sources va utiliser les dépendances entre les valeurs générées par l'application des portées.

Nous donnons maintenant plus formellement les éléments caractéristiques des documents structurés.

## 7.2 Caractérisation des documents structurés

Nous rappelons ici les caractéristiques fondamentales des documents structurés de notre modèle qui peuvent être mises en jeu lors d'une phase de correspondance. Ces caractéristiques proviennent tout d'abord du modèle de document initial puis des extensions que nous avons intégrés, à savoir l'introduction des portées des attributs définis dans le modèle initial.

*Le Modèle de Document (MD)*

Le modèle de document, établi pour représenter les documents structurés, comporte plusieurs types d'informations relatives à la structure du document, aux types des composants d'un document et à leur contenu. Le document structuré est défini comme une arborescence d'éléments de structure typés. Notre modèle de document comporte donc deux relations de structure, la relation de composition, notée $\prec_{comp}$, et la relation de séquence, notée $\prec_{seq}$, qui permettent de définir d'une part l'agrégation des éléments de structure et d'autre part leurs enchaînements.

A partir de ces relations de structure sont définis des opérateurs déterminant l'ensemble des prédécesseurs et l'ensemble des successeurs d'un élément de structure: $Next_{rel}$ et $Prev_{rel}$ où $rel$ est soit la composition, soit la séquence.

Les types de documents (Livre, Article, ...) introduits comportent des contraintes sur la composition des éléments de structure. Ces contraintes permettent de spécifier une application valide des deux relations de structure. Il s'agit d'un ensemble de types ($TYPE_{ST}$) organisés par deux relations définies sur cet ensemble ($\preceq_{tcomp}$ et $\preceq_{tseq}$).

A chaque type d'élément de structure correspond un ensemble d'attributs. Ces attributs sont nommés attributs structurels car ils dépendent des types d'éléments structurels. La relation $\Re_{ST}$ détermine quels attributs pris dans l'ensemble $NAME$ sont admis par les types structurels de $TYPE_{ST}$.

Enfin, le modèle de document comporte des informations sur les médias du document. Pour chaque élément de structure, il est possible d'obtenir le ou les médias. De plus, il existe des informations sur le contenu de ces éléments structurels qui sont décrites par des attributs dits attributs médias. C'est la relation $\Re_M$ qui détermine ces attributs pour chaque type de média de l'ensemble $MEDIA$.

*Le Document Structuré (DS)*

Le document structuré se conforme au modèle de document (MD) c'est-à-dire qu'il correspond à un type de document défini dans le modèle et il est représenté par un ensemble d'éléments structurels typés, ensemble $OS$, liés par les trois relations de structure du modèle. Il s'agit ici d'instances.

Par ailleurs ces éléments sont décrits par des attributs conformément au modèle de document. Les éléments structurels admettant un attribut $\alpha$ conformément à leur type ou à leur média sont accessibles via l'ensemble $OS^1_\alpha$. Pour tout attribut $\alpha$, il existe un ensemble $OS^1_\alpha$ qui contient l'ensemble des éléments structurels admettant $\alpha$, $OS^1_\alpha \subseteq OS$.

Les fonctions $Next_{rel}$ et $Prev_{rel}$ sont ici utilisables sur chacun des éléments structurels pour déterminer l'ensemble de leurs prédécesseurs et l'ensemble de leurs successeurs selon une relation de structure $rel$.

*Le Modèle de Document Etendu (MDe)*

Nous ajoutons au modèle de document les caractéristiques provenant de l'introduction des portées. Au niveau du modèle de document, cela signifie que les relations $\Re_{ST}$ et $\Re_M$, qui permettent de déterminer quels attributs sont valides pour quels éléments, sont étendus selon les définitions des portées. Nous les notons $\Re'_{ST}$ et $\Re'_M$. Un élément d'un type de structure donné va pouvoir admettre de nouveaux attributs par rapport à ce qui était autorisé dans le modèle de document initial. Considérons l'ensemble $A_t$ des attributs structurels admis par les éléments de type $t$, $t \in TYPE_{ST}$ dans le modèle de document et l'ensemble $A'_t$ dans le modèle de document étendu.

$$A_t = \{a \mid a \in NAME \wedge \Re_{ST}(t, a)\}$$

$$A'_t = \{a \mid a \in NAME \wedge \Re'_{ST}(t, a)\} \text{ et } A_t \subseteq A'_t$$

*Le Document Structuré Etendu (DSe)*

Dans le document structuré étendu, les portées des attributs sont instanciées et la couverture du document pour chaque attribut est étendue conformément au modèle de document étendu. Pour tout attribut $\alpha$, il existe un ensemble $OS_\alpha$ qui contient l'ensemble des éléments structurels admettant $\alpha$. L'ensemble $OS^1_\alpha$ est un sous-ensemble de $OS_\alpha$, ce qui signifie bien que la couverture a été étendue.

$$\forall \alpha \in NAME,\ OS^1_\alpha \subseteq OS_\alpha \text{ et } OS_\alpha \subseteq OS$$

D'autre part, l'instanciation des portées des attributs fournit des propriétés sur les valeurs de ces attributs au sein du document structuré étendu. Ces propriétés n'existaient pas auparavant.

Soit l'ensemble $OS_\alpha$ des *éléments sources*. La relation $C_\alpha$ définie sur les valeurs des attributs $\alpha$ des éléments de $OS_\alpha$ indique la dépendance entre leurs valeurs. Cette dépendance (ou propriété) que nous notons $C_\alpha(o_i.\alpha, o_j.\alpha)$ est déduite de la fonction de combinaison et elle est donc complètement dépendante du domaine de l'attribut et des caractéristiques de cette fonction. La relation $C_\alpha$ n'est pas totale sur l'ensemble considéré mais elle donne toutes les dépendances entre valeurs d'attributs $\alpha$ sur les éléments de $OS_\alpha$.

Par ailleurs, les portées génèrent des ensembles spécifiques à chaque élément de l'ensemble $OS_\alpha$. Ces ensembles sont disjoints deux à deux et sont notés $EP^o_{\alpha,rel}$ pour chaque élément $o$ de $OS_\alpha$. Les éléments d'un ensemble $EP^o_{\alpha,rel}$ sont caractéristiques de l'élément $o$ puisque cet élément peut être considéré comme leur unique source d'information pour l'attribut $\alpha$.

Les valeurs des attributs $\alpha$ des éléments $o_k$ d'un ensemble $EP^o_{\alpha,rel}$ sont construites à partir de la fonction de propagation notée $f_\alpha$ et définie dans la portée d'un attribut $\alpha$. La relation $R_\alpha$ joue un rôle similaire à $C_\alpha$, mais à l'intérieur de chaque ensemble $EP^o_{\alpha,rel}$ décrivant la portée d'un élément $o$.

*Synthèse*

Au final, nous distinguons quatre sortes de connaissance qui peuvent être utilisées lors du processus d'interrogation : le modèle de document initial (MD), le document structuré (DS)

conforme au modèle de document, le modèle de document étendu (MDe) et enfin le document structuré étendu (DSe) conforme au modèle de document étendu. C'est notre processus d'indexation qui nous permet de distinguer ces connaissances puisque par l'introduction des portées nous modifions le modèle de document et l'adaptons aux portées.

Un processus classique d'indexation utilise un unique modèle de document auquel les documents de la base doivent se conformer. Nous introduisons une processus d'indexation structurelle qui implique une extension du modèle de document et par conséquent une extension des documents.

## 7.3 Quelles requêtes pour quels résultats ?

Notre objectif n'est pas ici de décrire de manière exhaustive toutes les requêtes qui peuvent être rencontrées, mais nous souhaitons mettre en évidence, sur des requêtes qui nous semblent suffisamment représentatives, l'originalité de notre travail ainsi que la valeur ajoutée qui en découle.

Nous présentons donc différentes requêtes que nous traitons sur un exemple de document structuré, pour lequel nous donnons son état initial, la définition des portées des attributs et enfin le document structuré étendu.

Nous donnons pour chaque requête les réponses que nous sommes capables de fournir ainsi que les informations et connaissances utilisées pour le traitement de la requête. Nous aboutissons ainsi à une classification des requêtes qui ne reposent plus seulement sur les caractéristiques structurelles ou de contenu de la requête mais sur les propriétés du modèle.

### 7.3.1 Un exemple de document structuré

Nous introduisons un exemple de document structuré de type livre selon les spécifications du type de document donné en section 6.1 (page 138). Avant de présenter l'état des attributs de ce document, nous rappelons les définitions issues du modèle de document initial et qui contraignent l'existence de ces attributs dans le document structuré :

$\Re_{ST}(Chapitre, Status)$ $\Re_{ST}(Chapitre, NumCh)$ $\Re_{MEDIA}(texte, Contenu)$
$\Re_{ST}(Chapitre, Auteur)$ $\Re_{ST}(Section, NumSec)$ $\Re_{MEDIA}(image, Contenu)$
$\Re_{ST}(Section, Auteur)$

Nous donnons une description générale de la forme de ces documents dans l'exemple de document structuré de la figure 7.4. Dans ce schéma, nous avons représenté les liens de composition et les liens de séquence, et nous donnons aussi les noms des attributs qui décrivent chaque élément.

Dans l'état initial, les attributs sont valués avec les valeurs données dans le tableau de la figure 7.5. Les "-" correspondent à des attributs non-valués ou bien qui ne sont pas définis pour le type d'élément de structure dans le DTD.

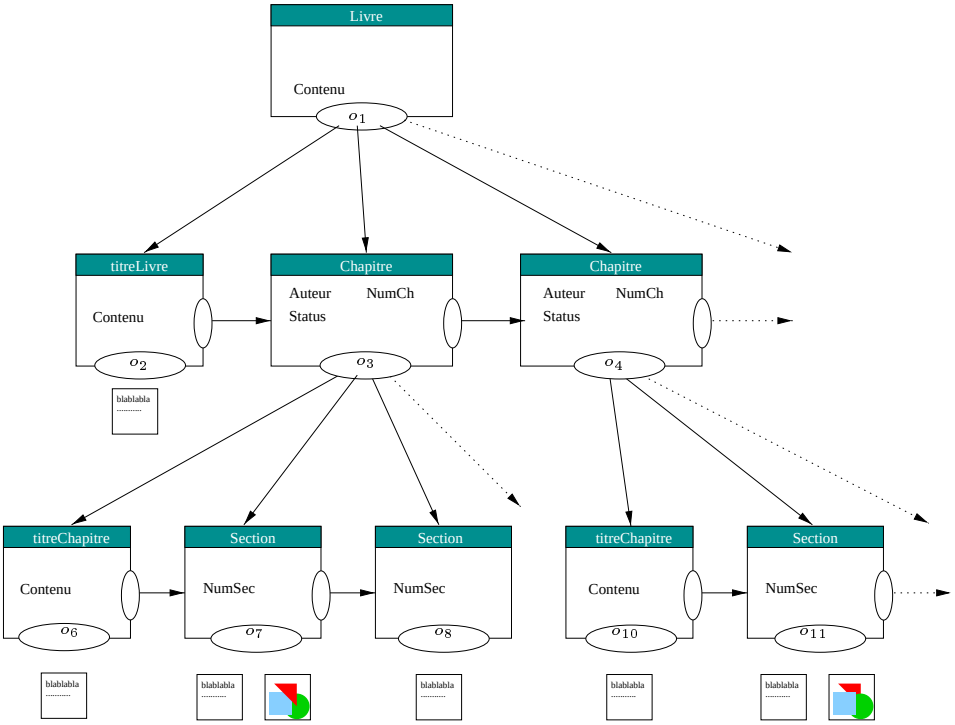



**Figure 7.4.** Exemple de document structuré

| Attributs | média | Id | Status | Contenu | Auteur | NumCh | NumSec |
|---|---|---|---|---|---|---|---|
| Livre | {texte, image} | $o_1$ | - | - | - | - | - |
| Titre | {texte} | $o_2$ | - | $\emptyset$ | - | - | - |
| Chapitre | {texte,image} | $o_3$ | draft | - | - | 1 | - |
| Chapitre | {texte, image} | $o_4$ | final | - | - | - | - |
| ... | | | | | | | |
| TitreChapitre | {texte} | $o_6$ | - | $\emptyset$ | - | - | - |
| Section | {texte, image} | $o_7$ | - | {SRI} | {B} | - | 1 |
| Section | {texte} | $o_8$ | - | {Logique} | {A} | - | - |
| ... | | | | | | | |
| TitreChapitre | {texte} | $o_{10}$ | - | $\emptyset$ | - | - | - |
| Section | {texte} | $o_{11}$ | - | $\emptyset$ | {A,C} | - | 1 |
| Section | {texte, image} | $o_{12}$ | - | {BD, Algèbre} | {A} | - | - |
| ... | | | | | | | |

**Figure 7.5.** Valeur des attributs d'un document structuré conforme au modèle de document initial

Les portées des cinq attributs que nous considérons dans cet exemple sont définies de la manière suivante :

Auteur : attribut de composition ascendant selon la relation de composition. La fonction de propagation est l'identité et la fonction de combinaison l'union ensembliste. La portée est alors définie de la manière suivante:

$$(OS_{Auteur}, \cup_{Auteur}, \mathcal{P}_{Auteur,comp})$$

$$\text{avec } \begin{cases} OS_{Auteur} = \{o_7, o_8, o_{11}, o_{12}\} \\ et \\ \mathcal{P}^o_{Auteur,comp} = (Auteur, o, \prec_{comp}, pred, o.Auteur, Id_{Auteur}, \emptyset) \end{cases}$$

Soit : $EP^{o_8}_{Auteur,comp} = \{o_3\}$ et $EP^{o_k}_{Auteur,comp} = \emptyset$, $k \neq 8$

Les informations suivantes sont introduites dans le modèle de document étendu : les éléments de type section, chapitre et livre peuvent dorénavant admettre l'attribut Auteur. Nous donnons ici le passage de la relation $\mathfrak{R}_{ST}$ du modèle de document initial à la relation $\mathfrak{R}'_{ST}$ du modèle de document étendu.

$$\begin{matrix} \mathfrak{R}_{ST}(Section, Auteur) \\ \mathfrak{R}_{ST}(Chapitre, Auteur) \end{matrix} \text{ et } \mathcal{P}_{Auteur,comp} \Rightarrow \begin{cases} \mathfrak{R}'_{ST}(Livre, Auteur) \\ \mathfrak{R}'_{ST}(Section, Auteur) \\ \mathfrak{R}'_{ST}(Chapitre, Auteur) \end{cases}$$

Contenu : portée ascendante selon la relation de composition. La fonction de propagation est l'identité et la fonction de combinaison l'union ensembliste.

$$(OS_{Contenu}, \cup_{Contenu}, \mathcal{P}_{Contenu,comp})$$

$$\text{avec } \begin{cases} OS_{Contenu} = \{o_1, o_3, o_4, o_7, o_8, o_{12}\} \\ et \\ \mathcal{P}^o_{Contenu,comp} = (Contenu, o, \prec_{comp}, pred, o.Contenu, Id_{Contenu}, \emptyset) \end{cases}$$

Soit : $EP^{o_{12}}_{Contenu,comp} = \{o_4\}$ et $EP^{o_k}_{Contenu,comp} = \emptyset$, $k \neq 12$

Conformément au modèle de document étendu tous les éléments de ce type de document peuvent dorénavant admettre l'attribut contenu. L'expression de cette portée permet principalement de n'indexer que les éléments feuilles et de remonter les informations contenues dans ces éléments vers la racine du document structuré.

Status : portée descendante selon la relation de composition. La fonction de propagation et a fonction de combinaison correspondent à l'identité.

$$(OS_{Status}, combine_{Status}, \mathcal{P}_{Status,comp})$$

$$\text{avec } \begin{cases} OS_{Status} = \{o_3, o_4\} \\ et \\ \mathcal{P}^o_{Status,comp} = (Status, o, \prec_{comp}, succ, o.Status, Id_{Status}, \emptyset) \end{cases}$$

Soit : $EP^{o_3}_{Status,comp} = \{o_6, o_7, o_8\}$ et $EP^{o_4}_{Status,comp} = \{o_{10}, o_{11}, o_{12}\}$

Les informations suivantes sont introduites dans le modèle de document étendu : les éléments de type chapitre, titrechapitre et section ainsi que leurs sous-types peuvent dorénavant admettre l'attribut Status.

$$\mathfrak{R}_{ST}(Chapitre, Status) \text{ et } \mathcal{P}_{Status,comp} \Rightarrow \begin{cases} \mathfrak{R}'_{ST}(TitreChapitre, Status) \\ \mathfrak{R}'_{ST}(Chapitre, Status) \\ \mathfrak{R}'_{ST}(Section, Status) \end{cases}$$

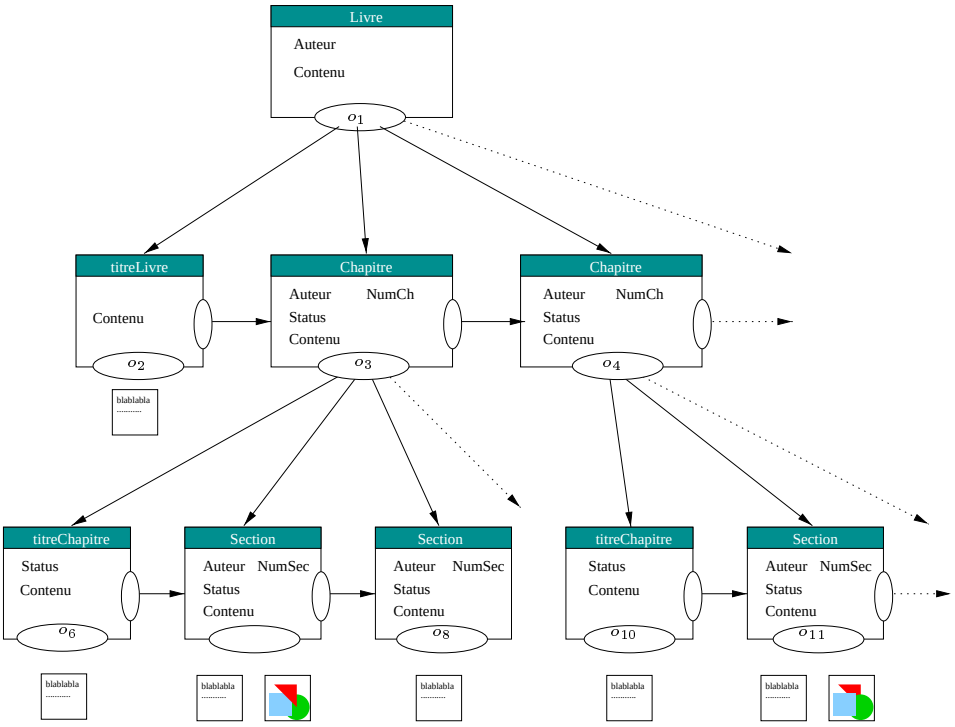



**Figure 7.6.** Exemple de document structuré étendu

NumCh : portée avant selon la relation de séquence. La fonction de propagation et la fonction de combinaison ajoutant 1 à la valeur précédente. Cet attribut s'applique uniquement sur des éléments de type chapitre.

$$(OS_{NumCh}, combine_{NumCh}, \mathcal{P}_{NumCh,comp})$$

$$\text{avec } \begin{cases} OS_{NumCh} = \{o_3\} \\ et \\ \mathcal{P}^{o}_{NumCh,seq} = (NumCh, o, \prec_{seq}, succ, o.NumCh, Add_{NumCh}, type_{str}(o_k) = Chapitre) \end{cases}$$

Soit : $EP^{o_3}_{NumCh,seq} = \{o_4\}$

NumSec : portée avant selon la relation de séquence. La fonction de propagation et la fonction de combinaison ajoutant 1 à la valeur précédente. Cet attribut s'applique uniquement sur des éléments de type section.

$$(OS_{NumSec}, combine_{NumSec}, \mathcal{P}_{NumSec,comp})$$

$$\text{avec } \begin{cases} OS_{NumSec} = \{o_7, o_{11}\} \\ et \\ \mathcal{P}^{o}_{NumSec,seq} = (NumSec, o, \prec_{seq}, succ, o.NumSec, Add_{NumSec}, type_{str}(o_k) = Section) \end{cases}$$

Soit : $EP^{o_7}_{NumSeq,seq} = \{o_8\}$ et $EP^{o_{11}}_{NumSeq,seq} = \{o_{12}\}$

Le tableau suivant résume l'état dans lequel se trouve la représentation du document structuré après le processus d'indexation structurelle. Lors de l'interrogation, nous allons faire référence aux valeurs qui se trouvent dans ce tableau.

| Attributs | média | Id | Status | Contenu | Auteur | NumCh | NumSec |
|---|---|---|---|---|---|---|---|
| Livre | {texte, image} | $o_1$ | - | {*SRI, Logique, BD, Algèbre*} | {*A,B,C*} | - | - |
| Titre | {texte} | $o_2$ | - | $\emptyset$ | - | - | - |
| Chapitre | {texte, image} | $o_3$ | draft | {*SRI, Logique*} | {*A,B*} | 1 | - |
| Chapitre | {texte, image} | $o_4$ | final | {*BD, Algèbre*} | {*A,C*} | *2* | - |
| ... | | | | | | | |
| TitreChapitre | {texte} | $o_6$ | *draft* | $\emptyset$ | - | - | - |
| Section | {texte, image} | $o_7$ | *draft* | {SRI} | {B} | - | 1 |
| Section | {texte} | $o_8$ | *draft* | {Logique} | {A} | - | *2* |
| ... | | | | | | | |
| TitreChapitre | {texte} | $o_{10}$ | *final* | $\emptyset$ | - | - | - |
| Section | {texte} | $o_{11}$ | *final* | $\emptyset$ | {A,C} | - | 1 |
| Section | {texte, image} | $o_{12}$ | *final* | {BD, Algèbre} | {A} | - | *2* |
| ... | | | | | | | |

**Figure 7.7.** Valeur des attributs d'un document structuré conforme à son modèle de document étendu

### 7.3.2 Des exemples de requêtes

Lors de la résolution de chacune des requêtes, nous indiquons d'où viennent les connaissances utilisées, du modèle de document initial, du document initial, du modèle de document étendu ou du document étendu. Nous fournissons pour certaines requêtes un classement des

réponses qui va dépendre de la provenance des connaissances. Il s'agit dans la résolution des requêtes d'exploiter les caractéristiques générées par les portées.

La requête suivante est une requête générique que tout système proposant des fonctionnalités de recherche sur des fonds constitués de documents structurés est capable de traiter.

**Requête 1** *Donnez les titres de chapitre des documents de type livre*

La requête 1 est une requête purement structurelle, c'est-à-dire que les critères spécifiés dans cette requête portent uniquement sur des caractéristiques structurelles: les types des éléments structurels et la composition des éléments. Il faut noter que la spécification de ce type de requête nécessite de la part de l'utilisateur une connaissance sur la structure des documents de la base de documents interrogée.

Dans le traitement de cette requête nous pouvons distinguer deux phases: la première contribue à la vérification des types des éléments structurels et la seconde porte sur les instances.

Dans la première phase, il est tout d'abord nécessaire de reconnaître les types d'éléments structurels spécifiés dans la requête, c'est-à-dire titreChapitre et Livre. Nous devons par ailleurs vérifier que ces types sont compatibles, autrement dit que la requête peut admettre un résultat. Ces types et leurs contraintes structurelles sont définis au niveau du modèle de document.

Le modèle de document est utilisé pour résoudre cette première phase, et l'on obtient un sous-ensemble des types d'éléments structurels qui vérifient les spécifications de la requête.

Dans la seconde phase, nous pouvons nous intéresser aux instances, c'est-à-dire aux documents structurés qui sont conformes aux types de documents précédemment sélectionnés. Cette résolution fait donc appel au modèle de document initial et au document structuré initial.

| Connaissances pour résoudre la requête 1: (MD) + (DS) |
|---|

**Requête 2** *Donnez les chapitres contenant au moins 3 sections.*

La requête 2 est assez similaire à la requête 1 dans le sens où elle contient uniquement des spécifications structurelles. En plus des types d'éléments structurels: Chapitre et Section intervient dans cette requête une condition sur la composition et sur le nombre d'éléments minimum de cette composition.

Nous sélectionnons tout d'abord les types de document correspondant aux types d'éléments spécifiés dans la requête. Ensuite, nous effectuons le traitement au niveau des instances, c'est-à-dire sur les éléments structurels des documents correspondant aux types présélectionnés.

$$R_0 = \{o \mid o \in OS \wedge type_{str}(o) = Chapitre\}$$

A partir de l'ensemble $R_0$, il s'agit de travailler sur la cardinalité des ensembles donnés par la fonction $Next_{comp}$ donnant l'ensemble des descendants structurels en y ajoutant les conditions sur les types d'éléments.

$$R = \{o \mid o \in R_0 \wedge |\{o_i \mid type_{str}(o_i) = Section \wedge o_i \in Next_{comp}(o)\}| \geq 3\}$$

Connaissances pour résoudre la requête 2 : (MD) + (DS)

**Requête 3** *Donnez les éléments contenant une image*

Il s'agit ici de l'utilisation de $MEDIA$ provenant du modèle de document initial, puis du passage aux instances, à savoir les documents structurés vérifiant la condition.

L'ensemble $R_0$ fournit l'ensemble des éléments de type image.

$$R_0 = \{o \mid o \in OS \wedge image = media(o)\}$$

A partir de $R_0$ et avec l'utilisation de la fonction $Prev_{comp}$ donnant accès aux ascendants structurels, nous retrouvons l'ensemble des éléments contenant une image.

$$R = \bigcup_{o \in R_0} (Prev_{comp}(o))$$

Cet ensemble doit être réduit en ne conservant que les ascendants structurels directs pour éviter les redondances d'informations dans l'ensemble $R$.

Connaissances pour résoudre la requête 3 : (MD) + (DS)

**Requête 4** *Donnez les titres de chapitre qui sont à l'état de draft.*

Nous devons considérer le type d'élément structurel titreChapitre et l'attribut status. Selon le modèle de document de notre exemple, le type titreChapitre n'admet pas l'attribut status. En effet la relation $\mathfrak{R}_{ST}(titreChapitre, Status)$ n'existe pas. A partir de ce seul modèle de document, le système retournerait un ensemble vide comme réponse à cette requête.

Nous utilisons alors le modèle de document étendu qui a été construit avec un attribut Status admettant une portée de catégorie *succ*: la relation $\mathfrak{R}'_{ST}(titreChapitre, Status)$ existe donc car le type *titreChapitre* est un sous-type de *Chapitre*.

Nous avons d'une part la contrainte sur le type structurel et d'autre part la contrainte sur la valeur de l'attribut Status. Afin de retrouver les éléments qui vérifient la contrainte sur la valeur de l'attribut, nous introduisons les ensembles $R.Status_s$ et $R.Status$ qui correspondent respectivement à l'ensemble des éléments structurels sources du document qui vérifient la spécification faite sur l'attribut Status dans la requête et à l'ensemble complet des éléments structurels du document qui vérifient cette même spécification :

*Les éléments structurels qui répondent exactement* : il s'agit de déterminer, pour l'attribut mono-valué Status l'ensemble des éléments structurels qui vérifient la spécification faite sur cet attribut dans la requête 4. Nous distinguons deux ensembles; l'ensemble $R.Status_s$ et l'ensemble $R.Status$ :

$$R.Status_s = \{o_s \mid o_s \in OS_{Status} \wedge o_s.Status = q.Status\}$$

L'ensemble $R.Status_s$ est uniquement composé d'éléments sources puisqu'il est obtenu en parcourant l'ensemble des éléments sources $OS_{Status}$.

- la fonction de propagation correspond à l'identité.

$$R.Status = R.Status_s \cup \{o \mid o \in EP^{o_s}_{Status,comp} \wedge o_s \in R.Status_s\}$$

L'ensemble $R.Status$ contient quant à lui l'ensemble des éléments structurels qui répondent c'est-à-dire l'ensemble des éléments sources de $R.Status_s$ ainsi que les éléments structurels appartenant aux ensembles décrivant les portées de ces éléments sources.

L'ensemble des éléments structurels vérifiant la contrainte sur le type structurel est donnée par $R_{type}$. Les éléments du document doivent être du type structurel titreChapitre pour appartenir à cet ensemble:

$$R_{type} = \{o \mid o \in OS \wedge type_{str}(o) = titreChapitre\}$$

Soit $R_{type} = \{o_6, o_{10}\}$ et $R.Status_s = \{o_3\}$ et $R.Status = R.Status_s \cup \{o_6, o_7, o_8\}$. Les éléments structurels qui répondent à la requête sont donc ceux qui vérifient les deux contraintes. Nous utilisons donc l'intersection entre les ensembles provenant des deux contraintes. L'ensemble $R_1$ est constitué des éléments sources de l'attribut Status dont la valeur de cet attribut est "draft" et ces éléments sont de type structurel titreChapitre. Cet ensemble est vide. L'ensemble $R_2$ réalise l'intersection entre les éléments non sources de l'attribut Status dont la valeur de cet attribut est "draft" avec les éléments de type structurel titreChapitre. Nous retrouvons ainsi l'élément structurel $o_6$ qui répond à la requête 4.

$$R_1 = R_{type} \cap R.Status_s = \emptyset$$
$$R_2 = (R_{type} \cap R.Status) \Leftrightarrow R1 = \{o_6\}$$

Connaissances pour résoudre la requête 4: (MDe) + (DSe)

**Requête 5** *Donnez les éléments structurels qui ont uniquement été écrits par A et B.*

La requête 5 ne précise pas quel type d'élément structurel est recherché mais elle spécifie que les auteurs sont *uniquement* A et B. Il s'agit donc d'une égalité stricte.

Si nous comparons le modèle de document initial et le modèle de document étendu, nous nous rendons compte que pour l'attribut Auteur l'introduction des portées a singulièrement augmenté sa couverture.

Nous pouvons comparer les résultats obtenus par la résolution de cette requête avec le modèle de document initial avec ceux obtenus en utilisant le modèle de document étendu. Il n'existe pas de réponse à cette requête dans le document structuré initial

Dans le modèle de document étendu, tout type d'élément admettant un attribut Auteur via la relation $\mathfrak{R}'_{ST}$ pourra alors être retrouvé. Le parcours de l'ensemble $OS_{Auteur}$ nous

permet de déterminer les éléments sources qui répondent exactement à la requête, c'est-à-dire l'ensemble $R.Auteur_s$. Dans ce cas précis, nous retrouvons l'élément $o_3$.

$$R.Auteur_s = \{o_3\} \quad et \quad R.Auteur = R.Auteur_s$$

Nous constatons ici que l'ensemble $R.Auteur$ est égal à l'ensemble $R.Auteur_s$. Ceci vient du fait que l'ensemble $EP^{o_3}_{Auteur,comp}$ décrivant la portée de l'attribut Auteur à partir de l'élément structurel $o_3$ est vide. Si cet ensemble n'était pas vide, chacun des éléments structurels de cet ensemble aurait aussi répondu exactement à la requête puisque la fonction de propagation de la portée est l'identité. Donc dans un tel ensemble, les attributs Auteur prennent la même valeur que l'attribut de l'élément source $o_3$.

Connaissances pour résoudre la requête 5 : (MDe) + (DSe)

Nous approfondissons les cas de figure possibles dans les exemples qui suivent (requête 6) en introduisant des réponses partielles.

**Requête 6** *Donnez les éléments structurels qui sont à propos des bases de données et de l'algèbre.*

La requête 6 est similaire à la requête 5 du fait qu'elle ne précise pas quel type d'élément structurel est recherché. Cependant cette requête fait intervenir l'attribut Contenu pour lequel les modes de correspondance peuvent être plus complexes. Le domaine de cet attribut est un ensemble de mots-clés, la correspondance exacte s'établit avec l'égalité ensembliste et une correspondance partielle peut reposer sur l'inclusion ensembliste.

Nous parcourons l'ensemble $OS_{Contenu}$, qui contient l'ensemble des éléments sources du document pour l'attribut Contenu, pour retrouver les éléments qui répondent exactement à la requête (si ces éléments existent). Cette recherche nous permet de construire l'ensemble $R.Contenu_s$ qui va être composé du seul élément $o_{12}$ dont l'attribut Contenu correspond exactement à ce qui est recherché : $o_{12}.Contenu = \{\text{BD, Algèbre}\}$. Nous cherchons maintenant à élargir ces réponses avec l'aide des ensembles décrivant les portées, les $EP^{o_i}_{Contenu,comp}$, c'est-à-dire en construisant $R.Contenu$. Ces réponses seront elles aussi exactes puisque la fonction de propagation est l'identité. L'ensemble décrivant la portée de l'élément $o_{12}$ est non vide : $EP^{o_{12}}_{Contenu,comp} = \{o_4\}$. Ceci signifie que l'élément $o_4$ répond lui aussi exactement à la requête 6.

Le résultat que nous pouvons fournir avec les correspondances exactes se compose donc de deux ensembles : $R.Contenu_s$ et $R.Contenu$ qui fournissent deux éléments structurels, $o_{12}$ et $o_4$.

Connaissances pour résoudre la requête 6 : (MDe) + (DSe)

L'attribut Contenu peut utiliser des langages complexes comparable à des langages de représentation de connaissances tels que les graphes conceptuels ou des logiques descriptives. Lors de l'utilisation de ces langages, il est possible d'obtenir des correspondances partielles

c'est-à-dire retrouver des éléments qui répondent correctement à la requête mais de manière moins précise.

Revenons à notre exemple pour montrer ce type de correspondance. Nous avons jusqu'alors uniquement utilisé l'ensemble des éléments sources du document et les ensembles décrivant les portées de ces éléments.

Pour étendre le résultat, nous considérons des correspondances partielles, en remplaçant l'égalité ensembliste par l'inclusion ensembliste. La portée définie pour l'attribut Contenu suit la relation de composition en remontant vers l'élément racine du document et en utilisant l'union comme fonction de combinaison qui correspond donc à l'agrégation des valeurs. La relation interne à l'ensemble $OS_{Contenu}$ exprime donc que pour tout élément élément $o_i$ qui est un prédécesseur de $o$ ($o_i$ et $o$ appartenant à cet ensemble) nous avons $o.Contenu \subseteq o_i.Contenu$.

L'élément $o_4$ est l'unique élément répondant exactement à la requête 6, nous pouvons donc considérer que la relation d'inclusion est stricte. Et par conséquent, tout élément structurel prédécesseur de l'élément structurel $o_4$ va répondre partiellement à la requête 6 comme l'indique la relation interne à $OS_{Contenu}$ sur les valeurs d'attribut.

Le résultat se décompose donc en deux parties, d'une part ce que nous nommons les *réponses exactes* représentées par les ensembles $R.Contenu_s$ et $R.Contenu$ et d'autre part ce que nous appelons les *réponses partielles* représentées par les ensembles $R.Contenu_s^-$ et $R.Contenu^-$ :

**les réponses exactes** : les éléments structurels sources $o_s$ de l'ensemble $OS_{Contenu}$ tels que $o_s.Contenu = q.Contenu$ et les éléments des ensembles décrivant les portées depuis les éléments sources $o_s$.

$$R.Contenu_s = \{o_s \mid o_s \in OS_{Contenu} \wedge o_s.Contenu = q.Contenu\}$$
*et*
$$\begin{aligned} R.Contenu &= R.Contenu_s \cup \{o \mid o \in EP^{o_s}_{Contenu,comp} \wedge o_s \in R.Contenu_s\} \\ &= R.Contenu_s \cup \bigcup_{o_s \in R.Contenu_s} (EP^{o_s}_{Contenu,comp}) \end{aligned}$$

Soit : $R.Contenu_s = \{o_{12}\}$ et $R.Contenu = \{o_{12}, o_4\}$.

**les réponses partielles** : il s'agit des éléments structurels sources de l'ensemble $OS_{Contenu}$ qui répondent partiellement, notés $o_s^-$ et tels que $o_s^-.Contenu \rightarrow_{Contenu} q.Contenu$, où $q.Contenu$ est la valeur recherchée et $\rightarrow_{Contenu}$ est une mesure de correspondance partielle sur les attributs de contenu.

L'ensemble noté $R.Contenu_s^-$ contient uniquement des éléments structurels sources alors que l'ensemble noté $R.Contenu^-$ contient l'ensemble des éléments qui répondent selon la relation d'inclusion.

$$R.Contenu_s^- = \{o_s^- \mid o_s^- \in OS_{Contenu} \wedge o_s^-.Contenu \rightarrow_{Contenu} q.Contenu\}$$
*et*
$$\begin{aligned} R.Contenu^- &= R.Contenu_s^- \cup \{o \mid o \in EP^{o_s^-}_{Contenu,comp} \wedge o_s^- \in R.Contenu_s^-\} \\ &= R.Contenu_s^- \cup \bigcup_{o_s^- \in R.Contenu_s^-} (EP^{o_s^-}_{Contenu,comp}) \end{aligned}$$

Soit : $R.Contenu_s^- = \{o_1\}$ et $R.Contenu^- = \emptyset$.

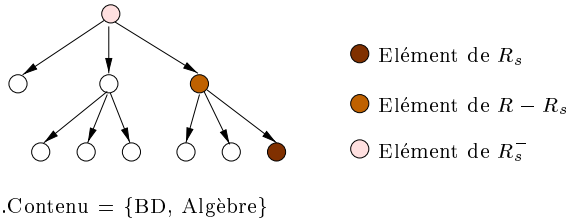



**Figure 7.8.** Résultat de la requête 6

Les éléments structurels des ensembles $R.Contenu_s^-$ et $R.Contenu^-$ correspondent à une extension du résultat de la requête par l'usage des portées, et donc par l'usage des caractéristiques structurelles de l'attribut Contenu. La définition de ces ensembles va dépendre des caractéristiques de la fonction de combinaison de la portée de l'attribut et de l'opérateur de recherche (égalité, inclusion, projection, etc).

Connaissances pour résoudre la requête 6 : (MDe) + (DSe)

**Requête 7** *Donnez les images traitant des SRI et de la logique.*

La requête 7 requiert une connaissance sur le média, ici l'image, et sur l'attribut Contenu. La connaissance sur le média tout comme celle sur l'attribut Contenu nous est donnée par le modèle de document.

Le modèle précise que les éléments de type image sont des éléments feuilles dans l'arborescence structurelle et les attributs Contenu apparaissent au niveau de ces éléments feuilles. Nous n'avons donc pas l'utilité du modèle de document étendu, ni celle du document structuré étendu pour répondre à cette requête.

$$R_{image} = \{o \mid o \in OS \wedge media(o) = image\}$$

Nous avons limité la profondeur de notre exemple de document structuré. Il ne nous est donc pas possible de donner la réponse sur cet exemple cependant il est intéressant de noter que puisque la requête 7 spécifie un média qui est obligatoirement une feuille de structure et sur lequel l'attribut Contenu est toujours défini, l'utilisation de notre modèle étendu n'est pas utile pour ce type de requête.

Toutefois, il est possible d'envisager des extensions pour cette requête. nous pouvons par exemple élargir la recherche afin de retrouver les images qui sont des composants d'éléments structurels traitant des SRI et de la logique. Nous traitons ceci dans la requête 8.

Connaissances pour résoudre la requête 7 : (MD) + (DS)

**Requête 8** *Donnez les éléments composés d'au moins une image et traitant des systèmes de recherche d'informations et de la logique.*

Par rapport à la requête précédente, requête 7, la requête 8 est modifiée du fait qu'il ne s'agit plus de retrouver des éléments provenant d'un média particulier, en l'occurrence des images, mais des éléments dans lesquels apparaît ce média.

Nous distinguons donc les deux parties de la requête; celle sur le média image et celle sur l'attribut Contenu.

La première partie se résout en utilisant le modèle de document et le document structuré. Les éléments structurels du document contenant une image sont donc les éléments $o_7$ et $o_{12}$ ainsi que tous leurs ascendants structurels. Nous notons $R_{image}$ l'ensemble des éléments structurels qui répondent à cette première partie de la requête 8.

$$\begin{aligned} R_{image} &= \{o \mid o \in OS \wedge media(o) = image\} \\ et & \\ R^{-}_{image} &= \{o \mid o \in OS \wedge image \subset media(o)\} \end{aligned}$$

Sur notre exemple de document structuré, cela nous donne :

$$\begin{aligned} R_{image} &= \emptyset \\ R^{-}_{image} &= \{o_7, o_{12}\} \cup Prev_{comp}(o_7) \cup Prev_{comp}(o_{12}) \\ &= \{o_7, o_{12}, o_3, o_4, o_1\} \end{aligned}$$

Nous résolvons le traitement de l'attribut Contenu de manière similaire à ce que nous avons présenté pour la requête 6 en utilisant bien entendu le modèle de document étendu ainsi que le document structuré étendu conforme à ce modèle. Le résultat de cette requête prend la forme suivante : un ensemble d'éléments sources correspondant exactement, $R.Contenu_s$, et un ensemble d'éléments sources correspondant avec moins de précision, $R.Contenu^{-}_{s}$ :

$$\begin{aligned} R.Contenu_s = \{o_3\} \quad et \quad R.Contenu = R.Contenu_s \\ R.Contenu^{-}_{s} = \{o_1\} \quad et \quad R.Contenu^{-} = R.Contenu^{-}_{s} \end{aligned}$$

Afin de résoudre la requête et sachant que les éléments structurels qui répondent doivent correspondre simultanément aux deux parties de la requête, nous établissons des niveaux de réponses selon les ensembles que nous avons construits. Nous considérons ainsi que les meilleures réponses sont données par les éléments structurels qui appartiennent à la fois à l'ensemble $R_s$ et à l'ensemble $R_{image}$, nous notons cet ensemble $R_1$. Nous donnons le descriptif

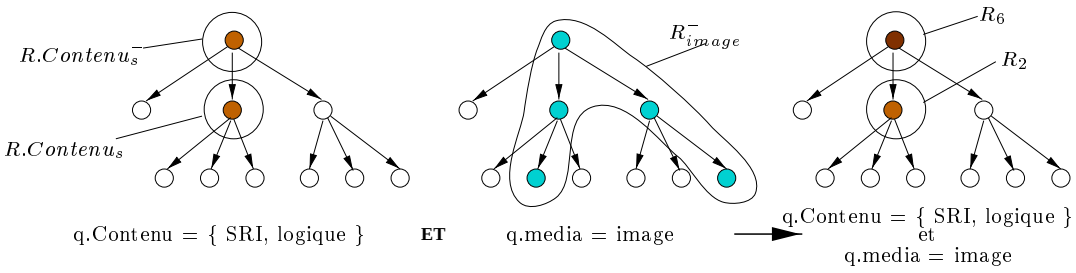



**Figure 7.9.** Résultat de la requête 8

de chacun des ensembles par la suite :

$$
\begin{aligned}
R_1 &= R.Contenu_s \cap R_{image} = \emptyset \\
R_2 &= R.Contenu_s \cap R_{image}^- = \{o_3\} \\
R_3 &= (R.Contenu \cap R_{image}) \Leftrightarrow R_1 = \emptyset \\
R_4 &= (R.Contenu \cap R_{image}^-) \Leftrightarrow R_2 = \emptyset \\
R_5 &= R.Contenu_s^- \cap R_{image} = \emptyset \\
R_6 &= R.Contenu_s^- \cap R_{image}^- = \{o_1\} \\
R_7 &= (R.Contenu^- \cap R_{image}) \Leftrightarrow R_5 = \emptyset \\
R_8 &= (R.Contenu^- \cap R_{image}^-) \Leftrightarrow R_6 = \emptyset
\end{aligned}
$$

- $\mathbf{R_1}$ : ensemble des éléments sources correspondant exactement pour le critère sur l'attribut Contenu et qui sont des images.
- $\mathbf{R_2}$ : ensemble des éléments sources correspondant exactement pour le critère sur l'attribut Contenu et qui contiennent une image.
- $\mathbf{R_3}$ : ensemble des éléments non sources correspondant exactement pour le critère sur l'attribut Contenu et qui sont des images.
- $\mathbf{R_4}$ : ensemble des éléments non sources correspondant exactement pour le critère sur l'attribut Contenu et qui contiennent une image.
- $\mathbf{R_5}$ : ensemble des éléments sources correspondant partiellement pour le critère sur l'attribut Contenu et qui sont des images.
- $\mathbf{R_6}$ : ensemble des éléments sources correspondant partiellement pour le critère sur l'attribut Contenu et qui contiennent une image.
- $\mathbf{R_7}$ : ensemble des éléments non sources correspondant partiellement pour le critère sur l'attribut Contenu et qui sont des images.
- $\mathbf{R_8}$ : ensemble des éléments non sources correspondant partiellement pour le critère sur l'attribut Contenu et qui contiennent une image.

Nous montrons les ensembles non vides $R_2$ et $R_6$ sur la figure 7.9, ainsi que les ensembles $R.Contenu_s$, $R.Contenu_s^-$ et $R_{image}^-$. Comme nous le voyons ici, il est assez facile d'obtenir

de nombreux niveaux de réponses en faisant intervenir les caractéristiques structurelles des éléments en introduisant les propriétés des portées des attributs.

Connaissances pour résoudre la requête 8 : (MDe) + (DSe)

**Requête 9** *Donnez les éléments structurels à l'état de draft qui traitent de la logique.*

Nous considérons ici la requête 9 dans laquelle deux caractéristiques sont requises : la première porte sur l'attribut Status et la second sur l'attribut Contenu. On se rend compte ici que si nous considérons uniquement le modèle de document initial et le document structuré qui se conforme à celui-ci, nous ne pouvons retrouver aucun élément structurel qui réponde à cette requête; dans le modèle de document initial l'attribut Status est uniquement rattaché à des éléments de type chapitre alors que l'attribut Contenu est rattaché aux types d'éléments feuilles, donc pas aux éléments de type chapitre. Nous devons donc explorer le modèle de document étendu ainsi que le document structuré qui s'y conforme.

Le modèle de document étendu nous autorise à penser que des éléments structurels peuvent répondre à cette requête puisque dans ce modèle étendu l'intersection est non vide entre les types compatibles pour les deux attributs.

Les deux attributs requis ont des portées définies selon la relation de composition mais ces portées sont aussi définies selon deux catégories différentes : *succ* pour Status et *pred* pour Contenu.

Etablissons tout d'abord les ensembles d'éléments structurels répondant aux attributs particuliers avant de voir comment peut se dérouler la fusion entre ceux-ci :

$$R.Status_s = \{o_3\} \quad et \quad R.Status = R.Status_s \cup \{o_6, o_7, o_8\}$$

$$R.Contenu_s = \{o_8\} \quad et \quad R.Contenu = R.Contenu_s,$$
$$R.Contenu_s^- = \{o_3, o_1\} \quad et \quad R.Contenu^- = R.Contenu_s^-$$

Considérons tout d'abord l'ensemble des éléments structurels répondant exactement à ce que nous cherchons, soit l'ensemble $R_1$, l'intersection des ensembles composés des éléments sources qui répondent exactement : $R_1 = R.Status_s \cap R.Contenu_s = \emptyset$.

Cet ensemble est vide, nous allons donc maintenant considérer deux ensembles : l'intersection de $R.Status_s$ avec $R.Contenu$ et l'intersection de $R.Status$ avec $R.Contenu_s$. Pour l'un comme pour l'autre des attributs, la fonction de propagation est l'identité, et donc les valeurs des attributs de ces éléments sont identiques à celles attachées aux attributs des éléments sources. Les éléments de ces deux ensembles, que nous notons $R_2$ et $R_2'$, doivent donc être considérés d'égale importance dans les résultats.

$$R_2 = R.Status_s \cap R.Contenu = \emptyset \quad et \quad R_2' = R.Status \cap R.Contenu_s = \{o_8\}$$

L'élément structurel $o_8$ est alors retrouvé comme un élément qui répond exactement à la requête 9 même s'il ne s'agit pas d'un élément qui est source d'information dans le document structuré étendu. Cet élément correspond à une section du document. Nous pouvons noter

que l'introduction des portées permet encre une fois de retrouver un élément qui n'aurait pu être retrouvé avec le modèle de document initial.

Nous pouvons étendre les résultats en considérant maintenant les ensembles qui ne répondent pas exactement à la requête, soit dans ce cas les ensembles $R.Contenu_s^-$ et $R.Contenu^-$.

$$R_3 = R.Status_s \cap R.Contenu_s^- = \{o_3\}$$
et
$$R_4 = R.Status \cap R.Contenu_s^- \Leftrightarrow R_3 = \emptyset \text{ et } R'_4 = R.Status_s \cap R.Contenu^- \Leftrightarrow R_3 = \emptyset$$

L'ensemble $R_3$ fournit un nouvel élément structurel qui répond à la requête, $o_3$ correspond au chapitre qui contient la section précédemment retrouvée dans l'ensemble $R'_2$.

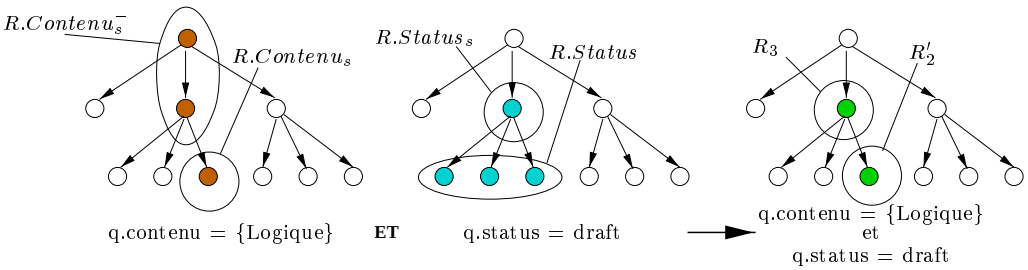



**Figure 7.10.** Résultat de la requête 9

Connaissances pour résoudre la requête 9 : (MDe) + (DSe)

**Requête 10** *Donnez les éléments structurels qui traitent des BD et des SRI et qui ont été écrits par A et B.*

Pour la requête 10, là encore, le modèle de document initial ne fournit pas de réponse. Avec le modèle de document étendu et le document structuré qui s'y conforme, les ensembles que nous construisons pour répondre sont alors les suivants :

$$R.contenu_s = \emptyset \quad et \quad R.contenu = R.contenu_s = \emptyset$$
$$R.contenu_s^- = \{o_1\} \quad et \quad R.contenu_s = R.contenu_s^-$$

et

$$R.Auteur_s = \{o_3\} \quad et \quad R.Auteur = R.Auteur_s = \{o_3\}$$
$$R.Auteur_s^- = \{o_1\} \quad et \quad R.Auteur^- = R.Auteur_s^- = \{o_1\}$$

Nous considérons tout d'abord les réponses exactes, et cela sans succès :

$$R_1 = R.contenu_s \cap R.Auteur_s = \emptyset$$
$$R_2 = R.contenu_s \cap R.Auteur = \emptyset \quad et \quad R'_2 = R.contenu \cap R.Auteur_s = \emptyset$$

Nous vérifions maintenant les réponses partielles :

$$R_3 = R.contenu_s \cap R.Auteur_s^- = \emptyset \quad et \quad R'_3 = R.contenu_s^- \cap R.Auteur_s = \emptyset$$
$$R_4 = R.contenu_s \cap R.Auteur^- = \emptyset \quad et \quad R'_4 = R.contenu^- \cap R.Auteur_s = \emptyset$$
$$R_5 = R.contenu_s^- \cap R.Auteur_s^- = \{o_1\}$$

Finalement nous retrouvons l'élément structurel $o_1$ qui répond partiellement à chacun des deux attributs de la requête. Dans la figure 7.11, nous mettons en évidence cet élément ainsi que les éléments qui répondent à l'un ou à l'autre des attributs.

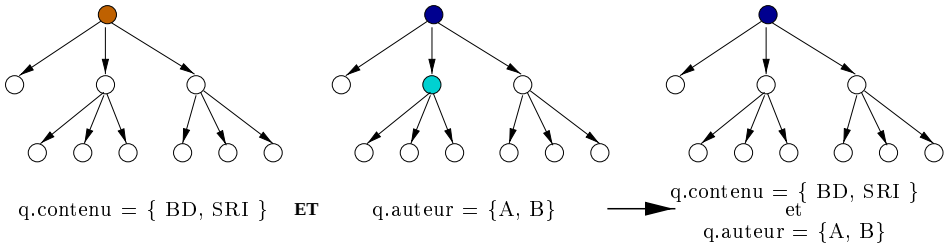



**Figure 7.11.** Résultat de la requête 10

Connaissances pour résoudre la requête 10 : (MDe) + (DSe)

## 7.4 Classification des requêtes

Après avoir vu un certain nombre d'exemples de requêtes, nous cherchons à donner une classification afin de déterminer pour une requête donnée les éléments de notre modèle mis en jeu ainsi que les stratégies utilisées.

### 7.4.1 Classification générale

**Les requêtes purement structurelles** : ces requêtes intègrent uniquement des critères structurels, c'est-à-dire des sélections sur les types d'éléments de structure, des contraintes de composition ou de séquence sur les éléments structurels ou tout autre type de critères relatifs à la structure.

Il est nécessaire, pour le traitement des requêtes purement structurelles, de mettre en jeu d'une part le modèle de document et ensuite les instances du modèle représentant les documents structurés. Le modèle permet de vérifier que la requête est conforme à un type de document au moins. Puis, en confrontant les critères de la requête aux documents, nous pouvons restituer le ou les documents répondant correctement en utilisant les foncions d'accès décrites à partir des relations de structure du modèle.

**Les requêtes de contenu mono-attribut** : ces requêtes intègrent des critères sur un unique attribut. Le traitement de la requête va dépendre de la nature de l'attribut, c'est-à-dire de son comportement : *statique* ou *dynamique*.

**Attribut statique** : ce type d'attributs a été défini sans portée et la résolution des requêtes sur ces attributs ne requiert pas de traitement spécifique. Puisqu'il n'y a pas de portée, les seuls éléments qui admettent des attributs statiques sont des éléments définis dans le modèle de document initial. Nous considérons donc uniquement l'ensemble $OS_\alpha$, où $\alpha$ est un attribut statique et où les éléments $o_i$ de cet ensemble admettent l'attribut $\alpha$ conformément au modèle de document initial.

Le modèle de document étendu n'est donc pas utile pour les attributs statiques.

**Attribut dynamique** : un attribut dynamique est un attribut auquel nous avons adjoint une portée selon un relation de structure (composition ou séquence). Ceci signifie que cet attribut qui a été défini initialement sur un ensemble d'éléments structurels conformément au modèle de document initial est dorénavant défini et valué sur un nouvel ensemble (étendu) d'éléments structurels conformément au modèle de document étendu. Nous considérons d'une part les éléments sources, c'est-à-dire les éléments de l'ensemble $OS_\alpha$ et les éléments des portées de ces éléments sources, les éléments des $EP^o_{\alpha,rel}$.

Nous avons énoncé comme principe de base de privilégier les éléments sources. Nous distinguons donc les réponses provenant de l'ensemble $OS_\alpha$ des autres. Nous construisons donc deux ensembles distincts et disjoints : $R.\alpha$ et $R.\alpha_s$ où $R.\alpha_s$ contient les éléments de $OS_\alpha$ qui répondent aux critères de la requête et $R.\alpha$ contient les éléments des portées qui répondent aussi à la requête :

$$R.\alpha_s \subseteq OS_\alpha \text{ et } R.\alpha \subseteq \bigcup_{o \in OS_\alpha} EP^o_{\alpha,rel}$$

La portée de ce type d'attribut requiert un traitement particulier qui dépend de la relation entre l'opérateur de recherche de la requête et la relation liant les éléments de $OS_\alpha$.

**Les requêtes de contenu multi-attributs** : ces requêtes proposent de combiner plusieurs critères sur des attributs différents. Par exemple *un article de journal parlant de privatisation de France Télécom et paru dans Libération.* Dans cette requête, nous identifions deux attributs : le propos, *la privatisation de France Télécom* et le nom de la parution, *Libération.* Nous proposons en fait de combiner plusieurs attributs par l'intermédiaire des trois opérateurs classiques des modèles booléens, à savoir le ET, le OU et le SAUF.

**Les requêtes mixtes** : ces requêtes comportent à la fois des spécifications structurelles et des spécifications sur les attributs. C'est dans ces requêtes que l'introduction des portées des attributs prend tout son sens puisqu'elle a permis d'étendre la couverture du document. Nous sommes donc en mesure de retourner des éléments de structure que le modèle initial ne pouvait pas atteindre. Les requêtes 4, 5, 6, 7, 8, 9 et 10 sont des exemples de telles requêtes mixtes.

Les critères de recherche structurels et sur les attributs sont combinés à l'aide des opérateurs booléens, ET, OU et SAUF.

## 7.5 Formalisation de la correspondance

Dans les définitions qui suivent nous nous limitons volontairement à des fonctions de propagation correspondant à l'identité. Dans le cas où cette fonction ne correspond pas à l'identité nous appliquons un parcours classique des ensembles considérés.

### 7.5.1 Définitions formelles des ensembles résultats

Soit une requête $Q$ portant sur l'attribut $\alpha$ exprimée de la manière suivante : $(x.\alpha\ op_\alpha\ q.\alpha)$ où $x$ est une variable à laquelle nous cherchons à affecter un élément de structure, $op_\alpha$ est un opérateur compatible avec le domaine de définition de l'attribut $\alpha$ et enfin $q.\alpha$ est la valeur requise pour l'attribut $\alpha$ sur l'élément $x$.

Nous construisons à partir de cette requête les ensembles $R.\alpha_s$ et $R.\alpha$ correspondant respectivement à l'ensemble des éléments sources qui répondent et à l'ensemble total des éléments qui répondent pour un document donné.

$$R.\alpha_s = \{o_s \mid o_s \in OS_\alpha \wedge (o_s.\alpha\ op_\alpha\ q.\alpha)\} \tag{7.1}$$

Les éléments de $R.\alpha$ peuvent être jugés d'égale importance vis à vis de ceux de $R.\alpha_s$ toutefois nous privilégions les éléments sources en conservant l'ensemble $R.\alpha_s$.

$$R.\alpha = \bigcup_{o \in R.\alpha_s} (EP^o_{\alpha,rel}) \cup R.\alpha_s \tag{7.2}$$

Nous avons vu au travers des exemples précédents que nous faisions intervenir une notion de *relaxation* en construisant les ensembles $R.\alpha_s^-$ et $R.\alpha^-$, qui correspondent respectivement à l'ensemble des éléments sources qui répondent partiellement et à l'ensemble total des éléments qui répondent partiellement pour un document donné. Ces ensembles interviennent lorsque l'opérateur de recherche le permet, par exemple lorsque cet opérateur est l'égalité et qu'il existe une relation d'inclusion entre les valeurs de l'attribut.

Ces ensembles $R.\alpha_s^-$ et $R.\alpha^-$ ont la forme suivante:

$$R.\alpha_s^- = \{o_s^- \mid o_s^- \in OS_\alpha \wedge o_s^-.\alpha \rightarrow_\alpha q.\alpha\} \tag{7.3}$$

où la notation $\rightarrow_\alpha$ exprime la correspondance partielle.

$$R.\alpha^- = \bigcup_{o \in R.\alpha_s^-} (EP^o_{\alpha,rel}) \cup R.\alpha_s^- \tag{7.4}$$

Nous donnons dans le tableau qui suit deux cas de correspondance partielle autorisant la construction des ensembles $R.\alpha_s^-$ et $R.\alpha^-$. Le premier exemple opère sur les ensembles.

Lorsque l'égalité est requise, il est alors possible de fournir en tant que réponse les éléments dont la valeur de l'attribut inclut la valeur requise. Le second exemple est fourni par l'utilisation du formalisme des graphes conceptuels. Dans ce formalisme, il existe un opérateur de correspondance partielle, noté $\leq'$, permettant de retrouver des graphes qui sont "partiellement" corrects pour une requête donnée [Che92].

| Domaine de $\alpha$ | Relation de $OS_\alpha : C_\alpha$ | Opérateur | Correspondance partielle |
|---|---|---|---|
| ensembles | $\subset$ | $=$ | $\subset$ |
| graphes conceptuels | $\leq$ | $\leq$ | $\leq'$ |

Nous allons pour notre part nous limiter à des correspondances partielles basées sur la relation $C_\alpha$, comme l'exemple des ensembles le montre dans le tableau.

Nous allons maintenant expliquer comment les ensembles sont construits à partir de la relation générée par l'application des portées.

### 7.5.2 Parcours de l'ensemble des éléments sources

Pour construire un ensemble résultat, noté $R.\alpha$ ou $R.\alpha_s$, il nous faut parcourir dans un premier temps l'ensemble des éléments sources de chaque document. Nous présentons comment nous utilisons la relation sur les valeurs des attributs qui a été générée par l'application des portées. Ce parcours est valable pour les attributs admettant une portée. Dans les autres cas, nous proposons un parcours séquentiel.

Nous rappelons tout d'abord le sens de la relation $C_\alpha$ générée sur les valeurs de l'attribut $\alpha$ des éléments sources $OS_\alpha$. Puis nous montrons comment nous l'utilisons pour parcourir l'ensemble des éléments sources et extraire les éléments qui répondent.

#### a) Relation sur l'ensemble des éléments sources

La relation interne à l'ensemble $OS_\alpha$ qui régit les dépendances entre les valeurs des attributs $\alpha$ des éléments sources du document se doit d'être précisée vis-à-vis des opérateurs de recherche susceptibles d'être rencontrés selon les différents domaines de définition des attributs et vis-à-vis de la relation interne à chaque ensemble décrivant une portée. Il nous faut donc établir ces relations de conformité entre les opérateurs de recherche, la relation de $OS_\alpha$, notée $C_\alpha$ et la relation des $\mathcal{P}^o_{\alpha,rel}$ où $o$ appartient à $OS_\alpha$ et qui est notée $R_\alpha$.

Dans le tableau qui suit, nous donnons des domaines d'attributs avec les relations de l'ensemble des éléments sources $OS_\alpha$ et les relations compatibles des portées. Les fonctions de propagation et les fonctions de combinaison des portées doivent donc respecter les compatibilités que nous donnons dans le tableau qui suit (figure 7.12).

Comme nous l'avons vu lors de la définition de la fonction de propagation, la compatibilité des relations $C_\alpha$ et $R_\alpha$ s'établit ainsi : si $R_\alpha(a,b)$ et $C_\alpha(b,c)$ alors $C_\alpha(a,c)$. Par ailleurs, de la compatibilité entre les opérateurs de recherche et ces deux relations va dépendre l'utilisation que l'on pourra faire de $C_\alpha$ et $R_\alpha$.

| Domaine | Relation de $OS_\alpha : C_\alpha$ | $R_\alpha$ [a] |
|---|---|---|
| entiers | $<$ | $<$, $\leq$ ou $=$ |
| | $\leq$ | $<$, $\leq$ ou $=$ |
| | $>$ | $>$, $\geq$ ou $=$ |
| | $\geq$ | $>$, $\geq$ ou $=$ |
| | $=$ | $=$ |
| ensembles | $\subset$ | $\subset$, $\subseteq$ ou $=$ |
| | $\subseteq$ | $\subset$, $\subseteq$ ou $=$ |
| | $\supset$ | $\supset$, $\supseteq$ ou $=$ |
| | $\supseteq$ | $\supset$, $\supseteq$ ou $=$ |
| | $=$ | $=$ |
| graphes conceptuels | $\leq$ [b] | $\leq$ ou copy [c] |
| | $\geq$ | $\geq$ ou copy |

[a] Nous rappelons que nous nous limitons ici à l'égalité
[b] $g_1 \leq g_2$ signifie que le graphe $g_1$ est une spécialisation du graphe $g_2$
[c] opération de copie d'un graphe qui correspond à l'égalité

**Figure 7.12.** Compatibilités des relations

La relation $C_\alpha$ définit un ordre non total sur l'ensemble $OS_\alpha$. Nous pouvons donc décrire deux ensembles sur $OS_\alpha$: celui des éléments minimum et celui des éléments maximum que nous notons respectivement $OS_{\alpha\_max}$ et $OS_{\alpha\_min}$:

$$OS_{\alpha\_max} = \{o_i \mid o_i \in OS_\alpha \text{ et } \nexists\, o \text{ tel que } C_\alpha(o_i.\alpha, o.\alpha)\}$$

$$OS_{\alpha\_min} = \{o_i \mid o_i \in OS_\alpha \text{ et } \nexists\, o \text{ tel que } C_\alpha(o.\alpha, o_i.\alpha)\}$$

### b) Parcours de recherche

Considérons que nous avons une requête dans laquelle nous recherchons les éléments dont l'attribut $\alpha$ a pour valeur $x$. Nous distinguons différents stratégies qui dépendent de la relation entre l'opérateur de recherche et la relation $C_\alpha$: soit ils sont identiques, soit ils sont opposés, soit l'opérateur de recherche est l'égalité alors que la relation n'est pas l'égalité. Nous présentons ces trois cas.

*L'opérateur de recherche correspond à la relation $C_\alpha$*

Nous initialisons notre recherche par le parcours de l'ensemble des éléments maximum.

1. Soit un élément $o$ de $OS_{\alpha\_max}$ et $C_\alpha(x, o.\alpha)$ n'est pas vérifiée. Alors l'élément $o$ ne répond pas à la requête ni les éléments qui lui sont inférieurs selon la relation $C_\alpha$. Il est inutile de poursuivre.

2. Soit un élément $o$ de $OS_{\alpha\_max}$ et $C_\alpha(x, o.\alpha)$ est vérifiée. Alors l'élément $o$ répond à la

requête. Il faut chercher parmi les éléments qui sont inférieurs à $o$ selon la relation $C_\alpha$ s'il y'en a d'autres qui répondent.

(a) Soit un élément $o_i$ tel que $C_\alpha(o_i.\alpha, o.\alpha)$ et $C_\alpha(x, o_i.\alpha)$ est vérifiée : il est encore possible de suivre la relation $C_\alpha$ pour chercher des éléments qui répondent plus précisément.

(b) Soit un élément $o_i$ tel que $C_\alpha(o_i.\alpha, o.\alpha)$ et $C_\alpha(x, o_i.\alpha)$ n'est pas vérifiée : il est inutile de chercher parmi les éléments qui sont inférieurs à $o_i$ selon la relation $C_\alpha$.

Tout les éléments $o$ qui répondent, c'est-à-dire qui vérifient la relation $C_\alpha(x, o.\alpha)$, appartiennent à l'ensemble des réponses. L'ordre donné par la relation $C_\alpha$ permet de les classer. il s'agit ici des éléments qui répondent parmi les éléments sources.

*L'opérateur de recherche correspond à l'inverse de la relation $C_\alpha$*

Nous appliquons un parcours similaire en partant de l'ensemble des éléments minimum et en considérant la relation inverse.

1. Soit un élément $o$ de $OS_{\alpha\_min}$ et $C_\alpha(o.\alpha, x)$ n'est par vérifiée. Alors l'élément $o$ ne répond pas à la requête ni les éléments qui lui sont supérieurs selon la relation $C_\alpha$. Il est inutile de poursuivre.

2. Soit un élément $o$ de $OS_{\alpha\_max}$ et $C_\alpha(o.\alpha, x)$ est vérifiée. Alors l'élément $o$ répond à la requête. Il faut chercher parmi les éléments qui sont supérieurs à $o$ selon la relation $C_\alpha$ s'il y'en a d'autres qui répondent.

   (a) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $C_\alpha(o_i.\alpha, x)$ est vérifiée : il est encore possible de suivre la relation $C_\alpha$ pour chercher des éléments qui répondent plus précisément.

   (b) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $C_\alpha(o_i.\alpha, x)$ n'est pas vérifiée : il est inutile de chercher parmi les éléments qui sont inférieurs à $o_i$ selon la relation $C_\alpha$.

*L'opérateur de recherche est l'égalité*

Nous distinguons deux cas dépendant des domaines et des possibilités d'étendre ou non l'égalité : le premier cas concerne généralement les valeurs mono-valuées pour lesquelles il n'est pas possible d'étendre l'égalité et le second cas concerne les valeurs multi-valuées (ensemble de mots-clés par exemple) ou qui reposent sur des formalismes de représentation de connaissance fournissant un opérateur de correspondance partielle (par exemple le formalisme des graphes conceptuels). Dans ce second cas, nous pouvons proposer des éléments répondant partiellement comme nous l'avons montré dans l'exemple de la requête

*Pas de réponses partielles*

Les éléments que nous allons sélectionner dans l'algorithme suivant vont former les éléments de l'ensemble $R.\alpha_s$.

1. Soit un élément $o$ de $OS_{\alpha\_max}$ et $x_\alpha = o.\alpha$ : l'élément $o$ répond exactement. Il est possible de trouver d'autres éléments parmi ses inférieurs selon la relation $C_\alpha$ qui répondent aussi

exactement.

   (a) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha \neq o.\alpha$ : Il est inutile de poursuivre.

   (b) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha = o.\alpha$ : Il est encore possible de trouver d'autres éléments parmi ses inférieurs selon la relation $C_\alpha$ qui répondent aussi exactement.

2. Soit un élément $o$ de $OS_{\alpha\_max}$ et $x_\alpha \neq o.\alpha$ : l'élément $o$ ne répond pas. Il est inutile de poursuivre.

*Réponses partielles possibles*

Nous sélectionnons deux types d'éléments: ceux qui répondent exactement et ceux qui répondent partiellement. Les premiers forment l'ensemble $R.\alpha_s$ alors que les seconds forment l'ensemble $R.\alpha_s^-$. Une réponse partielle se caractérise ici par la vérification de la relation $C_\alpha(x_\alpha, o.\alpha)$, où $x_\alpha$ est la valeur recherchée pour l'attribut $\alpha$.

1. Soit un élément $o$ de $OS_{\alpha\_max}$ et $x_\alpha = o.\alpha$ : l'élément $o$ répond exactement. Il est possible de trouver d'autres éléments parmi ses inférieurs selon la relation $C_\alpha$ qui répondent aussi exactement.

   (a) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha \neq o.\alpha$ : Il est inutile de poursuivre.

   (b) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha = o.\alpha$ : Il est encore possible de trouver d'autres éléments parmi ses inférieurs selon la relation $C_\alpha$ qui répondent aussi exactement.

2. Soit un élément $o$ de $OS_{\alpha\_max}$ et $x_\alpha \neq o.\alpha$ et $C_\alpha(x_\alpha, o.\alpha)$ n'est par vérifiée. Alors l'élément $o$ ne répond pas à la requête ni les éléments qui lui sont inférieurs selon la relation $C_\alpha$. Il est inutile de poursuivre.

3. Soit un élément $o$ de $OS_{\alpha\_max}$ et $x_\alpha \neq o.\alpha$ et $C_\alpha(x_\alpha, o.\alpha)$ est vérifiée. Alors l'élément $o$ ne répond pas exactement à la requête mais il y répond partiellement. Il faut chercher parmi les éléments qui sont inférieurs à $o$ selon la relation $C_\alpha$ s'il y'en a d'autres qui répondent soit partiellement, soit exactement.

   (a) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha \neq o.\alpha$ et $C_\alpha(o_i.\alpha, x_\alpha)$ est vérifiée : il est encore possible de suivre la relation $C_\alpha$ pour chercher des éléments qui répondent plus précisément.

   (b) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha \neq o.\alpha$ et $C_\alpha(x_\alpha, o.\alpha)$ n'est par vérifiée : il est inutile de chercher parmi les éléments qui sont inférieurs à $o_i$ selon la relation $C_\alpha$.

   (c) Soit un élément $o_i$ tel que $C_\alpha(o.\alpha, o_i.\alpha)$ et $x_\alpha = o.\alpha$ : l'élément $o_i$ répond exactement. On se retrouve dans le cas 1.

### 7.5.3 Interprétation des opérateurs booléens

Nous considérons les trois opérateurs du modèle booléen ET, OU et SAUF pour combiner les critères de recherche dans les requêtes. Les interprétations dans le modèle booléen de recherche d'information de ces opérateurs est assimilée aux opérateurs ensemblistes suivants : intersection, union et différence. Nous allons donc voir l'interprétation que nous donnons à ces opérateurs dans notre modèle de correspondance.

Soit les ensembles suivants construits comme indiqué précédemment pour des requêtes $(q.\alpha\ op_\alpha\ x_\alpha)$ et $(q.\beta\ op_\beta\ x_\beta)$ :

| | |
|---|---|
| Soient $R.\alpha_s$, $R.\alpha$, $R.\alpha_s^-$ et $R.\alpha^-$ | Soient $R.\beta_s$, $R.\beta$, $R.\beta_s^-$ et $R.\beta^-$ |
| tels que $\forall o \in R.\alpha_s$ on a $o.\alpha\ op_\alpha\ x_\alpha$ | tels que $\forall o \in R.\beta_s$ on a $o.\beta\ op_\beta\ x_\beta$ |
| et $\forall o \in R.\alpha$ on a $o.\alpha\ op_\alpha\ x_\alpha$ | et $\forall o \in R.\beta$ on a $o.\beta\ op_\beta\ x_\beta$ |
| et $\forall o \in R.\alpha_s^-$ on a $o.\alpha \rightarrow_\alpha x_\alpha$ | et $\forall o \in R.\beta_s^-$ on a $o.\beta \rightarrow_\beta x_\beta$ |
| et $\forall o \in R.\alpha^-$ on a $o.\alpha \rightarrow_\alpha x_\alpha$ | et $\forall o \in R.\beta^-$ on a $o.\beta \rightarrow_\beta x_\beta$ |

### 7.5.4 L'opérateur ET

Soit la requête $(q.\alpha\ op_\alpha\ x_\alpha)$ ET $(q.\beta\ op_\beta\ x_\beta)$ et les ensembles de réponses respectifs. Nous considérons alors les éléments des intersections suivantes :

1. $R_1 = R.\alpha_s \cap R.\beta_s$ : ensemble des éléments structurels sources de $\alpha$ et de $\beta$ qui répondent aux critères sur les attributs $\alpha$ et $\beta$.
2. $R_2 = (R.\alpha \cap R.\beta_s) \Leftrightarrow R_1$ : ensemble des éléments structurels qui répondent aux critères sur les attributs $\alpha$ et $\beta$, et qui sont éléments sources pour $\beta$.
3. $R'_2 = (R.\alpha_s \cap R.\beta) \Leftrightarrow R_1$ : ensemble des éléments structurels qui répondent aux critères sur les attributs $\alpha$ et $\beta$, et qui sont éléments sources pour $\alpha$.
4. $R_3 = (R.\alpha \cap R.\beta) \Leftrightarrow (R_1 \cup R_2 \cup R'_2)$ : ensemble des éléments structurels qui répondent aux critères sur les attributs $\alpha$ et $\beta$, et qui ne sont pas des éléments sources.
5. $R_4 = R.\alpha_s^- \cap R.\beta_s$ : ensemble des éléments structurels sources de $\alpha$ et de $\beta$ qui répondent partiellement aux critères de $\alpha$ et exactement aux critères de $\beta$.
6. $R'_4 = R.\alpha_s \cap R.\beta_s^-$ : ensemble des éléments structurels sources de $\alpha$ et de $\beta$ qui répondent exactement aux critères de $\alpha$ et partiellement aux critères de $\beta$.
7. $R_5 = R.\alpha^- \cap R.\beta_s$ : ensemble des éléments structurels qui répondent partiellement aux critères de $\alpha$ et exactement aux critères de $\beta$, et qui sont éléments sources pour $\beta$.
8. $R'_5 = R.\alpha_s \cap R.\beta^-$ : ensemble des éléments structurels qui répondent exactement aux critères de $\alpha$ et partiellement aux critères de $\beta$, et qui sont éléments sources pour $\alpha$.

Nous continuons ainsi à construire des ensembles tant que nous n'avons pas trouvé d'éléments qui répondent. Dans l'ensemble $R_6$, nous allons considérer les éléments non sources

qui répondent exactement pour $\alpha$ et les éléments sources qui répondent partiellement pour $\beta$, et réciproquement pour $R'_6$. Dans $R_7$ n'apparaitront que des éléments non sources qui reprennent les critères précédents sur $\alpha$ et $\beta$, et réciproquement pour $R'_7$. Enfin pour $R_8$, nous ne considérons plus que des éléments qui répondent partiellement parmi les éléments sources. Puis nous considérons les éléments non sources dans $R_9$, $R'_9$ et $R_{10}$ selon un modèle similaire à ce que nous avons fait pour $R_2$, $R'_2$et $R_3$

Dans le cas où la correspondance partielle n'est pas possible pour les deux attributs, nous nous limitons aux ensembles $R_1$, $R_2$, $R'_2$ et $R_3$.

### 7.5.5 L'opérateur OU

Soit la requête ($q.\alpha\ op_\alpha\ x_\alpha$) OU ($q.\beta\ op_\beta\ x_\beta$) et les ensembles de réponses respectifs. Nous proposons le même modèle que précédemment en remplaçant l'opérateur d'intersection par l'union ensembliste.

1. $R_1 = R.\alpha_s \cup R.\beta_s$ : ensemble des éléments structurels sources de $\alpha$ et de $\beta$ qui répondent soit aux critères sur l'attribut $\alpha$, soit aux critères sur l'attribut $\beta$.

2. $R_2 = (R.\alpha \cup R.\beta_s) \Leftrightarrow R_1$ : ensemble des éléments structurels qui répondent soit aux critères sur l'attribut $\alpha$, soit aux critères sur l'attribut $\beta$ et qui ne sont pas des éléments sources pour $\alpha$.

3. $R'_2 = (R.\alpha_s \cup R.\beta) \Leftrightarrow R_1$ : ensemble des éléments structurels qui répondent soit aux critères sur l'attribut $\alpha$, soit aux critères sur l'attribut $\beta$ et qui ne sont pas des éléments sources pour $\alpha$.

4. $R_3 = (R.\alpha \cup R.\beta) \Leftrightarrow (R_1 \cup R_2 \cup R'_2)$ : ensemble des éléments structurels qui répondent soit aux critères sur l'attribut $\alpha$, soit aux critères sur l'attribut $\beta$ et qui ne sont pas des éléments sources pour $\beta$.

5. ... *la construction des ensembles se poursuit comme pour l'opérateur ET.*

Comme pour le ET, dans le cas où la correspondance partielle n'est pas possible pour les deux attributs, nous nous limitons aux ensembles $R_1$, $R_2$, $R'_2$ et $R_3$.

### 7.5.6 L'opérateur SAUF

Soit la requête ($q.\alpha\ op_\alpha\ x_\alpha$) SAUF ($q.\beta\ op_\beta\ x_\beta$) et les ensembles de réponses respectifs. L'opérateur SAUF se traduit par une différence ensembliste en lieu et place des opérateurs intersection et union.

1. $R.\alpha_s \Leftrightarrow R.\beta)$ : ensemble des éléments structurels sources de $\alpha$ qui répondent exactement dont nous retirons les éléments structurels qui répondent exactement aux critères sur $\beta$.

2. $(R.\alpha \Leftrightarrow R.\alpha_s) \Leftrightarrow R.\beta_s$ : ensemble des éléments structurels non sources de $\alpha$ qui répondent exactement dont nous retirons les éléments structurels qui répondent exactement aux critères sur $\beta$.

3. $R.\alpha_s^- \Leftrightarrow (R.\beta \cup R.\beta^-)$ : ensemble des éléments structurels sources de $\alpha$ qui répondent partiellement dont nous retirons les éléments structurels qui répondent exactement et partiellement aux critères sur $\beta$.

4. $(R.\alpha^- \Leftrightarrow R.\alpha_s^-) \Leftrightarrow (R.\beta \cup R.\beta^-)$ : ensemble des éléments structurels non sources de $\alpha$ qui répondent partiellement dont nous retirons les éléments structurels qui répondent exactement et partiellement aux critères sur $\beta$.

### 7.5.7 Les limites actuelles

Notre modèle de correspondance connaît actuellement certaines limites que nous nous devons de mettre à jour. Nous reprenons ainsi les différents points que nous ne traitons pas dans le mode de correspondance tel qu'il est défini.

*L'opérateur between* : nous ne donnons pas d'interprétation pour cet opérateur. Nous pensons qu'il est toutefois possible de coupler deux parcours sur l'ensemble des éléments sources. Le premier parcours se faisant par exemple avec l'opérateur qui fixe la borne inférieure. Puis le second parcours se fait sur le résultat fourni par le premier parcours et va fixer la borne supérieure avec l'opérateur inverse.

*Des termes d'indexation pondérés* : nous ne considérons jamais dans cette approche des termes d'indexation pondérés. Il est clair que la mise en jeu de pondérations introduirait un nouveau niveau de classement dans les réponses. Toutefois cela nécessite la définition de fonctions de combinaison cohérentes qui puissent générer des relations sur les valeurs réutilisables.

*La fonction de propagation* : nous avons posé somme hypothèse que l'utilisation de nos ensembles résultats dépend d'une fonction de propagation correspondant obligatoirement à l'identité. Cette hypothèse forte intervient pour simplifier les traitements.

Considérons maintenant que la fonction de propagation ne correspond pas à l'identité. Cela signifie que les valeurs des attributs des éléments appartenant aux portées ont chacune des valeurs différentes mises en évidence par la relation $R_\alpha$. Il est donc nécessaire de fouiller dans chacun de ces ensembles et nous ne pouvons nous contenter de la définition que nous avons donnée de l'ensemble $R.\alpha$. De plus nous ne pourrions plus éliminer les éléments des portées dont l'élément source ne répond pas. Il faudrait consulter la relation $R_\alpha$ pour déterminer s'il faut ou non consulter les éléments de ces portées.

L'absence de cette hypothèse nécessite une nouvelle étude sur le modèle de correspondance et de nouvelles stratégies de recherche.

## 7.6 Synthèse

Maintenant que nous avons décrit les caractéristiques de notre modèle de correspondance ainsi que ces limites actuelles, nous discutons des principaux points que nous souhaitons

mettre en avant. Nous souhaitons dans un premier temps rappeler l'usage que nous avons fait des portées des attributs et l'impact naturel qui découle d'une meilleure prise en compte de la structure dans le processus d'indexation. Nous poursuivons et insistons sur la visualisation des résultats qui n'a pas été décrite dans ce chapitre.

*L'usage des portées des attributs*

L'espace d'informations constitué par le fonds de documents structurés a été considérablement modifié de deux points de vue. Tout d'abord, il a été nécessaire de considérer que le document est une agrégation de composants et que la plupart de ces composants sont dorénavant des réponses possibles à des requêtes, c'est-à-dire qu'ils peuvent indépendamment du document complet remplir les besoins d'un utilisateur. Nous avons donc augmenté le nombre d'éléments accessibles par des requêtes. Le second aspect concerne l'introduction des portées des attributs et l'impact de leur application sur l'espace d'informations. Les portées ne modifient pas les informations présentes dans cet espace, mais elles les répartissent sur les composants du document, c'est-à-dire que les informations pertinentes vont atteindre chacun des composants concernés. Autrement dit, les portées rendent explicites des informations qui demeuraient auparavant implicites.

L'effet au niveau de l'interrogation est sensible puisque nous sommes dorénavant en mesure de restituer les composants des documents plutôt que les documents complets. Nous gagnons donc en précision dans les réponses. De plus, la nouvelle couverture du document acquise par l'application des portées rend accessible, par l'interrogation, des composants régulièrement oubliés pas les approches traditionnelles.

*La visualisation des réponses*

Nous souhaitons insister sur la phase de visualisation des réponses, qui consiste à présenter à l'utilisateur la liste des composants qui répondent pour lui permettre de consulter ces composants. Traditionnellement, les systèmes affichent une liste triée de documents qui sont des pointeurs vers les documents complets. L'application de cette technique sur des documents monolithiques demeure contestable [DBM97]. Avec l'apparition des documents structurés, et la possibilité de restituer des composants de documents, ces techniques traditionnelles semblent désuètes.

Nous sommes capables de rendre des composants de documents, assimilables à des sous-arborescences de l'arborescence structurelle complète du document, mais nous devons éviter d'une part la redondance dans les réponses et nous devons d'autre part refléter la structure des documents.

Il faut donc, comme nous l'avons fait dans le système PRIME-GC, mettre en évidence les différents niveaux de réponse que nous proposons. Ainsi, nous devons proposer un protocole de visualisation qui, à partir des ensembles de réponses, construit une présentation des documents qui répondent en reflétant autant la position des composants pertinents que le degré de pertinence que le système leur a assigné. La visualisation des réponses est selon nous un problème ouvert qui associe les systèmes de recherche d'informations aux techniques de visualisation d'interface homme-machine. Nous ne pouvons pour notre part répondre complètement à ce problème et adoptons des méthodes ad-hoc de présentation pour éviter la redondance et refléter la structure mais sans pouvoir généraliser ni formaliser cette approche.

# Chapitre 8

# Mise en oeuvre du modèle, le prototype *my Personal Daily News*

> "The economic models of media today are based almost exclusively on "pushing" the information and entertainment out into the public. Tomorrow's will have as much or more to do with "pulling," where you and I reach into the network and check out something the way we do in a library or video-rental store today. This can happen explicitly by asking or implicitly by an agent asking on your behalf."
>
> Nicholas Negroponte - "Being Digital"

## 8.1 Introduction

Le modèle général de représentation des documents structurés ainsi que les protocoles d'indexation et de recherche définis dans les chapitres précédents sont à la base d'une expérimentation dont le résultat est un prototype de système de recherche d'informations sur une base constituée de quotidiens d'informations : *my Personal Daily News*, nommé *myPDN*.

A travers l'implantation de notre modèle et des protocoles que nous avons définis, nous cherchons d'une part à valider notre étude et d'autre part à évaluer les capacités de ce modèle. L'idée est de pouvoir consulter des journaux, qui sont de plus en plus fréquemment disponibles sur Internet, en effectuant des recherches sur le contenu de ces journaux et/ou leur structure. Ainsi, il devient possible d'extraire des articles de plusieurs journaux traitant d'une même thématique, de consulter directement les titres des "unes" de ces différents journaux sans avoir à les fouiller un à un, etc. Nous nous concentrons dans ce chapitre sur la partie de ce système ayant trait à notre modèle de représentation et de recherche.

Le prototype *myPDN* est basé sur le système à objets $O_2$. Le choix de ce système s'explique tout d'abord parce qu'un système à objets correspond bien à la modélisation que nous avons faite des documents structurés (chapitre 5). De plus, le système $O_2$ présente des outils multimédias et des outils de communication suffisants avec les serveurs HTTP pour consulter

et interroger un corpus de documents depuis des clients WWW.

Nous présentons dans une première partie la base de documents qui a été constituée pour le prototype *myPDN*. Par la suite, nous donnons l'architecture générale du prototype en explicitant les différents traitements subis par les documents ainsi que les principaux modules qui interviennent dans le prototype *myPDN*. Nous décrivons plus spécifiquement dans la section 8.4 les choix généraux de conception de *myPDN* en nous attardant particulièrement sur certains modules : le gestionnaire de documents et le gestionnaire d'attributs. Nous donnons par la suite la description du gestionnaire de requêtes ainsi que les résultats obtenus.

## 8.2 La base de documents

Actuellement, le prototype *my Personal Daily News* gère 6 journaux du quotidien d'informations Libération disponible sur internet [1]., ce qui représente près de 100 articles. Nous avons choisi ces documents pour des raisons de disponibilité mais aussi pour les caractéristiques suivantes :

- Les quotidiens d'informations sont de plus en plus fréquemment *disponibles* sur Internet et ils peuvent donc être entièrement récupérés sous leur forme électronique.

  Nous avons ainsi rapatrié le site complet du journal Libération à 6 dates différentes afin de constituer notre base de documents. Les documents récupérés se présentent sous la forme de fichiers HTML inter-reliés qu'il nous a, par la suite, fallu charger dans une base $O_2$. Le rapatriement d'un site Web s'est fait à l'aide d'un utilitaire écrit en Perl qui à partir d'une adresse initiale ramène sur notre machine locale le fichier correspondant à cette adresse ainsi que les fichiers référencés par des liens de navigation. Ce processus est appliqué récursivement sur chaque fichier HTML du site avec une profondeur qui est un paramètre de l'utilitaire.

- Les quotidiens d'information que nous avons rencontrés partagent une structure commune que nous avons présentée dans le chapitre 5. Cette structure présente la particularité d'être suffisamment profonde pour pouvoir exploiter pleinement notre modèle. L'application PRIME-GC [BFM+97], décrite dans la partie 2.5.3 (page 53), sur laquelle nous avons déjà expérimentée cette approche ne présentait que deux niveaux de profondeur, Nous avons jugé cela insuffisant pour mettre complètement en oeuvre le processus d'indexation structurelle.

  Même si nous pouvons regretter de ne pas traiter plusieurs types de structure dans notre prototype, il nous semble primordial d'évaluer dans un premier temps notre approche sur la portée des attributs sur un corpus constitué de documents partageant une même structure avant d'évoluer vers des corpus constitués de structures hétérogènes.

- Les quotidiens d'information sont composés d'articles qui alternent les parties textuelles et les illustrations graphiques. Il s'agit donc de documents combinant les deux médias que nous cherchons à représenter : les images et le texte.

1. Adresse du site de Libération: http://www.liberation.com

– Enfin, les quotidiens d'information abordent des thèmes variés et ne se limitent pas à un domaine particulier comme le domaine médical dans l'application PRIME-GC. Il s'agit donc d'un système de recherche d'informations au vocabulaire d'indexation ouvert et qui se veut généraliste.

Nous devons insister sur le fait que les documents, une fois récupérée sous la forme de fichiers HTML, ont été traitées afin de pouvoir être chargés dans la base. Ce traitement, en grande partie manuelle, a consisté à identifier dans les fichiers HTML les éléments de structure correspondant aux types de structure que nous proposons, puis à introduire des marqueurs pour délimiter ces éléments dans les fichiers.

Comme le montre différents travaux [Chr96, CPC94], l'accueil de documents structurés dans un SGBD OO, c'est-à-dire la définition du schéma d'une base à partir de la structure des documents nécessite un mécanisme de traduction qui n'est pas abordé ici. Notre objectif prioritaire est de démontrer la validité et l'intérêt des attributs dynamiques dans le cadre de la recherche sur une base constituée de documents structurés.

Notre protocole de chargement, adapté aux documents que nous manipulons, s'établit de la manière suivante : un objet de la classe Un_Element est créé pour chaque élément de structure reconnu dans un fichier HTML, les attributs de cet objet décrivant les relations de composition et de séquence sont ensuite instanciés.

Nous avons aussi dû choisir un mode de stockage des données physiques des documents. Il est évident pour nous que la séparation entre la structure et les données doit être explicite dans la base. Deux possibilités se présentent alors à nous :

– Conserver le document sous forme monolithique et mettre en place un système de pointeurs permettant d'identifier, pour chaque objet de la base, la partie du document qui lui correspond. C'est pas exemple l'approche adoptée par Navarro & Baeza-Yates dans [NBY95b]. Cette approche est souhaitable dans le cadre d'applications manipulant plusieurs vues structurelles du document comme le fait Navarro. Cette approche devient toutefois délicate à gérer pour des documents qui évoluent.

– Stocker le document sous la forme de l'arborescence de ses éléments structurels. Nous conservons ainsi dans la base les différentes parties du document et représentons à l'aide d'agrégation ou de listes les relations structurelles entre les éléments. Cette approche nécessite un module particulier pour la visualisation du document qui permette de reconstruire le document à partir de ces composants. C'est aussi cette approche qui montre le mieux la différence entre les structures logiques et de présentation: nous stockons l'arborescence correspondant à la structure logique et pouvons présenter de différentes manières un même document en appelant des méthodes de présentation différentes.

Nous avons choisi la seconde solution qui consiste à assigner aux éléments feuilles de l'arborescence structurelle les données qui constituent le document. D'une part cette solution présente l'avantage de pouvoir manipuler les documents sans se soucier de la gestion de la cohérence des pointeurs. Notre choix a donc été guidé par le souci de reconstruction et de présentation des réponses. Nous revenons sur ce point par la suite.

La conséquence de notre choix est l'éclatement du contenu du document, du moins dans la base de données. L'utilisateur, quant à lui, ne visualisera pas cet éclatement car nous proposons un module de présentation qui permet de visualiser uniquement des éléments de la structure logique compréhensibles par le lecteur. Ainsi il ne permet pas d'afficher seulement un objet représentant une ancre. Bien que nous nous soyons peu préoccupés jusqu'ici de la structure de présentation, il est apparu indispensable lors de notre expérimentation de choisir des entités "présentables" parmi les éléments structurels. Il s'agit ici du rôle dévolu à l'interface de présentation et d'une illustration de la différence notable entre la structure syntaxique du document et sa structure de présentation.

## 8.3 L'architecture

Notre prototype est construit au dessus du SGBD à objets $O_2$ version 4.6 et s'organise autour de plusieurs entités dont nous donnons les principales fonctionnalités et sur lesquelles nous revenons par la suite. Nous référençons ces entités dans la figure 8.2 qui décrit l'architecture générale de l'application.

- Un *noyau de gestion des documents structurés*. Cette entité offre les fonctions essentielles de manipulation des documents structurés en respectant notre modèle de représentation du document structuré. Chacun des trois relations de structure que nous avons proposées pour articuler l'organisation des éléments de structure sont présentes dans cette entité ainsi que les fonctionnalités de parcours de l'arborescence structurelle.

- Un *noyau de gestion des attributs dynamiques*. Dans *myPDN*, nous séparons explicitement les documents de leurs attributs. Ainsi, la gestion des attributs dynamiques est indépendante de la gestion des documents. Au niveau de l'application, nous établissons une communication entre les attributs et les éléments de structure qui représente au niveau du modèle les relations $R_{ST}$ et $R_M$ (chapitre 5), et surtout la fonction *value* qui fournit la valeur d'un attribut pour un élément structurel donné.

  Cette séparation nous permet par ailleurs de conserver les informations issues du modèle de document parallèlement aux informations issues du modèle de document étendu, et donc de l'indexation structurelle.

- Un *noyau de mise en correspondance*. Ce noyau prend une requête en entrée et restitue des documents ou des composants en réponse. Il s'agit ici de la mise en oeuvre du processus de correspondance dépendant de notre modèle de représentation et de notre protocole d'indexation structurelle.

- Une *interface d'interrogation* élémentaire et une *interface de présentation*. Nous utilisons pour cela l'outil fourni par $O_2$ permettant de coupler une base de données $O_2$ à un serveur Web: $O_2$Web. Nous donnons dans la figure 8.1, les principaux éléments de l'architecture de $O_2$Web couplés au serveur $O_2$ de notre base:

  **o2webgateway** : la porte d'accès à $O_2$Web. Lorsqu'un client du World Wide Web émet une requête, celle-ci est transmise par le serveur HTTP au o2webgateway.

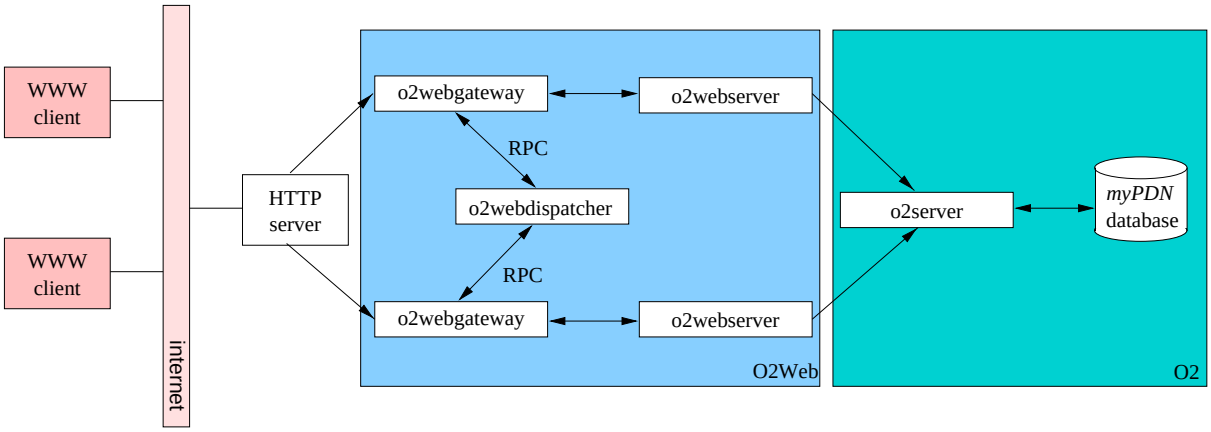



**Figure 8.1.** Architecture Générale d'accès à *myPDN* via O2Web

**o2webdispatcher** : l'élément de contrôle pour distribuer les requêtes des o2webgateway. Ceux-ci se connectent à l'o2webdispatcher afin de savoir quel serveur $O_2$Web ils doivent contacter.

**o2webserver** : les serveurs $O_2$Web reçoivent les requêtes des o2webgateway, les exécutent sur la base spécifiée puis les transforment en document HTML pour les restituer au client du World Wide Web qui avait émis la requête.

Cet outil nous permet de visualiser les données de la base $O_2$, ici des documents structurés, sur tout client du World Wide Web (Netscape Communicator, Microsoft Explorer, etc). C'est l'interface de présentation qui va faire appel au gestionnaire de documents structurés pour récupérer les données et les transmettre au serveur $O_2$Web qui va alors les restituer au serveur HTTP.

Nous décrivons maintenant plus précisément les éléments que nous avons implantés pour accueillir nos documents sur le SGBD à objets $O_2$ en respectant le schéma de la figure 8.2. Dans cette figure, nous séparons les éléments de notre application et ceux du système $O_2$ et de l'outil $O_2$Web.

Il apparaît tout d'abord, d'après cette architecture générale, que nous avons clairement séparé le stockage des descripteurs des éléments structurels (2) de celui du contenu des éléments structurels (1). D'autre part, nous introduisons au dessus de ces deux sortes de données deux gestionnaires élémentaires : le premier pour la gestion des documents structurés (3) et le second pour la gestion des attributs de ces documents structurés (4). Ces deux gestionnaires communiquent pour savoir quels attributs décrivent quels éléments de structure.

Plus proche de l'utilisateur, et pour communiquer avec le client WWW, notre système propose deux modules d'interfaçage : l'un pour l'interrogation (8) et l'autre pour la présentation des documents et la consultation hypertextuelle des documents (7). L'interface d'interrogation est en relation directe avec un gestionnaire de requêtes (5) qui prend en charge la compréhension des requêtes et les transmet par la suite à un répartiteur de requêtes (6) dont le rôle

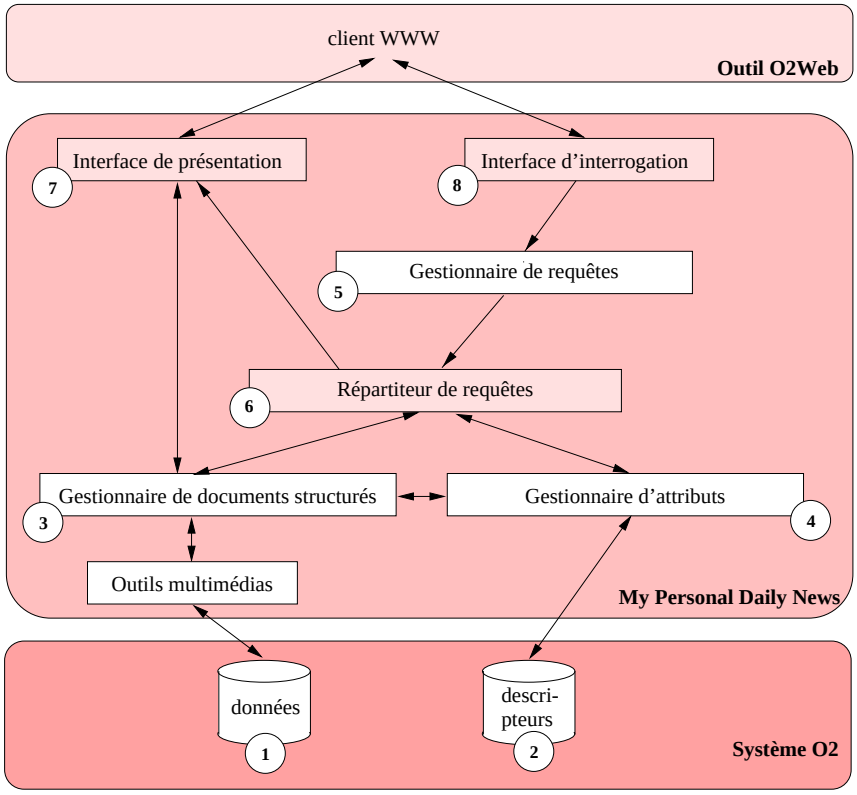



**Figure 8.2.** Architecture Générale de *MyPDN*

est double: répartition des requêtes structurelles et des requêtes de contenu ainsi que prise en charge de la fusion des résultats fournis par les gestionnaires de documents structurés et d'attributs. Ce répartiteur fournit par la suite les résultats à l'interface de présentation qui les affiche à l'utilisateur puis réalise les accès aux documents pour leur consultation.

Dans la figure 8.3, nous donnons un exemple de présentation à partir de notre prototype d'une "une" du journal Libération du 25 Septembre 1997. Chaque lien de cette page pointe sur un objet de la base $O_2$ et l'activation de ce lien déclenche l'appel de la méthode de présentation de l'objet qui fournit une nouvelle page HTML.

Dans les sections qui suivent, nous détaillons le gestionnaire de documents structurés et le gestionnaire d'attributs. Nous spécifions les choix de conception que nous avons faits pour ces modules. Ensuite, nous explicitons les traitements des requêtes.


**Figure 8.3.** Présentation de la "une" du journal Libération avec *myPDN*

## 8.4 Le gestionnaire de documents et leur présentation

Nous décrivons ici les choix que nous avons faits lors de la conception de notre application. Dans un premier temps, nous abordons les choix généraux de conception puis nous décrivons plus précisément les choix pour la représentation des documents aux travers de certaines classes que nous avons définies. Nous revenons aussi sur la gestion des données multimédias ainsi que sur les méthodes de présentation que nous proposons dans notre application.

### 8.4.1 Choix généraux de conception

Notre modèle de représentation du document structuré comporte trois types de relations de structure qui lient des éléments typés dont les données physiques sont soit du texte, soit des images, soit du texte et des images. De plus, la description de ces éléments se fait par des attributs.

Il existe en général deux manières de porter des documents structurés sur un SGBD à objets. La première, dite "fortement typée", associe à chaque type d'élément du document une classe d'objets dans le SGBD. C'est cette approche qui a été adoptée dans le travail de Christophides [Chr96]. Elle nécessite la création de nouveaux types qui ne sont pas actuellement disponibles sur le SGBD $O_2$. La seconde approche, dite "faiblement typée", considère une unique classe d'objets pour tous les types d'éléments du document. Le type est alors fourni par un marqueur, c'est-à-dire un attribut de la classe.

C'est cette dernière approche que nous avons adoptée, même si nous pensons qu'il ne s'agit pas de la plus valorisante pour l'accueil de documents sur un SGBD. Mais comme l'a montré le travail de Christophides, définir une structure d'accueil pour des documents structurés sur un SGBD à objets est un travail en soi qui nécessite des modifications internes au SGBD. Il est aussi possible, comme cela est préconisé dans [CPC94], de fournir une modélisation plus fine tenant compte de types de structures plus génériques et proposant par exemple des entités spécifiques pour les relations hypertextuelles. Cependant nous souhaitions "coller" à notre modèle et considérer chaque élément de la structure des documents de manière uniforme, que ce soit un article complet ou la source d'une relation hypertextuelle, ceci afin d'évaluer la faisabilité de cette approche.

Nous devons aussi représenter les attributs ainsi que les propriétés structurelles que nous leur avons adjointes. Nous décrivons donc les choix que nous avons faits en précisant qu'actuellement le processus d'indexation structurelle qui modifie à la fois le modèle de document et les attributs qui le décrivent n'est pas totalement dynamique dans notre application. En effet, la modification du modèle de document dérivant de la définition des portées des attributs ne se fait pas automatiquement.

Nous avons décrit la structure des documents sur lesquels nous travaillons dans le chapitre sur le modèle de représentation des documents structurés. Nous allons ici donner les choix d'implantation ainsi que les méthodes que nous avons adoptées pour stocker et présenter ces documents.

### 8.4.2 Définition de classes

Le choix d'une approche faiblement typée implique que la représentation interne des documents structurés s'articule autour d'une unique classe d'objets : la classe Un_Element. Cette classe, en dehors de la considération des descripteurs que nous voyons par ailleurs, comporte les informations nécessaires à la représentation des relations de structure des documents. Nous considérons ainsi deux attributs pour la relation de composition, ascendant et descendants, deux attributs pour la relation de séquence, prev et next, et enfin un attribut pour la relation

de référence, ref. Nous explicitons ici les choix faits pour ces trois relations.

*La composition* : l'attribut ascendant pointe sur l'objet correspondant à l'ascendant structurel direct dans l'arborescence et l'attribut descendants pointe sur la liste des objets correspondant aux descendants structurels. L'ordonnancement de cette liste correspond à la séquence linéaire. La classe Liste_Element est une liste d'objets de classe Un_Element avec des propriétés particulières pour le traitement des listes.

Les fonctions $Prev_{comp}$ et $Next_{comp}$ qui fournissent respectivement l'ensemble des ascendants et l'ensemble des descendants sont instanciés sous la forme de méthodes de la classe Un_Element.

*La séquence* : l'attribut prev pointe sur l'objet correspondant au prédécesseur dans la séquence et l'attribut next pointe sur l'objet correspondant au successeur dans la séquence.

Les fonctions $Prev_{seq}$ et $Next_{seq}$ qui fournissent respectivement l'ensemble des ascendants et l'ensemble des descendants sont instanciées sous la forme de méthodes de la classe Un_Element. Elles comportent les mêmes particularités que les fonctions du modèle puisqu'elles parcourent la structure en largeur afin de restituer l'ensemble des successeurs d'un élément structurel, c'est-à-dire qu'elles ne considèrent pas seulement les éléments qui ont même père que l'objet considéré.

*La référence* : l'attribut ref pointe sur l'objet destination de la relation de référence.

Nous donnons la déclaration de la classe Un_Element sur $O_2$. Cette classe hérite des méthodes d'initialisation et de présentation de la classe Une_Page. Comme nous allons le voir par la suite, elle possède des méthodes de présentation complémentaires dépendantes du type de l'élément structurel.

```
class Un_Element inherit Une_Page public type
tuple(
    adresse: string,              /* provenance de l'élément: une adresse http */
    type: string,                 /* type de l'élément pris dans l'ensemble TYPE */
    ascendant: Un_Element,        /* référence à l'ascendant structurel direct */
    descendants: Liste_Element,   /* référence aux descendants structurels directs */
    prev: Un_Element,             /* référence au prédécesseur séquentiel direct */
    next: Un_Element,             /* référence au successeur séquentiel direct */
    ref: Un_Element,              /* référence à la destination d'une relation de référence */
    content: Multimedia)          /* s'il s'agit d'un élément feuille: référence à son contenu
                                     physique, sinon objet NIL*/
/* méthode d'accès à l'ensemble des ascendants structurels */
method Prev_comp (): set(Un_Element);
/* méthode d'accès à l'ensemble des descendants structurels */
method Next_comp (): set(Un_Element);
/* méthode d'accès à l'ensemble des prédécesseurs séquentiels */
method Prev_seq (): set(Un_Element);
/* méthode d'accès à l'ensemble des successeurs séquentiels */
method Next_seq (): set(Un_Element);
end;
```

### 8.4.3 Les éléments multimédias

Nous avons tout d'abord été confrontés aux problèmes liés à la prise en compte dans des documents d'éléments multimédias. Nous devons gérer indifféremment des éléments monomédias (texte ou image) et des éléments multimédias (texte et image).

Les données sont stockées au niveau des feuilles de la structure des documents. Ce sont donc des éléments monomédias, texte ou image. Les objets représentant des feuilles ont donc leur attribut content qui est instancié alors que les objets représentant des éléments non feuilles n'admettent pas de valeur. Leur contenu est construit à partir des objets qui les composent.

Nous avons réutilisé les classes Text et Image, préexistantes sur le système $O_2$, et créé notre propre classe capable d'accueillir indifféremment du texte ou des images. Cette classe possède deux attributs : le premier permet d'avoir des données textuelles alors que le second est pour les images. Un marqueur indique de quel type de donnée il s'agit.

Là encore des méthodes génériques de présentation de ces documents sont fournies pour cette classe. Lors de la présentation d'un élément, le type du média n'interviendra donc pas et nous pourrons indifféremment afficher du texte ou des images.

### 8.4.4 Construction et présentation des documents

Les éléments structurels sont stockés dans la base sous la forme d'objets. Comme nous l'avons dit précédemment, nous avons utilisé l'outil $O_2$Web pour interfacer notre application. Il a donc fallu décrire des propriétés de présentation pour les documents qui utilisent les propriétés des objets que nous stockons. Il s'agit d'un module d'interaction avec l'utilisateur chargé de présenter les objets stockés dans la base.

Nous avons donc décrit pour chaque type d'éléments de structure une propriété de présentation en cherchant à respecter leur présentation originale. Ces méthodes nécessitent un parcours en profondeur de la structure afin de retrouver le contenu des éléments que nous stockons dans les objets représentant les feuilles de la structure.

Nous créons également des relations de navigation hypertextuelle entre les éléments appartenant à une séquence afin de donner une représentation de cette relation de séquence. Par exemple, chaque article est présenté sur une page HTML, et il est possible d'accéder à l'article suivant et à l'article précédent. Il en va de même pour les journaux que nous avons stockés sous la forme d'une liste.

Par exemple, la présentation de l'article de la figure 8.4 s'effectue par l'appel de la méthode générique html_report qui, après avoir reconnu le type de l'élément, fait appel à la méthode spécifique au type d'élément article : display_article.

En général, chaque article comporte un titre, un sous-titre, des éléments de référence (présentés sur la gauche du document) et un corps composé de plusieurs paragraphes et/ou d'images. Chacun des éléments possède sa propre méthode de présentation dépendante du type de ces éléments.

Nous pouvons aussi remarquer que chaque lien de référence présent dans le document est restitué au niveau de la présentation sous la forme d'une relation de navigation dont l'activation déclenche la présentation de l'élément cible de la référence.

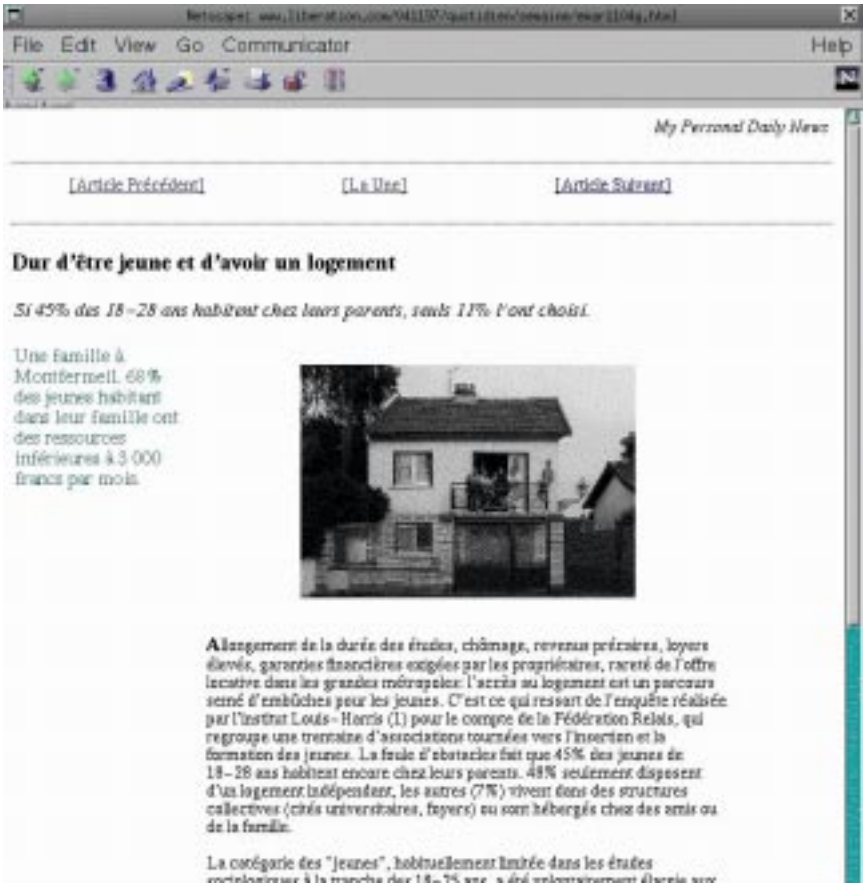



**Figure 8.4.** Présentation d'un article du journal Libération avec myPDN

Nous donnons deux exemples de navigation qui permettent de se déplacer dans la structure du document. Le premier exemple traite de la première page des journaux: la "une", et le second décrit les relations pour naviguer parmi les articles.

*Exemple de la "une" du journal Libération (figure 8.5)*

1. *Accès à la page originale*: nous proposons un accès à la page originale du site, c'est-à-dire à un fichier HTML qui n'est pas stocké dans notre base mais dont nous avons conservé la référence (adresse du site). L'utilisateur peut ainsi vérifier que la présentation originale des documents est restituée dans sa majeure partie.

2. *Accès au journal précédent et au journal suivant*: nous proposons ici des moyens pour suivre la relation de séquence basée sur les dates de publication des journaux. Ces liens permettent d'accéder aux "unes" des journaux précédents et suivants. Il s'agit ici d'accès à des objets de notre base qui font donc appel à nos méthodes de présentation spécifique, méthode display_editorial.

3. *Accès à chaque article du journal*: la "une" d'un journal fait office de sommaire, c'est-

à-dire qu'elle présente de manière synthétique chacun des articles contenus dans le journal complet. Les liens que nous proposons à partir de chacune de ces présentations synthétiques pointent sur les articles et représentent les relations de composition entre le journal complet et les composants de type article. L'activation de ces liens engendre l'appel de la méthode de présentation d'un article, display_article.

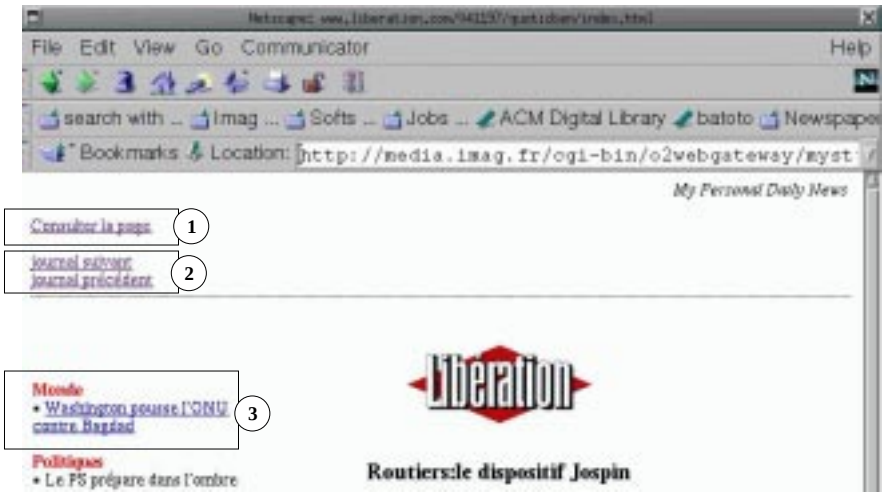



**Figure 8.5.** Eléments de navigation de la "une" d'un journal avec myPDN

*Exemple d'un article du journal Libération (figure 8.6)*

Les trois exemples de navigation proposés dans l'en-tête d'un article correspondent à des représentations au niveau de la présentation des relations de séquence et de composition qui interviennent sur les éléments de type article. L'activation de ces liens déclenche l'appel de méthodes de présentation sur des objets de la base.

1. *Accès à l'article précédent*: nous proposons un accès à l'article suivant, c'est-à-dire une représentation par un lien hypertextuel de la relation de séquence établie entre les articles d'un journal.

2. *Accès à la "une" du journal*: comme nous l'avons dit précédemment, la "une" d'un journal représente au niveau de l'affichage le journal complet. Cet accès symbolise donc la remontée dans la structure depuis un article vers le journal parent. Il existe une différence fondamentale entre la structure logique, celle que nous représentons dans le SGBD, et la structure de présentation de nos documents. La "une" d'un journal correspond à la racine de la structure de présentation alors qu'il s'agit d'un composant du journal dans la structure logique.

3. *Accès à l'article suivant*: il s'agit ici d'accéder au prédécesseur direct dans la structure selon la relation de séquence.


**Figure 8.6.** Eléments de navigation de chaque article d'un journal avec myPDN

## 8.5 Le gestionnaire d'attributs

Il s'agit tout d'abord de déterminer comment nous allons représenter les attributs dynamiques au niveau du schéma $O_2$, puis nous établissons des choix quant à la gestion de ces attributs en cherchant à respecter les éléments décrits dans notre modèle. Les fonctionnalités que nous donnons sont particulièrement importantes pour la gestion des éléments sources d'information.

Les attributs statiques sont traités par une approche traditionnelle.

### 8.5.1 Choix de représentation

Nous présentons tout d'abord la hiérarchie des classes que nous avons définies et implantées pour les attributs dynamiques. Nous expliquons d'où viennent les choix que nous avons faits en reprenant les caractéristiques de notre modèle. Ensuite nous décrivons plus spécifiquement les propriétés de l'une de ces classes, celles pour les attributs de composition ascendant.

*Hiérarchie des classes d'attributs dynamiques*

Nous avons établi lors de la définition du processus d'indexation structurelle une classification des attributs dépendante du type de portée. Nous la reprenons ici pour établir les classes ainsi que la hiérarchie de ces classes. Dans notre classification, les attributs d'une même classe partagent des propriétés communes que nous allons décrire dans le schéma de la base par l'intermédiaire de *classes virtuelles*.

Comme nous le montrons dans la figure 8.7, chaque type d'attribut correspond à une classe virtuelle (noms en italique dans la hiérarchie). Nous définissons ces classes ainsi que les propriétés particulières de ces classes : la propriété de propagation, la propriété de combinaison et l'ensemble des éléments atteints.

Nous définissons donc tout d'abord une classe virtuelle AttributDyn et cinq autres classes

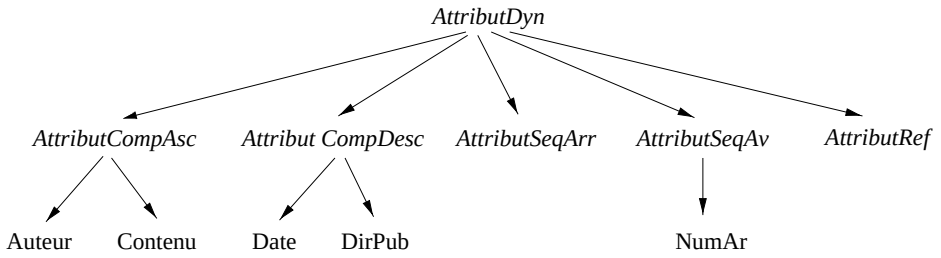



**Figure 8.7.** Hiérarchie des classes d'attributs

virtuelles correspondant à la classification que nous avions établie (section 6.5 page 161) :

AttributCompAsc : attribut de composition ascendant, qui se propage des feuilles vers la racine du document en suivant la relation de composition.

AttributCompDesc : attribut de composition descendant, qui se propage de la racine vers les feuilles du document en suivant la relation de composition.

AttributSeqArr : attribut séquentiel arrière, qui se propage en suivant la relation de séquence dans le sens opposé à la lecture linéaire.

AttributSeqAv : attribut séquentiel avant, qui se propage en suivant la relation de séquence dans le sens de la lecture linéaire.

AttributRef : attribut de référence, qui suit une relation de référence.

Ces cinq classes héritent de la classe AttributDyn. Nous voyons sur la figure que pour un attribut Auteur, défini comme un attribut de composition ascendant, nous créons une classe Auteur qui hérite de la classe virtuelle AttributCompAsc. Dans la classe Auteur, la propriété contenant la valeur de l'attribut est ajouté aux propriétés de AttributCompAsc. Cette propriété, ainsi que les propriétés de propagation et de combinaison doivent aussi apparaître.

Nous passons maintenant à la description de la classe virtuelle pour les attributs de composition ascendants. Les autres classes sont construites selon ce même modèle en suivant les caractéristiques définies dans la classification des attributs que nous avons établie.

*Classe générique pour les attributs de composition ascendants*

Nous donnons un exemple plus précis avec la classe virtuelle des attributs de composition ascendants, que nous nommons AttributCompAsc, et qui hérite des propriétés de la classe AttributDyn.

```
class AttributCompAsc inherit AttributDyn public type
tuple(
    source: boolean,                /* marqueur indiquant si l'attribut est source */
    node: Un_Element,               /* référence à l'élément structurel décrit */
    pred: list(AttributCompAsc),    /* liste des attributs donnant la provenance de la valeur */
    succ: AttributCompAsc)          /* référence à l'attribut recevant une valeur */
/* méthode représentant la propagation des valeurs dans une portée: fonction f_α */
method private propage();
/* méthode représentant la combinaisons inter-portées: fonction combine_α */
method private combine();
/* méthode restituant la portée d'un élément source: ensemble EP^o_{α,comp} */
method public scope(): set(Un_Element)
end;
```

Parmi les propriétés de cette classe, nous trouvons un marqueur permettant d'identifier les éléments sources, la référence à l'élément structurel décrit, node, ainsi que des références aux éléments concernés par la portée dans le cas d'éléments sources, par l'intermédiaire d'objets AttributCompAsc.

Les propriétés pred et succ fournissent une représentation de la relation $C_\alpha$ pour un attribut de composition ascendant $\alpha$.

Les méthodes propage et combine sont quant à elles respectivement des instanciations des fonctions de propagation et de combinaison. Pour tous les types d'attributs, la propagation correspond à la copie (égalité). Pour un attribut de composition ascendant, la propriété de combinaison est par exemple l'union pour un attribut Auteur. La méthode scope fournit l'ensemble des éléments structurels concernés par un attribut. Cette méthode ne peut être appelée que pour des éléments sources.

### 8.5.2 Fonctionnalités du gestionnaire d'attributs

Nous avons vu dans le mode de correspondance que nous proposons l'importance de l'ensemble $OS_\alpha$ et de son ordonnancement par la relation $C_\alpha$. Nous cherchons ici à fournir une structure permettant de rendre cette organisation au niveau du schéma de la base $O_2$.

*Gestion des attributs des éléments sources*

Il s'agit actuellement d'une gestion des attributs qui est locale au document. La gestion des attributs des éléments sources passe par la définition d'une structure permettant de représenter dans le schéma $O_2$ l'ensemble $OS_\alpha$ et l'ordre non total $C_\alpha$ sur les valeurs d'attribut des éléments sources. Cet ensemble $OS_\alpha$, outre la caractéristique d'accueillir les éléments sources de l'attribut $\alpha$ d'un document, est utilisé lors de l'interrogation, et particulièrement les ensembles d'éléments minimums et maximums de $OS_\alpha$ selon la relation $C_\alpha$.

Nous construisons une classe, LesAttributs, pour les ensembles $OS_\alpha$ et l'ordre $C_\alpha$. Dans cette classe, nous décrivons uniquement les éléments minimums et les éléments maximums. A partir de ces éléments, provenant de la classe AttributDyn, et des relations qu'ils contiennent (propriétés pred et succ), nous retrouvons l'ensemble des éléments de $OS_\alpha$. De plus le parcours de ces relations nous restituent l'ordre décrit par la relation $C_\alpha$.

```
class LesAttributs inherit Object public type
tuple(
    max: list(AttributDyn),    /* liste des attributs maximums selon C_α */
    min: ist(AttributDyn))     /* liste des attributs minimums selon C_α */
method parcours_min(): list(AttributDyn);
method parcours_max(): list(AttributDyn);
end;
```

Les deux méthodes parcours_min et parcours_max permettent respectivement de parcourir l'ensemble d'attributs correspondant à l'ensemble des attributs $\alpha$ des éléments sources de $OS_\alpha$ d'un document. C'est par ces méthodes que nous allons effectuer la recherche des attributs répondant à une requête, c'est à dire la recherche des éléments de structure pertinents.

## 8.6 Le gestionnaire de requêtes

Dans notre architecture, nous avons privilégié une séparation explicite entre les données représentant les éléments structurels et les attributs décrivant ces éléments. Toutefois, nous rappelons que ces deux entités communiquent entre elles: d'une part pour indiquer quel élément structurel est décrit par un attribut et d'autre part pour la propagation des attributs lors de l'indexation structurelle.

Au niveau de l'interrogation, les attributs sont indépendants et comme nous l'avons vu dans le chapitre précédent la recherche des éléments pertinents dépend de l'opérateur de recherche de la requête et de la relation liant les valeurs d'attributs au sein de l'ensemble des éléments sources. Nous avons donc développé des classes de requêtes comportant des méthodes spécifiques à chaque attribut. Il faut noter que dans notre application, ces classes dépendent des attributs mais il est possible de les généraliser de manière à fournir une bibliothèque de classes dépendant des propriétés des attributs dynamiques et des opérateurs possibles sur ces attributs.

L'autre aspect de l'interrogation met en jeu les relations structurelles et les éléments de structure. Nous avons ainsi développé une bibliothèque d'opérateurs de recherche basés sur les relations de composition et de séquence. Ces opérateurs prennent un ensemble d'éléments structurels en entrée pour restituer un ensemble d'éléments structurels vérifiant la condition structurelle de l'opérateur. Il s'agit d'opérateurs basiques dont la combinaison permet d'exprimer des conditions complexes. Voici quelques exemples de ces opérateurs que nous exprimons aux travers des notations[2] de notre modèle. Nous notons $E_D$ l'ensemble de départ des éléments et $E_R$ l'ensemble résultat:

- *Eléments ayant des descendants directs de type $t$*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } \exists o_j \in OS \text{ tel que } o_i \prec_{comp} o_j \text{ et } type(o_j) = t\}$$

- *Eléments ayant des descendants de type $t$*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } \exists o_j \in OS \text{ tel que } o_j \in Next_{comp}(o_i) \text{ et } type(o_j) = t\}$$

2. $\prec_{comp}$ est la relation de composition, $\prec_{seq}$ est la relation de séquence, *type* est la fonction d'assignation de type structurel, ...

– *Eléments ayant au moins n descendants de type t*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } R_j \subset Next_{comp}(o_i) \text{ tel que } \forall o_j \in R_j, type(o_j) = t \text{ et } |R_j| > n\}$$

– *Eléments ayant un successeur direct de type t*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } \exists o_j \in OS \text{ tel que } o_i \prec_{seq} o_j \text{ et } type(o_j) = t\}$$

– *Eléments ayant un successeur de type t*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } \exists o_j \in OS \text{ tel que } o_j \in Next_{seq}(o_i) \text{ et } type(o_j) = t\}$$

– *Eléments ayant au moins n successeurs de type t*:

$$E_R = \{o_i \mid o_i \in E_D, \text{ et } R_j \subset Next_{seq}(o_i) \text{ tel que } \forall o_j \in R_j, type(o_j) = t \text{ et } |R_j| > n\}$$

Le gestionnaire de requêtes prend en charge la répartition des requêtes en fonction des spécifications contenues dans celles-ci. Il est ainsi possible d'introduire des critères structurels sur la composition des éléments recherchés, leurs enchaînements ou leurs positions dans la séquence. De plus, ces critères peuvent être complétés par des critères sur les attributs des éléments recherchés. Le gestionnaire de requêtes crée des objets requêtes correspondant à ces critères. La résolution de la requête s'établit alors par l'appel d'une méthode de la classe de l'objet dont le paramètre principal est un ensemble d'éléments de structure parmi lesquels nous cherchons les éléments pertinents.

Dans la figure 8.8, nous donnons la hiérarchie des classes de requêtes pour les requêtes structurelles, les requêtes sur les médias ainsi que les requêtes sur certains des attributs de nos documents. La classe d'objets RequeteSurAtt a la particularité de pouvoir accueillir plusieurs requêtes sur des attributs et elles partagent des propriétés communes à ces requêtes.

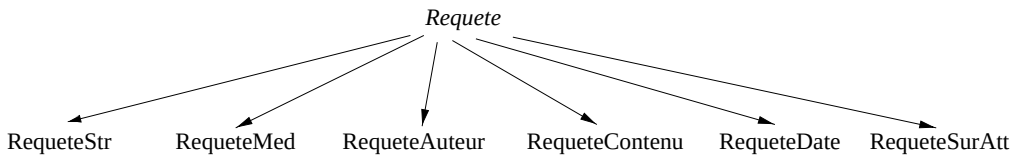



**Figure 8.8.** Hiérarchie des classes de requêtes

## 8.7 Exemples et Résultats

Nous présentons tout d'abord quelques exemples de requêtes puis nous discutons des répercussions possibles de cette application.

### 8.7.1 Résultats obtenus

Nous n'avons pas défini formellement un langage de requêtes mais nous proposons une interface contrôlant la construction de la requête et s'établissant en deux étapes : définition des critères structurels puis ajout des critères sur les attributs en s'adaptant aux critères structurels.

Nous allons montré à partir d'exemples de requêtes d'une part certaines des fonctionnalités que nous avons développées et d'autre part nous cherchons à mettre en évidence l'apport des portées par rapport à une approche traditionnelle.

Considérons la requête suivante: «je recherche un article composé d'au moins 5 paragraphes. Cet article a été écrit par GERARD DUPUY ou par HERVE NATHAN ou par ISABELLE MANDRAU, et il a été publié après le 25/10/97. L'article est indexé par les termes "transports" et "routiers".»

Nous présentons dans la figure 8.9 le résultat de cette requête. Nous identifions quatre parties dans la présentation du résultat:

1. les critères structurels de la requête, c'est-à-dire le type de l'élément recherché, article, ainsi que le type d'éléments qui le compose et le nombre minimum d'apparitions de ce type. Il faut noter qu'il est possible de spécifier une composition directe ou indirecte qui est alors similaire à un chemin d'accès à un élément. Il s'agit ici d'une composition indirecte. Donc les paragraphes peuvent être situés à n'importe quel niveau dans l'arborescence structurelle dont l'article est la racine.

2. les critères sur les attributs de la requête. Notre requête comporte des critères sur trois attributs: Auteur («GERARD DUPUY ou HERVE NATHAN ou ISABELLE MANDRAU»), Date de publication («après le 25/10/97») et Contenu sémantique («"transports" et "routiers"»).

3. des informations sur l'espace des éléments pertinents ainsi que des moyens de navigation pour revenir sur la construction de la requête. Nous indiquons le nombre d'éléments pertinents ainsi que le nombre de journaux qui contiennent ces éléments. Nous offrons aussi le moyen pour l'utilisateur de revenir soit sur la construction de la requête structurelle, soit sur celle de la requête sur les attributs.

4. les éléments pertinents sous forme de lien hypertextuel. L'activation d'un de ces liens permet la consultation d'un élément. Par ailleurs, les éléments sont regroupés par journaux afin d'indiquer leur provenance.

Cette même requête, avec des critères de contenu non plus sur l'article mais sur les paragraphes qui composent les articles, fournit des résultats différents (figure 8.10). Il s'agit ici d'une illustration de l'effet des portées des attributs et du choix de l'opérateur d'union lors de la remontée des termes d'indexation. Les paragraphes sont les éléments feuilles pour l'indexation tandis que les articles contiennent les termes d'indexation qui décrivent tous les paragraphes qui les composent puisque leur attribut de contenu est obtenu par propagation et combinaison des index des paragraphes.

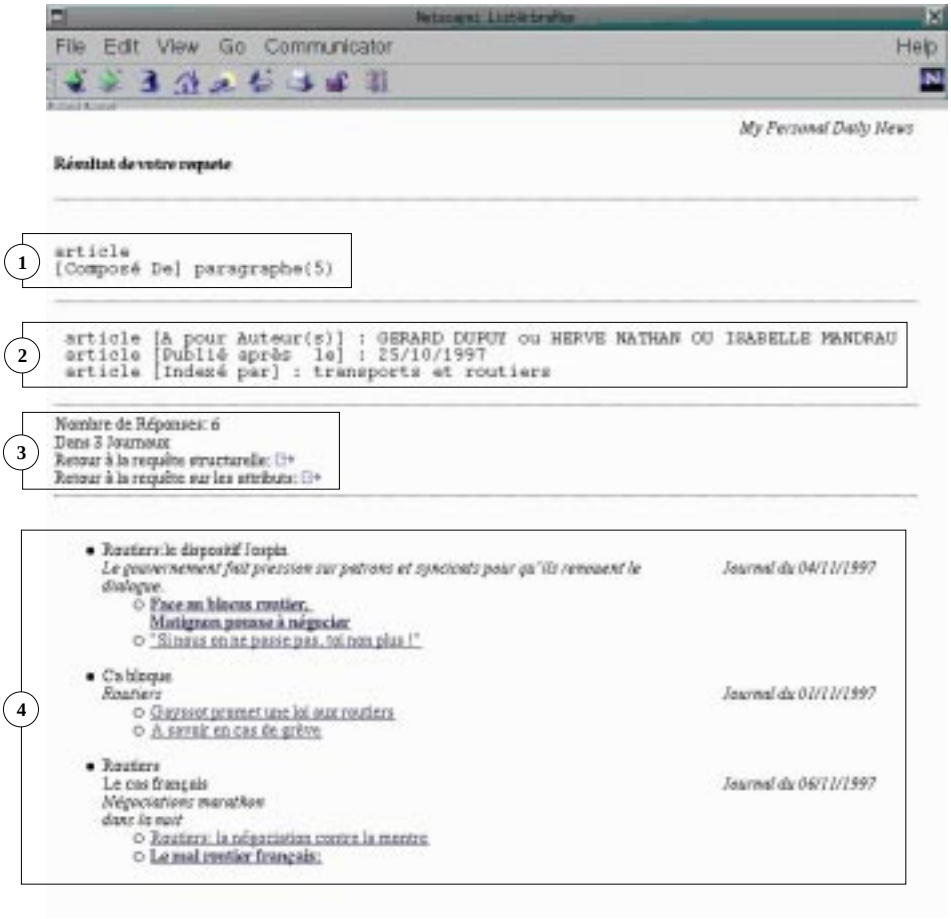



**Figure 8.9.** Résultat d'une requête combinant structure et attributs avec myPDN

A partir des réponses fournies par notre application, il devient possible de consulter chacun des éléments retrouvés et de parcourir ces éléments par l'intermédiaire de liens hypertextuels créés dynamiquement entre ces éléments. Nous proposons ainsi plusieurs modes de consultation pour les éléments retrouvés. Nous identifions dans la figure 8.11 ces principaux modes qui, à partir d'un ensemble de points de départ dans la base c'est-à-dire les éléments retrouvés, autorisent la navigation hypertextuelle dans la base.

1. *accès au journal contenant l'élément retrouvé.*

   L'activation de ce lien donne accès à la page de "une" du journal. Il s'agit, par l'intermédiaire de ce lien, de permettre à l'utilisateur d'identifier le contexte des informations qui lui ont été restituées.

2. *accès à la réponse suivante ou à la réponse précédente de même niveau dans la structure.*

   Le processus de correspondance crée une structure à partir des éléments retrouvés. Cette structure reprend les relations de structure des documents en ne rendant visible que les

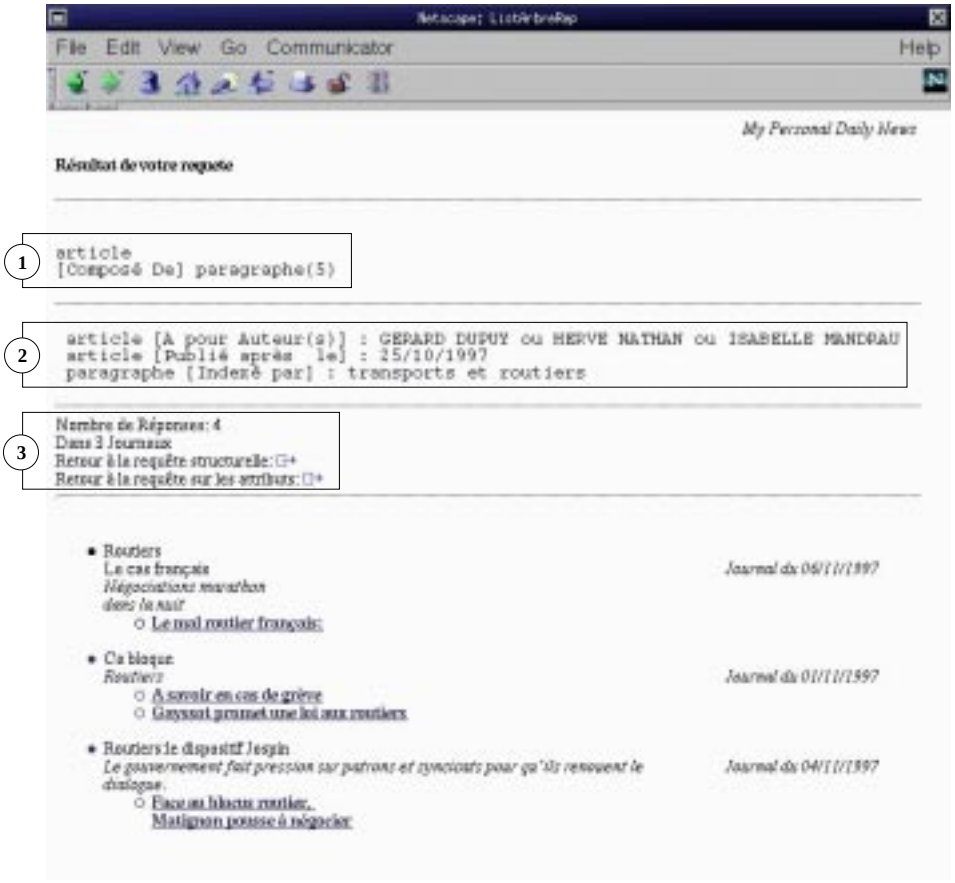



**Figure 8.10.** Requête avec un critère sur l'élément composé

éléments pertinents et les éléments de l'arborescence qui sont des parents de ces éléments pertinents. Il s'agit ici de donner une visibilité sur la position de l'élément retrouvé dans les documents.

3. *retours sur les critères des requêtes structurelles ou sur les critères des requêtes sur les attributs.*

   Ces deux liens permettent à l'utilisateur de retourner sur les interfaces de construction de requête afin de modifier les critères de recherche de celle-ci.

4. *parcours dans la structure des éléments.*

   Ce parcours est identique à celui que nous avons présenté précédemment. Nous permettons à partir des éléments d'accéder à n'importe quel autre élément, même ceux qui ne sont pas pertinents, en suivant les relations de composition et de séquence.

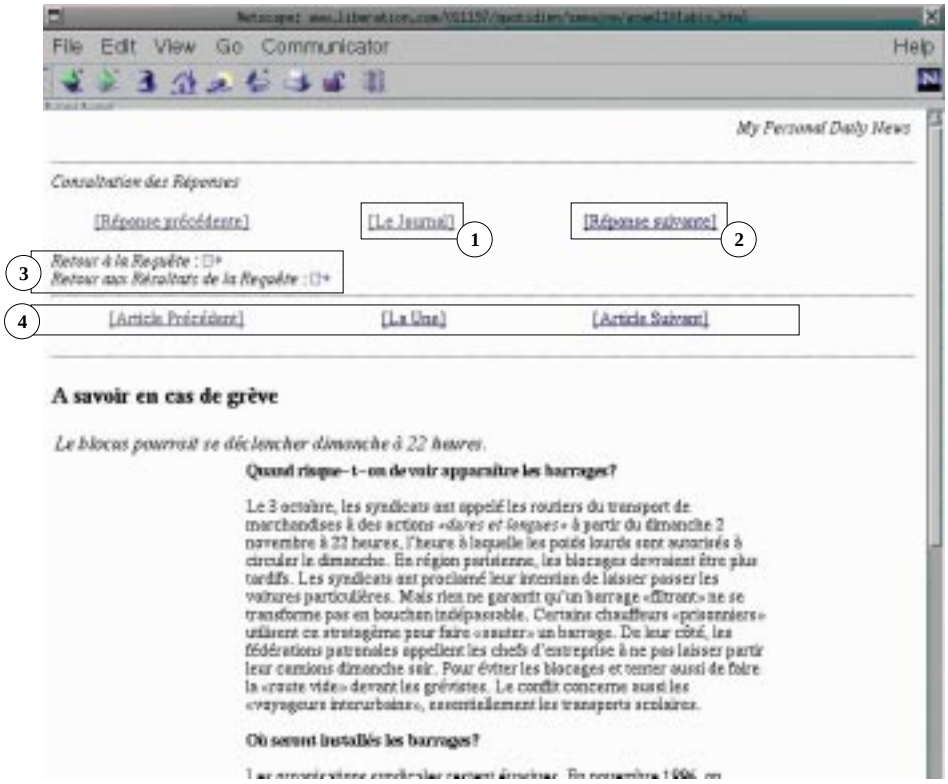



**Figure 8.11.** Consultation des réponses d'une requête combinant structure et attributs avec myPDN

### 8.7.2 Discussion

L'application *my Personal Daily News*, si elle n'est pas complète par certains aspects (notamment l'automatisation du chargement des documents dans la base ou la définition d'un langage de requêtes) met toutefois en évidence certains points délicats que nous allons discuter ici :

*L'usage de la structure* : nous montrons dans notre application qu'il est possible d'utiliser conjointement la structure des documents et leur contenu. L'usage de la structure combiné à celui des portées offre plusieurs avantages. Tout d'abord cela permet de focaliser sur un élément de structure pertinent, c'est-à-dire que soit l'utilisateur choisit la granularité des éléments recherchés, soit le système se charge de retourner les éléments pertinents en indiquant leur granularité respective. Ensuite, l'usage des portées rend accessible des éléments traditionnellement non retrouvés : dans l'exemple 1 de la figure 8.12, nous recherchons «tout élément de structure qui est un composant d'un article et qui est indexé par les termes "routiers" et "accord"». L'élément identifié par "Livre Blanc. Ainsi, en juillet, la Commissi ..." est un paragraphe qui est indexé par ces deux termes alors que l'élément corpsarticle ...4 de l'article intitulé "Si nous on ne passe pas,

toi non plus!" comporte au moins un paragraphe indexé par routiers et un autre indexé par accord, mais aucun indexé par les deux à la fois. C'est l'application de l'opérateur d'union lors de la résolution des conflits sur les portées qui construit un index pour l'élément de type corpsarticle décrit par les deux termes routiers et accord.

*La mise en évidence de l'origine des informations dans la structure* : toujours en reprenant les exemples de la figure 8.12 et en comparant les exemples 1 et 2, nous nous apercevons de l'utilité de la structure d'indexation. Dans l'exemple 1, les éléments feuilles initialement indexés par les termes requis sont au nombre de 2 sur un corpus de 1035 éléments qui sont des composants d'un article. Pour notre part, nous restituons 9 éléments dans 7 articles différents. Dans l'exemple 2, la même recherche est effectuée mais en posant les critères de contenu non plus sur les éléments qui composent l'article mais sur l'article. Nous retrouvons à nouveau les 7 articles ainsi que l'ensemble des composants de ces articles, 123 au total.

A travers ces deux exemples, nous identifions bien la provenance des informations qui sont recherchées et montrons l'importance de l'opérateur de combinaison. Dans ce cas précis, une indexation classique des articles et une recherche avec les deux termes routiers et accord sur ceux-ci auraient fourni les 7 articles que nous avons retrouvés, mais l'utilisateur aurai dû fouiller dans ceux-ci pour trouver les 2 paragraphes plus particulièrement indexés par ces deux termes.

*Une plus grande facilité d'utilisation* : un autre aspect nous semble primordial dans l'étude de notre application : il n'est pas nécessaire d'avoir une connaissance précise de la structure pour interroger les documents. Notre approche n'est pas comparable à celle basée sur l'ajout d'opérateurs dans les langages de requêtes tel que le propose Christophides [Chr96] car notre apport se situe en amont de l'interrogation, lors de la préparation du document. En augmentant la couverture du document par les attributs, nous proposons plus de solutions d'interrogation et donc de plus nombreux moyens d'accès à une même partie de document.

Combiné au point précédent, nous pouvons ainsi interroger la base complète en recherchant n'importe quel élément de structure traitant d'un sujet particulier. L'application restitue alors les éléments pertinents, et cela quel que soit leur position dans la structure, mais en indiquant toujours à l'utilisateur cette position.

A la suite de ce développement, certains éléments n'ont pas été intégrés dans *my Personal Daily News* et doivent désormais être pris en compte pour améliorer les capacités de notre application et la rendre plus généraliste tant en amont qu'en aval de l'interrogation :

*L'application d'un langage d'indexation ayant un pouvoir d'expression plus grand* : dans l'application PRIME-GC, nous avons expérimenté notre approche en utilisant un langage d'indexation basé sur le formalisme des graphes conceptuels. Ce formalisme de représentation de connaissances autorise, entre autres, une représentation des relations entre les termes d'indexation ainsi qu'une classification des termes d'indexation (spécialisation / généralisation). Nous avions alors défini des opérateurs spécifiques pour la combinaison des informations lors de la remontée de l'attribut de contenu. L'usage d'un tel langage d'indexation augmente considérablement la précision des réponses comme nous l'avons démontré dans PRIME-GC [MMFB97, BFMM97].

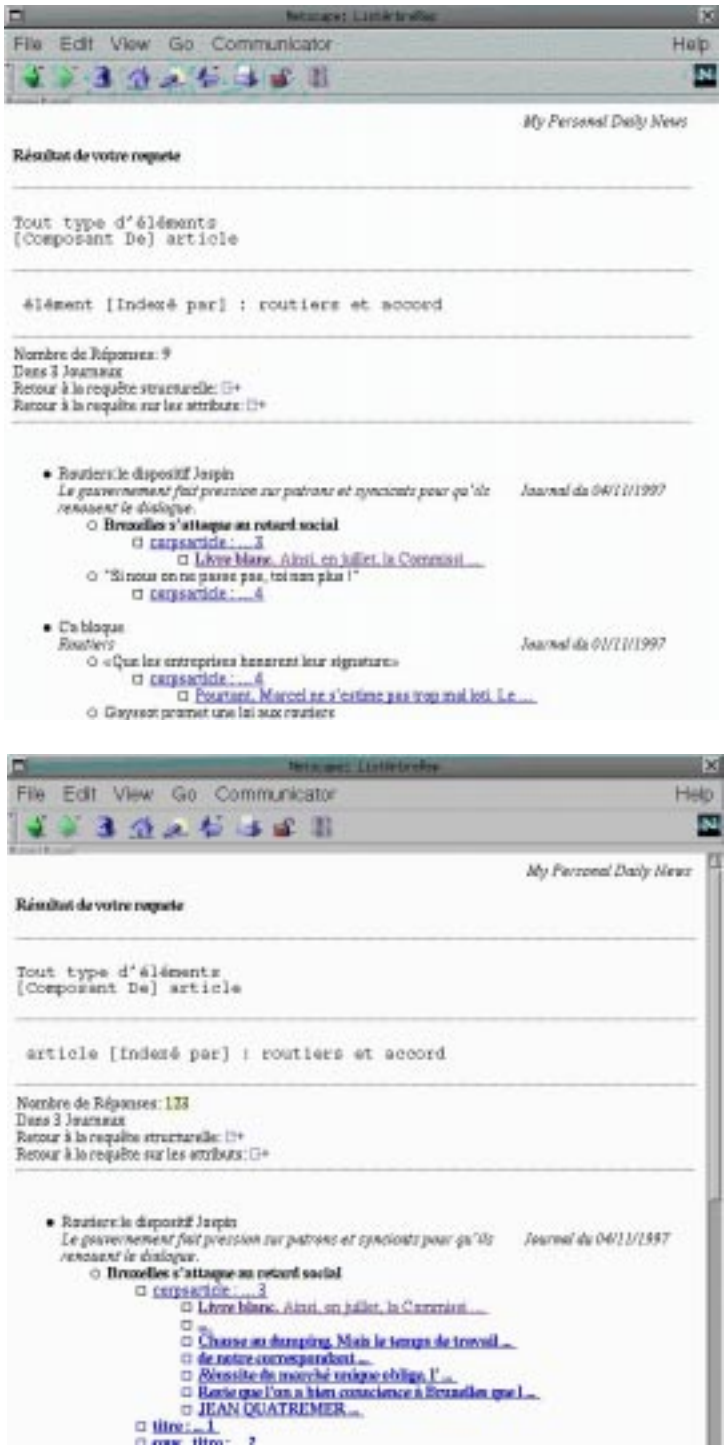



**Figure 8.12.** Deux exemples de résultats avec *myPDN*

Le principal obstacle lié à l'utilisation de ce formalisme réside dans l'obtention des index sous la forme de graphes conceptuels de manière automatique ainsi que la construction de la base canonique contenant les concepts du langage et les relations de spécialisation / généralisation qui les lient. Dans le cadre d'une application dédiée à un domaine particulier comme PRIME-GC avec le domaine médical, le vocabulaire restreint ainsi que des études spécifiques [Ber88] nous ont permis d'automatiser partiellement le processus d'obtention des index sous forme de graphes conceptuels. Pour l'application *my Personal Daily News*, nous n'avons pas été en mesure de fournir ce type d'indexation. De plus le domaine particulièrement large des documents que nous traitons nécessite des outils de construction de graphes conceptuels non encore maîtrisés.

Nous souhaitons toutefois nous orienter vers d'autres langages d'indexation et appliquer notre processus d'indexation structurelle sur ceux-ci, par exemple sur un langage à base de vecteurs de termes fournissant une pondération des réponses.

*Le développement d'une interface d'interrogation plus souple* : l'interface que nous proposons actuellement, basée sur l'enchaînement de formulaires HTML, ne permet pas d'exploiter l'ensemble des fonctionnalités que nous sommes en mesure de proposer. Nous pensons développer une interface graphique à manipulation directe offrant la possibilité de construire des arborescences semblables aux documents recherchés sur lesquelles des critères sur les attributs seraient collés.

*L'automatisation du chargement des documents dans la base* : cet aspect n'a été que très peu abordé dans notre travail car nous avons pris comme postulat que le SGBD fournit la structure d'accueil des documents structurés et que notre travail s'articule autour de la préparation de ces documents pour leur interrogation.

Les méthodes que nous avons adoptées permettent d'accueillir les documents se conformant au type de structure que nous avons défini. Nous n'avons par contre que très peu abordé la problématique liée à la définition de structure générique ni à la vérification syntaxique de ces structures. Pour fournir une application totalement opérationnelle, une étude dans ce sens mérite d'être menée. Elle nécessite la définition de mécanismes génériques de traduction des documents vers leur représentation dans la base [Chr96] et par conséquent une modélisation plus fine au niveau du schéma de la base [CPC94].

# Chapitre 9

# Conclusion

"Imagine a future in which your interface agent can read every newswire and newspaper and catch every TV and radio broadcast on the planet, and then construct a personalized summary."

Nicholas Negroponte - "Being Digital"

## 9.1 Synthèse et Apports

Nous avons présenté un travail sur la modélisation des documents structurés dans le cadre d'un système de recherche d'informations. Nous proposons ainsi un processus d'indexation complémentaire au processus d'indexation classique des SRI. Ce processus est entièrement guidé par les relations de structure et nous l'avons désigné par le terme *processus d'indexation structurelle*.

En premier lieu, nous avons porté notre attention sur la modélisation du document structuré en nous attardant particulièrement sur les relations de structure qui organisent ce type de documents. L'étude des relations de structure du document textuel fait apparaître trois types de relations : la composition, la séquence et la référence. Nous montrons ensuite comment l'image fixe, qui est le second média que nous considérons, peut être modélisée dans un cadre similaire à celui du texte. Le modèle de représentation du document structuré découle alors naturellement d'une uniformisation de ces trois types de relations dans le document structuré.

Pour définir les propriétés du processus d'indexation structurelle, nous sommes partis de deux constats : d'une part les relations de structure sont trop souvent ignorées dans l'indexation des documents et d'autre part, il existe de nombreuses informations au sein du document qui restent implicites. Ces informations "implicites" sont donc difficilement exploitables lors du processus de recherche et constituent un frein dans la recherche du ou des éléments de structure qui répondent à une requête. Nous proposons, dans le processus d'indexation structurelle, d'utiliser nos trois types de relations de structure pour rendre explicites les informations qui demeuraient implicites dans les approches traditionnelles.

Dans notre modèle de document structuré, les informations sont décrites sous la forme d'attributs rattachés aux éléments de structure. Pour parvenir à expliciter les informations implicites, nous avons donc travaillé sur les attributs et leurs propriétés dans le document structuré. Nous les caractérisons en leur attribuant un comportement lié à l'une des relations de structure. Ce comportement se traduit par l'introduction de la notion de portée d'un attribut dans le document structuré. Avec cette notion, nous voulons signifier qu'une information, ici représentée par un attribut, ne concerne pas un unique élément de structure du document, mais qu'elle concerne également d'autres éléments du document. Nous considérons ainsi des éléments de structure qui sont sources d'informations, sur lesquels sont définis des attributs, et également des éléments de structure concernés par ces informations : les éléments atteints par les portées des attributs des éléments sources. D'une part, les éléments atteints par les portées admettent des informations qui, auparavant, restaient implicites. D'autre part, nous générons des dépendances entre les valeurs des attributs en combinant les informations qui proviennent de plusieurs sources. Là encore, il s'agit d'une explicitation des informations décrivant le document.

Grâce à l'introduction et à l'application des portées des attributs, le modèle de correspondance que nous proposons dispose d'un espace d'information plus complet, au sein duquel les informations se sont propagées en suivant les relations de structure liant les éléments. De plus, le modèle de correspondance, malgré ses limites actuelles, est capable de restituer des éléments de structure qui ne pouvaient être atteints par des approches traditionnelles. L'absence de la prise en compte des caractéristiques structurelles lors du processus d'indexation, et plus généralement dans la description du document et de ces composants, s'avère dommageable pour la recherche d'informations. Elle introduisait du silence dans les réponses, dû principalement aux informations qui demeuraient implicites, et elle nécessitait, lors de l'interrogation, une connaissance précise de la structure des documents, Cette connaissance reste aujourd'hui le principal obstacle à une plus grande flexibilité des langages d'interrogation sur les documents structurés.

Notre approche autorise donc une connaissance moindre sur la structure des documents. Il n'est plus nécessaire de savoir quel attribut décrit quel élément puisque cette connaissance est prise en charge simultanément par la définition d'une portée sur l'attribut et par l'indexation structurelle. De même, les requêtes combinant structure et contenu voyaient trop souvent leur champ de recherche restreint par une trop faible couverture du document. Avec notre approche, nous pensons avoir réduit cette lacune récurrente des systèmes traditionnels.

Bien entendu, introduire de nouvelles informations signifie augmenter la masse d'informations à gérer. Nous pensons qu'il est désormais intéressant de travailler sur des techniques appropriées d'optimisation des recherches dans la masse d'informations que nous fournissons. Ce point, délicat à traiter, n'a ici été envisagé que partiellement lors du processus d'interrogation.

Notre travail, axé sur l'usage de la structure des documents pour les modes d'accès à des documents structurés, a aussi démontré "l'utilisabilité" du processus proposé dans le cadre de documents multimédias. Nous avons ainsi montré comment combiner des informations provenant de médias différents à l'aide du processus d'indexation structurelle. Les composants monomédias et multimédias interrogeables sont ainsi rendus accessibles tant par leur contenu que par des attributs provenant d'autres éléments structurels.

L'expérimentation de notre approche sur une base de quotidiens d'informations électroniques a abouti au prototype *my Personal Daily News*. Les quotidiens considérés fournissent des documents qui sont à la fois structurés et multimédias. L'implantation, basée sur le système à objets $O^2$, nous a permis de valider l'intérêt du processus d'indexation que nous avons élaboré ainsi que les avantages significatifs qui peuvent en être tirés. Ce prototype représente pour nous un premier pas vers l'élaboration d'un système plus complet dans lequel le processus d'indexation structurelle serait automatisé.

## 9.2 Problèmes ouverts

Dans le travail que nous avons présenté, il reste plusieurs problèmes ouverts sur lesquels nous revenons ici. Tout d'abord les fonctionnalités d'interrogation qui peuvent être associées à notre modèle de représentation, mais aussi l'intégration possible de nouvelles relations de structure.

*De nouvelles fonctionnalités d'interrogation?*

Les fonctionnalités de recherche des modèles et systèmes existants tendent vers une plus grande flexibilité. Nous pensons, pour notre part, avoir contribué dans ce sens mais nous estimons aussi que certains aspects n'ont pas été couverts par notre travail.

Le mode de correspondance que nous proposons connaît actuellement certaines limites. Par exemple, nous ne proposons pas de pondération sur les termes d'indexation et nous posons des hypothèses fortes sur les fonctions de propagation. Il nous semble intéressant de compléter les fonctionnalités d'interrogation que nous proposons en traitant ces points. De même, nous pensons qu'il est possible d'intégrer dans notre modèle une notion d'incertitude en utilisant des théories existantes, par exemple la théorie de Dempster-Shafer [Lal97].

De plus, nous pensons qu'il nous faut désormais nous intéresser à des techniques d'optimisation de la gestion des informations décrivant le document structuré afin de diminuer les temps de réponses, spécialement si nous voulons confronter notre modèle à de grandes quantités de documents structurés.

*Quelles nouvelles relations de structure?*

Dans notre étude, nous avons considéré trois types de relations de structure pour établir notre modèle de document structuré. Pour élargir cette étude et accueillir naturellement des documents hypertextuels, il serait intéressant de caractériser précisément les spécificités des relations de ce type de documents. Une première analyse nous laisse penser que notre modèle est capable de représenter des documents hypertextuels en considérant les relations de navigation sous la forme de relations de référence.

Cependant, il semble préférable de mener une étude sur ces relations afin de déterminer leur véritable influence dans les documents hypertextuels : permettent-elles de propager des informations? Si oui, selon quels modes? L'étude aurait donc pour objectif de préciser l'impact de leur prise en compte sur notre classification d'attributs et de déterminer quelles sont les modifications à apporter, s'il y a lieu, sur cette classification pour accueillir ces nouvelles relations.

L'étude de nouvelles fonctionnalités d'interrogation et l'étude complémentaire de nouvelles

relations de structure ainsi que de leurs comportements dans notre modèle d'indexation nous semblent les deux points les plus importants à approfondir encore dans notre travail.

## 9.3 Perspectives

Le modèle de document et le processus d'indexation structurelle que nous avons proposés dans cette thèse ouvrent selon nous des perspectives intéressantes tant sur le plan pratique que sur le plan théorique.

*Vers l'accueil de nouveaux médias*

Dans le cadre d'applications multimédias, les données temporelles sont désormais largement utilisées mais peu d'outils offrent les fonctionnalités nécessaires à leur recherche. Les travaux que nous avons étudiés sur ces données [SW95, DK95], principalement sur les séquences vidéos, tendent vers un consensus général pour représenter les données temporelles de manière structurée sous la forme d'une arborescence. Une vidéo est composée de séquences qui sont elles-mêmes composées de plans, etc. La différence la plus notable par rapport à la structure que nous proposons, pour les médias statiques que nous avons considérés, provient de l'introduction de la notion de temps.

L'accueil de ce type de données dans notre modèle signifie qu'il faut étudier les caractéristiques des relations temporelles, proposer une représentation de ces relations ainsi que leur réel impact sur les informations qui décrivent le document. Dans notre travail, nous avons décrit différents types de comportement des informations à l'aide des trois relations de structure de notre modèle. Il nous semble que, de manière similaire, les relations temporelles sont un support pour propager et pour combiner les informations entre les noeuds de la structure du document.

Il réside toutefois des difficultés supplémentaires liées à la cohabitation dans un document multimédia de données statiques et temporelles. Comment les informations peuvent-elles se comporter dans de tels documents? Quelles extensions à notre modèle sont alors nécessaires pour accueillir ces nouvelles données et ces nouveaux comportements?

*Vers de nouvelles formes de réponses*

Enfin, nous voulons conclure sur la forme des réponses des systèmes actuels. Les systèmes de recherche d'informations proposent d'aider et d'assister l'utilisateur pour qu'il exprime ses besoins et qu'il retrouve des éléments d'informations qui comblent ses besoins. Ces systèmes fournissent les documents pertinents à la requête, sous forme de liste. Les documents renvoyés sont exactement ceux qui ont été analysés par le système.

Fournir les documents originaux comme réponse est satisfaisant dans de nombreux cas (comme dans le domaine médical par exemple où les composants d'un document sont fortement connectés), mais dans le cas où une même information est présentée suivant des aspects différents (par exemple dans le cas de quotidiens d'information électronique où une même information peut être obtenue depuis plusieurs sources), cette approche peut être remise en cause. Dans ce cas, l'utilisation de documents virtuels peut être une solution en proposant des réponses mieux adaptées au besoin de l'utilisateur, en regroupant et organisant des parties de documents du corpus. Un document virtuel, dans ce cas, se définit comme une agrégation

de parcelles d'information basée sur des règles de construction syntaxiques et sémantiques. Un document virtuel est un donc document qui n'existe que par l'expression des besoins d'un utilisateur. Ce document n'existe pas physiquement dans la base de documents mais il est construit par le système pour remplir les besoins de l'utilisateur. Il s'agit ici de modifier la forme des réponses pour ne plus proposer une liste de documents ou de composants pertinents que l'utilisateur doit consulter un à un. Nous pensons que notre modèle, dont les relations de structure servent de support pour propager les informations, peut être un point de départ intéressant pour l'élaboration de règles de construction syntaxique et sémantique des documents virtuels.

Récemment encore, la qualité des SRI se jugeait au travers de mesures ne laissant que peu de place à l'utilisateur. Des travaux récents [Den97] tendent à prouver que l'utilisateur doit être replacé au centre du système et qu'il doit donc diriger le processus de recherche. Nous pensons pour notre part, que cette approche centrée sur l'utilisateur prend tout son sens avec l'introduction des documents complexes contenant des données hétérogènes et multimédias. Le concept de document virtuel va dans ce sens et nous souhaiterions le développer plus concrètement.

# Bibliographie

[ABL95] Ahanger (Gulrukh), Benson (Dan) et Little (T.D.C.). – Video query formulation. *In : Storage and Retrieval for Images and Video Databases II*. IS&T/SPIE. – San Jose, CA USA, February 1995.

[ABV95] Aberer (Karl), Böhm (Klemens) et Volz (Marc). – Providing information retrieval and database mechanisms for structured documents. *In : SIGIR'95*. ACM. – Seattle, USA, July 1995. position paper on the research workshop "Information retrieval and Databases".

[AC95] Analyti (Anastasia) et Christodoulakis. (Stavros). – Multimedia object modelling and content-based querying. *In : Advanced Course Multimedia Database in Perspective*, pp. 213–240. – Enschede, The Netherlands, June 1995.

[ACC+97] Abiteboul (Serge), Cluet (Sophie), Christophides (Vassilis), Milo (Tova), Moerkotte (Guido) et Simeon (Jerom). – Querying structured documents in object databases. *International Journal on Digital Libraries*, vol. 1, n° 1, April 1997, pp. 5–19.

[ACG91] Agosti (M.), Colotti (R.) et Gradenigo (G.). – A two-level hypertext retrieval model for legal data. *ACM*, 1991, pp. 316–325.

[ADHC94] Arman (F.), Depommier (R.), Hsu (A.) et Chiu (M-Y.). – Content-based browsing of video sequences. *In : ACM Multimedia 94*. ACM, pp. 97–103. – San Fransisco, USA, October 1994.

[AK92] Aberer (K.) et Klas (W.). – *The Impact of Multimedia Data on Database Management*. – Rapport technique n° ICSI-TR-92-065, Berkeley, International Computer Science Institute, 1992.

[All83] Allen (J. F.). – Maintaining knowledge about temporal intervals. *Communication of the ACM*, vol. 26, 1983, pp. 832–843.

[AMC95] Agosti (M.), Melucci (M.) et Crestani (F.). – Automatic construction of hypermedia for information retrieval. *Multimedia Systems*, 1995.

[Ass91] Association (Danish Standards). – *SGML-ODA, Présentation des concepts et comparaison fonctionnelle*. – afnor technique, 1991.

[BCK+94] Blake (G.E.), Consens (M.P.), Kilpelinen (P.), Larson (P-A.), Snider (T.) et Tompa (F.W.). – Text / relational database management systems: Harmonizing

sql and sgml. *In : Applications of Databases (ADB'94)*, pp. 267–280. – Vadstena, June 1994.

[Ber88] Berrut (Catherine). – *Une méthode d'indexation fondée sur l'analyse sémantique de documents spécialisés. Le prototype RIME et son application à un corpus médical.* – Thèse de PhD, Université Joseph Fourier, 1988.

[BFM+97] Berrut (Catherine), Fourel (Franck), Mechkour (Mourad), Mulhem (Philippe) et Chiaramella (Yves). – *Indexing, navigation and retrieval of multimedia structured documents : the PRIME information retrieval system.* – Chapter of deliverable d11, FERMI ESPRIT BRA 8134, 1997.

[BFMM97] Berrut (Catherine), Fourel (Franck), Mechkour (Mourad) et Mulhem (Philippe). – Indexation, navigation et recherche de documents structurés multimédia sur o2 : le système de recherche d'informations prime appliqué à un un corpus médical. *In : Conférence O2 dans le cadre du congrès Bases de Données Avancées 1997 (BDA'97).* – Grenoble, Septembre 1997.

[BGMS93] Berra (P.B.), Golshani (F.), Mehrotra (R.) et Sheng (O.R. Liu). – Guest editors' introduction multimedia information systems. *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, n° 4, August 1993.

[Bla90] Blair (D.C.). – *Language and Representation in Information Retrieval.* – Elsevier, 1990.

[Bla95] Blanken (H.). – Sql3 and multimedia applications. *In : Advanced Course Multimedia Database in Perspective*, pp. 155–178. – Enschede, The Netherlands, June 1995.

[Bla98] Blair (D.C.). – An extended relational document retrieval model. *Information Processing & Management*, vol. 24, n° 3, 1998.

[BMB95] Berrut (Catherine), Mulhem (Philippe) et Bouchon (Pascal). – Modelling and indexing medical images : the rime approach. *In : HIM 95 (Conference Hypertext, Information Retrieval, Multimedia), Constance, Allemagne*, pp. 105–115.

[BMN94] Böhm (K.), Müller (A.) et Neuhold (E.). – *Sructured Document Handling - a Case for Integrating Databases and Information Retrieval.* – Rapport technique, Darmstatd, Germany, GMD-IPSI, 1994.

[Bou95] Bouchon (Pascal). – *Indexation d'objets complexes multimedia : application aux dossiers medicaux.* – mémoire d'ingénieur CNAM n° RAP95-002, Groupe MRIM – LGI-IMAG, 1995.

[BR90] Bertino (Elisa) et Rabitti (Fausto). – *The MULTOS Query Language*, chap. 4, pp. 53–74. – North-Holland, 1990.

[BR94] Böhm (K.) et Rakow (T.C.). – Metadata for multimedia documents. *SIGMOD RECORD*, vol. 23, n° 4, December 1994.

[BRG88] Bertino (Elisa), Rabbiti (Fausto) et Gibbs (Simon). – Query processing in a multimedia document system. *Transactions on Information Systems*, vol. 6, n° 1, jan 1988, pp. 1–41.

[Bru87] Bruandet (Marie-France). – Outline of a knowledge base model for an intelligent information retrieval system. *In : ACM-SIGIR, New Orleans*, pp. 33–43.

[Bru93] Bruza (P.D.). – *Stratified Information Disclosure, a Synthesis between Hypermedia and Information Retrieval.* – Thèse de PhD, Katholieke Universiteit Nijmegen, March 1993.

[Bur91] Burkowski (Forbes J.). – the use of retrieval filters to localize information in a hierarchically tagged text-dominated database. *In : RIAO 91.* – Barcelonne, Espagne, April 1991.

[Bur92] Burkowski (Forbes J.). – Retrieval activities in a database consisting of heterogeneous collection of structured text. *In : Proceedings of ACM SIGIR'92.* ACM, pp. 112–125. – Denmark, jun 1992.

[Bur94] Burnard (Lou). – What is SGML and does it help ? *In : ACM SIGIR, tutorial material.* – Dublin, Ireland, July 1994.

[Bus45] Bush (Vannevar). – As we may think. *The Atlantic Monthly*, July 1945.

[Cal94] Callan (James P.). – Passage-level evidence in document retrieval. *In : Proceedings of ACM SIGIR'94*, éd. par Croft (W.B.) et van Rijsbergen (C.J.). ACM, pp. 302–310. – Dublin, Ireland, July 1994.

[CCB94] Clarke (C.L.A.), Cormack (G.V.) et Burkowski (F.J.). – *An Algebra for Structured Text Search and a Framework for its Implementation.* – Rapport technique n° CS-94-30, University of Waterloo, December 1994.

[CCB95] Clarke (C.L.A.), Cormack (G.V.) et Burkowski (F.J.). – An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 1995.

[CDBK86] Chiaramella (Y.), Defude (B.), Bruandet (M.F.) et Kerkouba (D.). – Iota: a full test information retrieval system. *In : ACM conference on research and development in information retrieval., Pisa, Italy*, pp. 207–213.

[CDDA88] Chang (S.K.), Donald (C.W.Y.), Dimitroff (C.) et Arndt (T.). – An intelligent image database system. *IEEE Transactions on Software Endineering*, vol. 14, n° 5, May 1988.

[CDY90] Constantopoulos (P.), Drakopoulos (Y.) et Yeorgaroudakis (Y.). – *Multimedia Document Retrieval by Pictorial Content*, chap. 8.2, pp. 331–349. – North-Holland, 1990.

[CFM96] Chiaramella (Yves), Fourel (Franck) et Mulhem (Philippe). – *Modelling Multimedia Structured Documents.* – Chapter of deliverable d4, FERMI BRA 8134, 1996.

[Che92] Chevallet (Jean-Pierre). – *Un Modèle Logique de Recherche d'Informations appliqué au formalisme des Graphes Conceptuels. Le prototype ELEN et son expérimentation sur un corpus de composants logiciels.* – Thèse de PhD, Université Joseph Fourier, Grenoble, 1992.

[CHN+95] Cody (W.F.), Haas (L.M.), Niblack (W.), M. Arya (M.J. Carey), Ragin (F.), Flickner (M.), Lee (D.), Petrovic (D.), Schwarz (P.M.), Thomas (J.), Roth (M. Tork), Williams (J.H.) et Wimmers (E.L.). – Querying multimedia data from multiple repositories by content: the garlic project. *In : IFIP 2.6 Third Working Conference on Visual Database Systems (VDB-3).* – Lausanne, Switzerland, March 1995.

[Chr96] Christophides (Vassilis). – *Documents Structurés et Bases de Données Objet.* – Thèse de PhD, Conservatoire National des Arts et Métiers, October 1996.

[CK96] Chiaramella (Yves) et Kheirbek (Amar). – An integrated model for hypermedia and information retrieval. *In : Information Retrieval and Hypertext.* – M. Agosti and A. Smeaton (Ed), Kluwer Academic (Publ). 1996., 1996.

[CM97] Chiaramella (Yves) et Mechkour (Mourad). – *Indexing an Image Test Collection.* – Rapport technique, FERMI-BRA Project, Chapter of the Delivrable D10, April 1997.

[CPC94] Comparot-Poussier (Catherine) et Chrisment (Claude). – Hyperbase pour la gestion electronique de documents techniques. *Ingéniérie des Systèmes d'Information*, vol. 2, n° 5, 1994, pp. 533–570.

[CR90] Conti (P.) et Rabitti (F.). – *Image Retrieval by Semantic Content*, chap. 8.1, pp. 299–329. – North-Holland, 1990.

[CR94] Christophides (V.) et Rizk (A.). – Querying structured documents with hypertext links using oodms. *In : ECHT '94.*

[Cra81] Crawford (R.G.). – The relationnal model in information retrieval. *American Society for Information Science*, vol. 32, n° 1, January 1981.

[DBM97] Denos (Nathalie), Berrut (Catherine) et Mechkour (Mourad). – An image retrieval system based on the visualization of system relevance via documents. *In : Database and Expert Systems Applications (DEXA), Toulouse, France*, pp. 214–224.

[DD94] DeRose (S.J.) et Durand (D.G.). – *Making Hypermedia Work - A user's guide to HyTime.* – Kluwer Academis Publishers, 1994.

[Def86] Defude (Bruno). – *Etude et réalisation d'un système intelligent de recherche d'informations: Le prototype IOTA.* – Thèse de PhD, Institut National Polytechnique de Grenoble, 1986.

[Den97] Denos (Nathalie). – *Modélisation de la pertinence en recherche d'information - modèle conceptuel, formalisation et application.* – Thèse de PhD, Université Joseph Fourier, October 1997.

[DK95] Duda (Andrzej) et Keramane (Chérif). – Structured temporal composition of multimedia data. *In : International Workshop on Multi-Media Database Management System*. IEEE, pp. 136–142. – New York, USA, August 1995.

[Dup95] Dupoirier (Gérard). – *Technologie de la GED*. – Paris, France, Herès, 1995, 2nd édition.

[Erf94] Erfle (Robert). – Hytime as the multimedia document model of choice. *IEEE*, 1994, pp. 445–454.

[Fal95] Faloutsos (Christos). – Indexing of multimedia databases. *In : Advanced Course Multimedia Database in Perspective*, pp. 239–278. – Enschede, The Netherlands, June 1995.

[FFE96] François (Patricia), Futtersack (Philippe) et Espert (Christophe). – Une structure d'accueil de documents sgml/hytime basée sur les technologies objets. *In : Journée O2 Technology, XIVe Congrès INFORSID*. – Bordeaux, June 1996.

[FM96] Fourel (Franck) et Mulhem (Philippe). – Modeling multimedia structured documents: A retrieval oriented approach. *In : DEXA 96 workshop, Zurich, Suisse*.

[FMB97] Fourel (Franck), Mulhem (Philippe) et Bruandet (Marie-France). – Dynamic properties of attributes for structured documents access. *In : International Symposium on Digital Media Information Base (DMIB'97), Nara, Japan*. – DMIB'97 Copyright.

[Fou96] Fourel (Franck). – Intégration de la structure du document dans le processus de recherche d'informations. *In : XIVe Congrès INFORSID*, pp. 249–267. – Bordeaux, France, Juin 1996.

[Fou97] Fourel (Franck). – Impact de la structure sur la recherche d'informations. *Ingéniérie des Systèmes d'Information*, vol. 5, n° 3, August 1997, pp. 339–365.

[FSN+95] Flickner (Myron), Sawhney (Harpreet), Niblack (Wayne), Ashley (Jonathan), Huang (Qian), Dom (Byron), Gorkani (Monika), Hafner (Jim), Lee (Denis), Petkovic (Dragutin), Steele (David) et Yanker (Peter). – Query by image and video content: The qbic system. *IEEE Computer*, vol. 28, n° 9, September 1995, pp. 23–32.

[Gha95] Ghafoor (Arif). – Special issue on multimedia database systems, introduction. *Multimedia Systems*, vol. 3, n° 5, November 1995, pp. 179–181.

[GT87] Gonnet (G.H.) et Tompa (F.W.). – Mind your grammar: a new approach to modelling text. *In : 13th VLDB Conference*.

[Hea94] Hearst (M.A.). – *Contextualizing Retrieval of Full-Length Document*. – Rapport technique n° UCB/CSD 94/789, Berkeley, January 1994.

[Her90] Herwijnen (Eric Van). – *Practical SGML*. – Kluwer, 1990.

[HK95] Hirzalla (N.) et Karmouch (A.). – A multimedia query specification language. *In : International Workshop on Multi-Media Database Management System*. IEEE. – New York, USA, August 1995.

[HP93] Hearst (M.A.) et Plaunt (C.). – Subtopic structuring for full-length access. *In : ACM SIGIR'93.* – Pittsburgh, USA, June 1993.

[HSD97] Harmandas (V.), Sanderson (M.) et Dunlop (M.D.). – Image retrieval by hypertext links. *In : Proceedings of ACM SIGIR'97.* ACM, pp. 296–303. – Philadelphia PA, USA, July 1997.

[ISO89] ISO. – *Information Technology-Database Language SQL.* – Rapport technique, ISO/IEC, April 1989.

[KA95] Klas (W.) et Aberer (K.). – Multimedia applications and their implications on database architecture. *In : Advanced Course Multimedia Database in Perspective*, pp. 19–72. – Enschede, The Netherlands, June 1995.

[KC95] Kheirbek (Ammar) et Chiaramella (Yves). – Integrating hypermedia and information retrieval with conceptual graphs. *In : HIM'95, Konstanz, Germany*, pp. 47–60.

[Ker84] Kerkouba (Dalila). – *Une méthode d'indexation automatique des documents fondée sur l'exploitation de leurs propriétés structurelles. Application à un corpus technique.* – Thèse de PhD, Institut National Polytechnique de Grenoble, 1984.

[Khe95] Kheirbek (Ammar). – *Modèle d'intégration d'un système de recherche d'informations et d'un système hypermédia basé sur le formalisme des graphes conceptuels. Application au système RIME.* – Thèse de PhD, Université Joseph Fourier, 1995.

[Kim93] Kimber (W. E.). – *HyTime and SGML Understanding the HyTime HyQ Query Language 1.1.* – Rapport technique, IBM, August 1993.

[KKT94] Kiyoki (Y.), Kitagawa (T.) et T.Hayama. – A metadatabase system for semantic image search by a mathematical model of meaning. *ACM - SIGMOD Record*, vol. 23, n° 4, December 1994.

[KSP+95] Kunii (T.L.), Shinagawa (Y.), Paul (R.M.), Khan (M.F.) et Khokhar (A.A.). – Issues in storage and retrieval of multimedia data. *Multimedia Systems*, vol. 3, n° 5, November 1995, pp. 298–304.

[KXW95] Kankanhalli (Mohan S.), Xunda (Jiang) et Wu (Jiankang). – A spatial algebra for content-based retrieval. *In : Asia Pacific Conference on Image Processing.*

[Lal97] Lalmas (Mounia). – Dempster-shafer's theory of evidence applied to structured documents: modelling uncertainty. *In : Proceedings of ACM SIGIR'97.* ACM, pp. 110–118. – Philadelphia PA, USA, July 1997.

[Loe94] Loeffen (A.). – Text databases: A survey of texts models and systems. *ACM SIGMOD Records*, vol. 23, 1994.

[LYYB96] Lee (Yong Kyu), Yoo (Seong-Joon), Yoon (Kyoungro) et Berra (P. Bruce). – Index structures for structured documents. *In : Digital Libraries 96.* ACM, pp. 91–99. – Bethesda, MD USA, March 1996.

[Mac81] Macleod (I.A.). – A database management system for document retrieval applications. *Information Processing & Managament*, vol. 6, n° 2, 1981, pp. 131–137.

[Mac90] Macleod (I.A.). – Storage and retrieval of structured documents. *Information Processing & Management*, vol. 26, n° 2, 1990, pp. 197–208.

[Mac91] Macleod (I.A.). – A query language for retrieving information from hierarchical text structures. *Computer Journal*, vol. 34, n° 3, 1991, pp. 214–222.

[Mec95a] Mechkour (Mourad). – A conceptual graphics model for information retrieval. *In : Work Part 3 Deliverable: A Model for the Semantic Content of Multimedia Data*, chap. 3. – ESPRIT BRA Project No. 8134 - FERMI, 1995.

[Mec95b] Mechkour (Mourad). – Emir2. an extended model for image representation and retrieval. *In : DEXA '95, Database and EXpert system Applications*, pp. 395–404. – London, September 1995.

[Mec95c] Mechkour (Mourad). – *EMIR2. Un Modèle étendu de représentation et de correspondance d'images pour la recherche d'informations. Application à un corpus d'images historiques.* – Thèse de PhD, Université Joseph Fourier, 1995.

[Mec95d] Mechkour (Mourad). – A multifacet formal image model for information retrieval. *In : MIRO final workshop, Glasgow, UK*, pp. 1–12.

[Meg95] Meghini (Carlo). – A model for image bases and its query facility. *In : ACM SIGIR.* – Seattle, USA, 1995.

[MMFB97] Mechkour (Mourad), Mulhem (Philippe), Fourel (Franck) et Berrut (Catherine). – PRIME-GC: a Medical Information Retrieval Prototype on the WEB. *In : RIDE'97. Seventh International Workshop on Research Issues in Data Engineering: High performance Database Management for Large Scale Applications.* – Birmingham, England, April 7–8 1997.

[MMR95] Maitre (Jacques Le), Murisasco (Elisabeth) et Rolbert (Monique). – SgmlQL, un langage d'interrogation de documents sgml. *In : Actes de la conférence Bases de Données Avancées (BDA)*, pp. 431–446.

[MP96] Minka (Thomas P.) et Picard (Rozalind W.). – *interactive learning using a "society of models".* – Rapport technique n° 349, M.I.T Media Laboratory Perceptual Computing, 1996.

[Nar95] Narasimhalu (A.Desai). – Special section on content-based retrieval - introduction. *Multimedia Systems*, vol. 3, n° 1, February 1995, pp. 1–2.

[Nav95] Navarro (Gonzalo). – *A Language for Queries on Structure and Contents of Textual Database.* – Santiago, Chile, Thèse de PhD, University of Chile, April 1995.

[NBY95a] Navarro (Gonzalo) et Baeza-Yates (Ricardo). – Expressive power of a new model for structured text databases. *In : PANEL'95*, éd. par Mauro (José) et de Castilho (Volkmer), pp. 1151–1162.

[NBY95b] Navarro (Gonzalo) et Baeza-Yates (Ricardo). – A language for queries on structure and contents of textual database. *In : ACM SIGIR.* – Seattle, USA, July 1995.

[NBY96] Navarro (Gonzalo) et Baeza-Yates (Ricardo). – Integrating contents and structure in text retrieval. *ACM SIGMOD Record*, March 1996.

[NF93] Niblack (W.) et Flickner (M.). – Find me the pictures that look like this: Ibm's image query project. *In : Advanced Imaging.*

[NFF+93] Niblack (W.), Faloutsos (C.), Flickner (M.), Petkovic (D.), Equitz (W.) et Barber (R.). – *Efficient and Effective Querying by Image Content.* – Rapport technique n° RJ 9453, IBM, Computer Science, August 1993.

[Nie90] Nie (Jianyun). – *Un modèle logique général pour les Systèmes de Recherche d'Informations. Application au prototype RIME.* – Thèse de PhD, Université Joseph Fourier, 1990.

[OAP97] Ounis (Iadh), Amati (Gianni) et Pasca (Marius). – *A polynomial matching algorithm for conceptual graphs.* – Rapport technique, CLIPS-MRIM, July 1997.

[ODM93] ODMG (Object Data Management Group). – *The Object Database Standard: Release 1.1.* – Morgan Kaufmann Publishers, San Fransisco, California, 1993.

[OP97a] Ounis (Iadh) et Pasca (Marius). – An extended inverted file approach for information retrieval. *In : International Database Engineering and Application Symposium (IDEAS'97).* – Montreal, Canada, August 1997.

[OP97b] Ounis (Iadh) et Pasca (Marius). – The relief retrieval system. *In : The IEEE Knowledge and Data Engineering Exchange Conference (KDEX-97).* IEEE. – Newport Beach, California, USA, November 1997.

[OS95] Ogle (Virginia E.) et Stonebraker (Michael). – Chabot: Retrieval from a relationnal database of images. *IEEE Computer*, vol. 28, n° 9, September 1995, pp. 40–48.

[Ozk95] Ozkarahan (E.). – Multimedia document retrieval. *Information processing & Management*, vol. 31, n° 1, 1995, pp. 113–131.

[Par96] Paradis (François). – *Un modèle d'indexation pour les documents textuels structurés.* – Thèse de PhD, Université Joseph Fourier, 1996.

[PB96] Paradis (François) et Berrut (Catherine). – Experiments with theme extraction in explanatory texts. *In : Second International Conference on Conceptions of Library and Information (CoLIS 2), Copenhagen, Denmark*, pp. 433–446.

[Pep95] Pepper (S.). – The whirlwind guide: Sgml tools and vendors. – Falch Infotek A/S, January 1995. ftp://ftp.falch.no/pub/SGML-Tools.

[Pic96] Picard (Rozalind W.). – A society of models for video and image libraries. *Submitted for Publication, IBM Systems Journal*, 1996.

[PM95] Picard (R.W.) et Minka (T.P.). – Vision texture for annotation. *Multimedia Systems*, vol. 3, n° 1, February 1995, pp. 3–14.

[PPS94] Pentland (A.), Picard (R.W.) et Sclaroff (S.). – Photobook: Tools for content-based manipulation of image database. *In : SPIE Conference on Storage and Retrieval of Image and Video Databases II.* – San Jose, USA, February 1994.

[Pre96] Prescod (Paul). – *Introduction to DSSSL.* – Rapport technique, http://itrc.uwaterloo.ca:80/ papresco/dsssl/tutorial.html, ITRC - University of Waterloo, 1996.

[Pri93] Price (Roger). – Mheg: An introduction to the future international standard for hypermedia objct interchange. *In : ACM Multimedia 93.* ACM, pp. 121–128. – USA, June 1993.

[RF96] Rölleke (Thomas) et Fuhr (Norbert). – Retrieval of complex objects using a four-valued logic. *In : Proceedings of ACM-SIGIR Annual International Conference on Research and Developments in Information Retrieval.* ACM. – Zurich, August 1996.

[Rij79] Rijsbergen (C.J. Van). – *Information Retrieval.* – Butterworths, 1979, second edition édition.

[RNL95] Rakow (T.C.), Neuhold (E.J.) et Löhr (M.). – *Multimedia Databases Systems - The Notions and the Issues.* – Rapport technique, GMD-IPSI, February 1995.

[RvOHB97] Rutledge (Lloyd), van Ossenbruggen (Jacco), Hardman (Lynda) et Bulterman (Dick C. A.). – A framework for generating adaptable hypermedia documents. *In : ACM Multimedia 97.* ACM, pp. 121–130.

[Sal71] Salton (Gerard). – *The SMART project.* – Prentice Hall, 1971.

[San94] Sandoval (Victor). – *SGML : un outil pour la gestion électronique de documents.* – Paris, Hermes, 1994.

[Sav90] Savino (Pasquale). – *Text Retrieval Techniques in Multos*, chap. 7.4, pp. 215–249. – North-Holland, 1990.

[SG83] Salton (G.) et Gill (M.J. Mc). – *Introduction to Moderm Information Retrieval.* – New-York, McGraw-Hill, 1983.

[Sow84] Sowa (J.F.). – *Conceptual Structures : Information Processing in Mind and Machine.* – Addison-Wesley, 1984.

[SS96] Straccia (U.) et Sebastiani (F.). – *The FERMI logic.* – Part I of the deliverable D2 : A logic for information retrieval, Basic Research Action FERMI, n. 8134, 1996.

[SSL+83] Stonebraker (M.), Stettner (H.), Lynn (N.), Kalassh (J.) et Guttman (A.). – Document processing in a relational database system. *ACM Transaction on Information System*, vol. 1, n° 2, 1983.

[ST94] Salminen (A.) et Tompa (F.W.). – Pat expressions: an algebra for text search. *In : Acta Linguista Hungarica41*, pp. 277–306.

[SVP+96] Scholl (Michel), Voisard (Agnés), Peloux (Jean-Paul), Raynal (Laurent) et Rigaux (Philippe). – *SGBD Géographique, spécificités*. – Paris, France, International Thomson Publishing, 1996.

[SW95] Schloss (Gerhard A.) et Wynblatt (Michael J.). – Providing definition and temporal structure for multimedia data. *Multimedia Systems*, vol. 3, n° 5, November 1995, pp. 264–277.

[tec95] technology (O2). – *O2 Documentation, release 4.6*, 1995.

[TEI94] TEI. – TEI guidelines for electronic text encoding and interchange (p3), 1994.

[Tha90] Thanos (C.). – *Multimedia Office Filling: The MULTOS Approach*. – North-Holland, 1990.

[VAB96] Volz (Marc), Aberer (Karl) et Böhm (Klemens). – An oodbms-irs coupling for structured documents. *Bulletin of the Technical Committee on Data Engineering*, vol. 19, n° 1, March 1996, pp. 34–42.

[VBA95] Volz (Marc), Böhm (Klemens) et Aberer (Karl). – *A Flexible Approach to Combine IR Semantics and Database Technology and its Application to Structured Document Handling*. – Rapport technique n° GMD-arbeitspapier Nr. 891, Darmstadt, Germany, GMD-IPSI, January 1995.

[WDG95] Weiss (R.), Duda (A.) et Gifford (D.K.). – Composition and search with a video algebra. *IEEE*, 1995, pp. 12–25.

[Wil94] Wilkinson (R.). – Effective retrieval of structured documents. *In : 17th ACM-SIGIR Conference*, pp. 311–317. – Dublin, Ireland, July 1994.

[WMG+93] Wu (J.K.), Mehtre (B.M.), Gao (Y.J.), Lam (P.C.) et Narasimhalu (A.D.). – Star - a multimedia database system for trademark registration, 1993.

[WNM+95] Wu (J.K.), Narasimhalu (A.D.), Mehtre (B.M.), Lam (C.P.) et Gao (Y.J.). – Core: a content-based retrieval engine for multimedia information systems. *Multimedia Systems*, vol. 3, n° 1, February 1995, pp. 25–41.

[WvdBH96] Worring (Marcel), van den Berg (Carel) et Hardman (Lynda). – System design for structured hypermedia generation. *In : Visual Information Systems '96*, pp. 254–261. – Melbourne, Australia, 1996.

[YA94] Yan (T.W.) et Annevelink (J.). – Integrating a structured-text retrieval system with an object-oriented system. *In : VLDB'94*. – Chile, September 1994.

[Zha95] Zhang (J.). – Application of oodb and sgml techniques in text database: An electronic dictionary system. *SIGMOD RECORD*, vol. 24, n° 1, March 1995.

# Index des citations

# Quatrième partie

# Annexes

# Annexe A

# Caractéristiques des relations de structure

## A.1 Les caractéristiques de la structure linguistique

### A.1.1 Propriétés de la relation de composition linguistique

**Propriété A.1.1.1 (non réflexive)** Un élément linguistique ne peut pas être composé de lui-même. La relation de composition linguistique est donc non réflexive.

**Propriété A.1.1.2 (non transitive)** La relation de composition linguistique décrit uniquement les descendants structurels directs pris dans l'ensemble $OS_{ling}$. La relation de composition est donc non transitive.

Nous avonc fait le choix d'avoir une relation de composition non transitive. Ce choix nous a été dicté par la définition même des eléments pères/fils dans l'existant sur les documents structurés. Les normes, telles que SGML ou ODA, décrivent par défaut les composants directs, c'est à dire les éléments qui sont des fils directs, et non l'ensemble des descendants dans l'arborescence.

Nous définissons par la suite une fonction basée sur cette relation de composition qui permet d'accéder à l'ensemble des descendants d'un élément de la structure linguistique.

**Propriété A.1.1.3 (asymétrique)** Un élément linguistique $o_j$ ne peut pas être à la fois père de l'élément linguistique $o_i$ et fils de l'élément $o_i$. En d'autres termes, nous décrivons une arborescence structurelle, c'est à dire un graphe sans cycles.

### A.1.2 Propriétés de la relation de séquence linguistique

**Propriété A.1.2.1 (non réflexive)** Un élément linguistique ne peut pas être successeur direct de lui-même. La relation de séquence linguistique est donc non réflexive.

**Propriété A.1.2.2 (non transitive)** Comme le montre la figure 3.1 la relation de séquence linguistique décrit uniquement le successeur séquentiel direct d'un élément parmi ses frères.

Cette relation n'exprime donc pas le fait qu'un élément est *avant* ou *après* un autre élément mais elle exprime que deux éléments linguistiques sont frères et se succèdent dans la lecture du texte. Cette relation n'est donc pas transitive.

Il est toutefois possible, comme nous le verrons lors de la définiton de la fonction $Desc_{seq\_ling}$ (section B), d'exprimer à partir de cette relation de sèquentialité le positionnement relatif de deux éléments non successeurs directs dans l'espace linéaire de lecture du texte.

**Propriété A.1.2.3 (asymétrique)** : un élément linguistique $o_j$ ne peut pas être à la fois successeur direct de l'élément $o_i$ ($o_i \prec_{seq\_ling} o_j$) et prédécesseur direct de ce même élément $o_i$ ($o_j \prec_{seq\_ling} o_i$).

## A.2 Les caractéristiques de la structure logique

### A.2.1 Propriétés de la relation de composition logique

**Propriété A.2.1.1 (non réflexive)** Un élément logique ne peut pas être composé de lui-même. La relation de composition logique est donc non réflexive.

**Propriété A.2.1.2 (non transitive)** La relation de composition logique décrit uniquement les descendants structurels directs d'un élément de l'ensemble $OS_{log}$. Nous montrons ceci dans la figure 3.3. La propriété de non transitivité doit donc être vérifiée pour être certain de décrire les descendants structurels directs (fils directs).

Comme pour la composition linguistique, nous avonc fait le choix d'avoir une relation de composition non transitive. Dans le cas de la structre logique, ce choix nous a en partie été dicté par la définition de la structure logique du document dans les normes de représentation du document structuré, particulièrement SGML et ODA [Bur94, Ass91]. Par la suite (section B), nous définissons une fonction basée sur la relation de composition qui permet d'accéder à l'ensemble des composants d'un élément structurel: la fermeture transitive de la relation de composition logique.

**Propriété A.2.1.3 (asymétrique)** Si un élément logique $o_i$ est l'élément logique père de l'élément $o_j$ alors la relation symétrique, à savoir l'élément $o_j$ est le père de l'élément $o_i$, n'est pas possible. En d'autres termes, nous souhaitons décrire une arborescence structurelle, c'est à dire un graphe sans cycles. La relation est donc asymétrique.

### A.2.2 Propriétés de la relation de séquence logique

**Propriété A.2.2.1 (non réflexive)** Par définition, le successeur direct d'une entité logique d'un document est une autre entité. Cette relation est donc non réflexive.

**Propriété A.2.2.2 (non transitive)** La relation de séquence logique décrit les enchaînements directs entre les éléments, c'est à dire pour un élément donné, la relation donne son successeur direct, si celui-ci existe. De manière à conserver uniquement le successeur direct la relation de séquence logique doit être non transitive.

Dans la figure 3.3, nous avons représenté la relation de séquence entre les éléments logiques d'un document. Cette relation existe uniquement entre frères structurels. Par exemple, l'élément $o_{13}$ est le successeur direct de l'élément $o_{12}$ qui est le successeur direct de l'élément $o_{11}$. Ces trois éléments ont même père. D'autre part, l'élément $o_2$ est *après* l'élément $o_{13}$ dans le document mais ces deux éléments ne sont pas frères. La relation de séquence n'existe donc pas entre ces deux éléments. IL est toutefois possible de la retrouver en utilisant la relation de séquence et la relation de composition.

En effet, l'élément $o_{13}$ est le fils de l'élément $o_1$ qui est le prédécesseur séquentiel direct de $o_2$ (selon la relation de séquence $o_1 \prec_{seq\_log} o_2$). On en déduit donc que l'élément $o_2$ est *après* l'élément $o_{13}$. Nous donnons par la suite (section B) les moyens d'accéder à l'ensemble des successeurs d'un élément de structure par l'intermédiaire d'une fonction.

**Propriété A.2.2.3 (asymétrique)** Si l'élément logique $o_j$ est successeur direct de l'élément logique $o_i$ ($o_i \prec_{seq\_log} o_j$), alors l'élément $o_i$ ne peut pas être successeur direct de l'élément $o_j$. La relation de symétrie introduirait un cycle dans la séquence linéaire. La relation est donc asymétrique.

## A.3 Les caractéristiques de la structure de discours

### A.3.1 Propriétés de la relation de dominance

**Propriété A.3.1.1 (non réflexive)** Un élément de discours ne peut pas être mis en relation avec lui-même. La relation de dominance est donc non réflexive.

**Propriété A.3.1.2 (non transitive)** La relation de dominance décrit l'arborescence des idées du document textuel. Nous décrivons cettearborescence par une relation de composition donnant uniquement les descendants directs, donc non transitive.

La fermeture transitive de la relation de dominance peut être décrite par l'intermdiaire d'une fonction.

**Propriété A.3.1.3 (asymétrique)** Si l'élément de discours $o_i$ dominel'élément de discours $o_j$ ($o_i \prec_{domine} o_j$) alors la relation symétrique (($o_j \prec_{domine} o_i$) ne peut être vérifiée. Nous rappelons que cette relation décrit l'arborescence des idées du document textuel.

### A.3.2 Propriétés de la relation d'intention

**Propriété A.3.2.1 (réflexive)** Tout élément de discours a pour intention les idées exprimées dans lui même: $\forall o \in OS_{disc}, o \preceq_{intention} o$

**Propriété A.3.2.2 (non transitive)** Nous considérons une relation d'intention non transitive car les éléments sources et les éléments destinations ont un rôle bien distincts dans cette relation. L'élément source contient une information concernant l'élément destination.

**Propriété A.3.2.3 (asymétrique)** Si un élément de discours $o_i$ a pour intention un élément de discours $o_j$, la relation symétrique n'est pas valide.

# Annexe B

# Opérateurs d'accès dans la structure

## B.1 Définition d'opérateurs d'accès

Les fonctions que nous définissons dans cette partie sont dédiées à la manipulation et à l'accès aux éléments structurels d'un document textuel. Elles permettent de manipuler plus simplement les éléments en se basant sur les propriétés des relations de composition et de séquence précédemment définies.

Le modèle de document textuel décrit une hiérarchie structurelle à partir de deux types de relations non transitives : une relation de composition et une relation de séquence. De plus cette hiérarchie contient deux sortes d'éléments correspondant à deux types de granularité : les éléments structurels linguistiques et les éléments structurels logiques.

Les besoins d'accès et de manipulation de ces éléments proviennent principalement de besoins propres à l'interrogation des documents. Nous donnons une liste d'opérateurs qu'un langage de requêtes dédié à des documents textuels doit inclure. La plupart de ces opérateurs apparaissent dans [NBY95b].

### B.1.1 Opérateurs structurels directs

Accès au prédécesseur directs et aux successeurs direct dans la hiérarchie structurelle d'un document structurel. Ce type d'opérateur a du sens uniquement sur les éléments logiques. Nous définissons ainsi les deux fonctions $pred_{str}$ et $succ_{str}$ qui vont permettre de répondre à une requête du type '*donner le successeur direct d'un élément logique de type t* :

- **pred_direct$_{str}$** : accès à l'élément logique père direct d'un élément logique donné.

  $$pred\_direct_{str} : OS_{log} \rightarrow OS_{log}$$

  Un élément logique $o$ admet un prdécesseur direct si et seulement la relation de composition avec un élément logique de l'ensemble $OS_{log}$ existe comme nous la décrivons

ci-dessous :

$$pred\_direct_{str}(o) = o_p \textbf{ ssi } \exists\, o_p \in OS_{log} \text{ tel que } o_p \prec_{comp\_text} o$$

- **succ_direct$_{str}$** : accès à l'ensemble des éléments logique fils directs d'un élément logique donné.

$$succ\_direct_{str} : OS_{log} \rightarrow 2^{OS_{log}}$$

Un élément logique $o$ non feuille admet au moins un successeur direct dans la hiérarchie structurelle d'un document textuel.

$$succ\_direct_{str}(o) = \{o_p | o_f \in OS_{log} \text{ et } o \prec_{comp\_text} o_f\}$$

Les opérateurs structurels directs sont :

- $EstSuccDirect(T)$ : ensemble des éléments qui sont successeurs directs des éléments de types $T$.
- $EstSuccDirect(T, P)$ : ensemble des éléments de type $P$ qui sont successeurs directs des éléments de types $T$.
- $EstPredDirect(T)$ : ensemble des éléments qui sont prédécesseurs directs des éléments de types $T$.
- $EstPredDirect(T, P)$ : ensemble des éléments de type $P$ qui sont prédécesseurs directs des éléments de types $T$.

### B.1.2 Opérateurs d'inclusion

Accès à l'ensemble des prédécesseurs et à l'ensemble des successeurs dans la hiérarchie structurelle d'un document structuré. Pour résoudre ce type de requêtes, nous définissons deux fonctions correspondant à la fermeture transitive de la relation de composition logique et une fonction permettant d'accéder aux feuilles de la structure logique:

- **pred$_{str}$** : accès à l'ensemble des éléments logique pères d'un élément logique donné.

$$pred_{str} : OS_{log} \rightarrow 2^{OS_{log}}$$

L'ensemble des pères (ou prédécesseurs) structurels d'un élément $o$ est défini comme l'union de l'ensemble des éléments $o_i$, descendants directs de $o$ (noté $pred_{str\_1}$), avec l'ensemble des descendants structurels des $o_i$ (noté $pred_{str\_2}$).

$$pred_{str}(o) = pred_{str\_1} \cup pred_{str\_2}$$

$$\textbf{avec} \begin{cases} pred_{str\_1} = \{o_i | o_i = pred\_direct_{str}(o)\} \\ \\ pred_{str\_2} = \{o_i \mid \exists\ o_j \in OS_{log} \text{ tel que } o_j = pred\_direct_{str}(o) \text{ et } o_i \in pred_{str}(o_j)\} \end{cases}$$

– **succ$_{\text{str}}$** : accès à l'ensemble des éléments logique fils d'un élément logique donné.

$$succ_{str} : OS_{log} \rightarrow 2^{OS_{log}}$$

L'ensemble des successeurs structurels d'un élément $o$ est défini comme l'union de l'ensemble des éléments $o_i$, descendants directs de $o$ (noté $succ_{str\_1}$), avec l'ensemble des descendants structurels des $o_i$ (noté $succ_{str\_2}$).

$$succ_{str}(o) = succ_{str\_1} \cup succ_{str\_2}$$

$$\textbf{avec} \begin{cases} succ_{str\_1} = succ\_direct_{str}(o) \\ \\ succ_{str\_2} = \{o_i \mid \forall\ o_j \in OS_{log} \text{ tel que } o_j \in succ\_direct_{str}(o) \text{ et } o_i \in succ_{str}(o_j)\} \end{cases}$$

– **F$_{\text{str_log}}$** : accès à l'ensemble des éléments logique feuilles qui sont des fils d'un élément logique donné.

$$F_{str\_log} : OS_{log} \rightarrow 2^{OS_{f\_log}}$$

Si l'élément $o$ est un élément feuille, l'ensemble résultat est le singleton $\{o\}$. Sinon l'ensemble résultat est composé des éléments fils de $o$ qui sont dans l'ensemble des éléments logiques feuilles, noté $OS_{f\_log}$.

$$F_{str\_log}(o) = \begin{cases} \{o\} & \text{si } o \in OS_{f\_log} \\ \\ \{o_i | o_i \in OS_{f\_log} \text{ et } o_i \in succ_{str}(o)\} & \text{sinon} \end{cases}$$

– $EstSucc(T)$ : ensemble des éléments qui sont successeurs des éléments de types $T$.

– $EstSucc(T, P)$ : ensemble des éléments de type $P$ qui sont successeurs des éléments de types $T$.

– $EstPredDirect(T)$ : ensemble des éléments qui sont prédécesseurs des éléments de types $T$.

– $EstPredDirect(T, P)$ : ensemble des éléments de type $P$ qui sont prédécesseurs des éléments de types $T$.

### B.1.3 Opérateurs de positionnement

Le texte peut être représenté sur un axe décrivant l'enchaînement des chaînes de caractères. Les travaux de Allen [All83] qui décrivent un ensemble complet de relations spataiales dans un espace à une dimension peuvent ici être réutilisés pour former des prédicats primitifs de positionnement.

*faire un schema sur une arborescence montrant le resultat des differentes fonctions pour un élément de l'arborescence*

- **succ_direct$_{seq}$** : accès au successeur séquentiel direct d'un élément logique donné.

$$succ\_direct_{seq} : OS_{log} \rightarrow OS_{log}$$

Nous donnons la définition suivante pour tout élément $o$ de $OS_{text}$. Notons toutefois qu'il existe des éléments qui n'ont pas de successeur direct, par exemple l'élément racine qui représente le document complet.

$$succ\_direct_{seq}(o) = \begin{cases} o_i & \textbf{si } o \prec_{seq\_log} o_i \text{ existe} \\ succ\_direct_{seq}(pred\_direct_{str}(o)) & \textbf{sinon} \end{cases}$$

- **succ$_{seq}$** : accès à l'ensemble des éléments logique successeurs séquentiels d'un élément logique donné.

$$succ_{seq} : OS_{log} \rightarrow 2^{OS_{log}}$$

Afin d'exprimer cet ensemble des successeurs d'un élément, nous décrivons trois sous-ensembles : l'ensemble composé du successeur direct (s'il existe), noté $succ_{seq\_1}$, l'ensemble des fils de ce successeur direct, noté $succ_{seq\_2}$, et enfin l'ensemble des successeurs séquentiels des péres de l'élément donné, noté $succ_{seq\_3}$.

$$succ_{seq}(o) = succ_{seq\_1} \cup succ_{seq\_2} \cup succ_{seq\_3}$$

$$\text{avec } \begin{cases} succ_{seq\_1} = \{o_i | o_i = succ\_direct_{seq}(o)\} \\ succ_{seq\_2} = \{o_i | o_j = succ\_direct_{seq}(o) \text{ et } o_i \in succ_{str}(o_j)\} \\ succ_{seq\_3} = \{o_i | \forall o_j \in pred_{str}(o),\ o_i \in succ_{seq}(o_j)\} \end{cases}$$

– **first$_{\mathbf{seq}}$** : premier élément structurel feuille dans la séquence qui est un composant d'un élément structurel donné.

$$first_{seq} : OS_{log} \rightarrow OS_{f\_log}$$

Si l'élément $o$ est une feuille de la structure logique alors $first_{seq}(o) = o$. Sinon, l'élément résultat noté $o_{first}$ est un élément fils de $o$ qui est aussi un élément feuille et dont l'ensemble des successeurs séquentiels ($succ_{seq}(o_{first})$) inclut l'ensemble des éléments feuilles de $o$ (d'où il a été exclu : $F_{str\_log}(o) \Leftrightarrow \{o_{first}\}$).

$$first_{seq}(o) = \begin{cases} o & \text{si } o \in OS_{f\_log} \\ o_{first} \quad \text{tel que } o_{first} \in F_{str\_log}(o) & \\ \qquad \text{et } F_{str\_log}(o) \Leftrightarrow \{o_{first}\} \subseteq succ_{seq}(o_{first}) & \text{sinon} \end{cases}$$

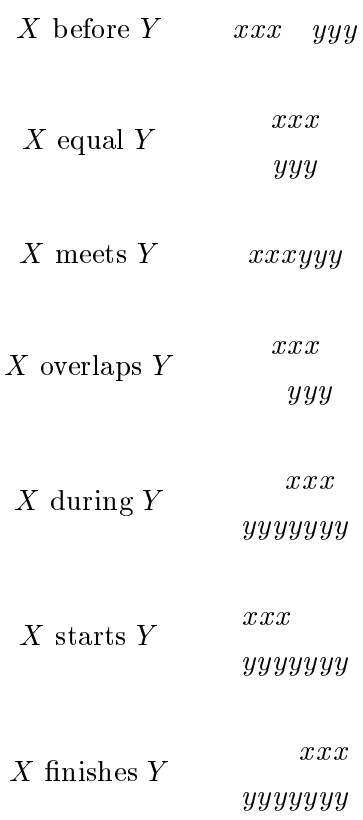
– **last$_{\mathbf{seq}}$** : dernier élément structurel feuille dans la séquence qui est un composant d'un élément structurel donné.

$$last_{seq} : OS_{log} \rightarrow OS_{f\_log}$$

Si l'élément $o$ est une feuille de la structure logique, alors $last_{seq}(o) = o$. Sinon, l'élément résultat noté $o_{last}$ est un élément fils de $o$ qui est aussi un élément feuille et vérifiant la propriété suivante : l'intersection des successeurs de $o_{last}$ avec les éléments feuilles qui composent $o$ ($F_{str\_log}(o)$) a pour résultat l'ensemble vide.

$$last_{seq}(o) = \begin{cases} o & \text{si } o \in OS_{f\_log} \\ o_{last} \quad \text{tel que } o_{last} \in F_{str\_log}(o) & \\ \qquad \text{et } F_{str\_log}(o) \cap succ_{seq}(o_{last}) = \emptyset & \text{sinon} \end{cases}$$

les relations primitives de Allen sont au nombre de 7 : before, equal, meets, overlaps, during, starts et finishes.




La seule relation que notre modèle n'autorise pas est la relation *overlpas*. En effet, nous ne permettons pas que deux éléments se recoupent. Ceci est du á la vision hiérarchique de la structure que nous avons utilisé.

- **before**$(\mathbf{o_x}, \mathbf{o_y})$ : l'élément $o_x$ précède l'élément $o_y$.

  $$before(o_x, o_y) \Leftrightarrow o_y \in succ_{seq}(o_x)$$

- **equal**$(\mathbf{o_x}, \mathbf{o_y})$ : l'élément $o_x$ est égal à l'élément $o_y$. Ceci est vrai uniquement lorsque $o_x = o_y$.

  $$equal(o_x, o_y) \Leftrightarrow o_x = o_y$$

- **meets**$(\mathbf{o_x}, \mathbf{o_y})$ : l'élément $o_x$ se termine lorsque l'élément $o_y$ débute.

  $$meets(o_x, o_y) \Leftrightarrow succ\_direct_{seq}(last_{seq}(o_x)) = first_{seq}(o_y)$$

- **during**$(\mathbf{o_x}, \mathbf{o_y})$ : l'élément $o_x$ est inclus dans l'élément $o_y$.

  $$during(o_x, o_y) \Leftrightarrow o_x \in succ_{str}(o_y)$$

– **starts**($\mathbf{o_x}, \mathbf{o_y}$) : l'élément $o_x$ et l'élément $o_y$ ont le même point de départ.

$$starts(o_x, o_y) \Leftrightarrow first_{seq}(o_x) = first_{seq}(o_y)$$

– **finishes**($\mathbf{o_x}, \mathbf{o_y}$) : l'élément $o_x$ et l'élément $o_y$ ont le même point d'arrivée.

$$finishes(o_x, o_y) \Leftrightarrow last_{seq}(o_x) = last_{seq}(o_y)$$

# Annexe C

# Complément du modèle d'images

## C.1 La structure spatiale

### C.1.1 Approche considérée

Nous considérons la structure spatiale dans une image selon deux points de vue: une structure spatiale dans l'espace à deux dimensions de l'image et une structure spatiale dans l'espace original des objets de l'image. Le plus souvent cette distinction nous amène à considérer d'une part des relations spatiales à deux dimensions et d'autre part des relations spatiales dans un espace à 3 dimensions.

Il nous semble primordial de pouvoir fournir au niveau de l'interrogation un ensemble de relations spatiales qui peuvent aider l'utilisateur dans l'expression d'une requête sur le contenu d'une image. C'est pour cette raison que nous introduisons, en plus des relations dites dans le plan de l'image (2D), des relations dans un espace à 3 dimensions.

Nous allons donc chercher d'une part à classer un certain nombre de relations dans ces deux espaces et leur donner une signification précise. Puis nous déterminerons selon les types d'objets les relations qui peuvent les lier.

Par ailleurs il sera intéressant de voir s'il est possible de déduire à partir de relations d'autres relations. Il a été montré que certaines règles de déductions s'appliquent pour les relations spatiales dites 2D [Mec95c]. Nous nous intéresserons donc particulièrement à des règles entre relations spatiales 2D et 3D.

### C.1.2 Classification des relations spatiales

De nombreux travaux ont abordé l'étude des relations spatiales dans différents cadres. Nous retiendrons parmi ceux-ci les travaux qui se sont intéressés au contenu de l'image [Mec95c, Meg95, CDDA88], l'élaboration plus récente d'une algèbre sur les relations spatiales [KXW95] et le cas plus pragmatique des systèmes de gestion de bases de données géographiques [SVP+96].

Habituellement on considère un ensemble de 13 relations spatiales pour décrire complè-

tement le positionnement relatif des objets les uns par rapport aux autres. Cet ensemble provient d'une dérivation dans un espace à deux dimensions des relations sur les intervalles dans un espace à une dimension [All83].

Ces relations se limitent toutefois à un espace à deux dimensions qui n'est pas forcément le plus intuitif pour un utilisateur. Il peut bien entendu souhaiter retroouver des objets qui sont *à droite*, *en haut* d'un objet ou qui *couvre* cet autre objet. Cependant il peut aussi souhaiter se positionner dans un espace à trois dimensions qui est plus naturel. Dans ce cas, il demandera un objet qui est *devant* ou *à côté* d'un autre objet.

Nous allons donc dans un premier temps distinguer ces différents relations et es classer selon l'espace dans lequel il faut les comprendre.

**Relations 2D** :

1. *disjoint*: deux objets $o_1$ et $o_2$ sont disjoints dans le plan de l'image s'il n'existe pas au moins un pixel qui soit commun aux deux objets.
   Tout type d'objet peut être lié par la relation *disjoint*.
2. *overlap* (chevauchement) : deux objets $o_1$ et $o_2$ se chevauchent si et seulement si la projection de l'objet $o_1$ sur l'axe des abscisses est incluse dans la projection de l'objet $o_2$ sur l'axe des abscisses et si la projection de l'objet $o_2$ sur l'axe des ordonnées est incluse dans la projection de l'objet $o_2$ sur l'axe des ordonnées (ou inversement).
   L"application de cette relation de *chevauchement* ne se fait pas sur les objets de type *point*.
3. *inside* : un objet $o_1$ est à l'intérieur d'un autre objet $o_2$ si et seulement si le contour de l'objet $o_1$ est englobé par le contour de l'objet $o_2$.
   Tout type d'objet peut être lié par la relation *inside*.
4. *equal*: deux objets $o_1$ et $o_2$ sont égaux si et seulement si ils sont décrits par le même ensemble de pixels.
   Tout type d'objet peut être lié par la relation *equal*.
5. *covers*: un objet $o_1$ couvre un objet $o_2$ si et seulement si l'ensemble de pixels composant $o_2$ est inclus dans l'ensemble de pixels composant $o_1$.
   Par définition, la relation de composition entre deux objets $o_1$ et $o_2$ implique cette relation entre $o_1$ et $o_2$ : si $o_1$ est composé de $o_2$, $o_1 \prec_{comp\_pic} o_2$ alors $o_1$ couvre $o_2$, $covers(o_1, o_2)$.
6. *covered_by*: un objet $o_1$ est couvert par un objet $o_2$ si et seulement si l'ensemble de pixels composant $o_2$ inclut l'ensemble de pixels composant $o_1$.
   C'est la relation inverse de la relation *covers*. Elle peut donc elle aussi être déduite de la relation de composition structurelle.
7. *touches*: deux objets $o_1$ et $o_2$ se touchent si et seulement si l'intersection de leur contour est non vide.
   Cette relation a un sens lorsque les objets sont de type région ou élément.
8. *north* : un objet $o_1$ est au nord d'un objet $o_2$ si et seulement si la projection de l'objet $o_1$ sur l'axe des ordonnées est *supérieur* à la projection de l'objet $o_2$ sur ce même axe.

Cette relation (et les trois qui suivent) peuvent lier tout type d'objets d'une image.

9. *south* : un objet $o_1$ est au sud d'un objet $o_2$ si et seulement si la projection de l'objet $o_1$ sur l'axe des ordonnées est *inférieur* à la projection de l'objet $o_2$ sur ce même axe.
10. *east* : un objet $o_1$ est à l'est d'un objet $o_2$ si et seulement si la projection de l'objet $o_1$ sur l'axe des abscisses est *supérieur* à la projection de l'objet $o_2$ sur ce même axe.
11. *west* : un objet $o_1$ est à l'ouest d'un objet $o_2$ si et seulement si la projection de l'objet $o_1$ sur l'axe des abscisses est *inférieur* à la projection de l'objet $o_2$ sur ce même axe.

La projection sur un axe est les opérateurs qui s'y rapportent sont décrits à la section c).

**Relations 3D** :

Il est difficile de définir ces relations dans l'espace à trois dimensions. D'une part la tâche qui consiste à instancier ces relations est une tâche humaine. Leur définiton va donc comporter une part de subjectivité. Par exemple une relation qui exprime la proximité entre deux objets sera basée sur la *distance réelle* entre ces deux objets. Cette distance réelle n'est pas mesurable avec les seules informations contenues dans l'image. Il s'agit donc d'une estimation de la part de l'indexeur qui dira s'il considère que les objets sont proches ou éloignés.

1. *près de* : la notion de proximité entre deux objets s'exprime uniquement dans le monde original (3D). De plus nous considérons qu'elle a un sens uniquement entre des objets de type *élement* ou *scène*, c'est à dire des objets ayant une représentation sémantique associée.
2. *loin de* : cette relation est la relation complémentaire de la relation précédente. Si la relation *près de* n'existe pas entre deux objets $o_1$ et $o_2$ alors on considère que la relation *loin de* est valable.

   Elle s'applique aussi uniquement su des objets de type *élement* ou *scène*.
3. *devant* : un objet $o_1$ est devant un objet $o_2$ si d'une part ils n'appartiennent pas au même plan dans l'espace à trois dimensions et si d'autre part ...
4. *derrière* : cette relation est la relation opposée de la relation précédente. Si la relation *devant* existe entre deux objets $o_1$ et $o_2$, l'objet $o_1$ est devant l'objet $o_2$, alors on considère que la relation *derrière* existe, l'objet $o_2$ est derrière l'objet $o_1$.

### C.1.3 L'ensemble des relations spatiales

L'ensemble des relations spatiales, noté $R_{sp}$, se décompose en deux sous-ensembles : l'ensemble des relations 2D, noté $R_{2D}$, et l'ensemble des relations 3D, noté $R_{3D}$.

$$R_{sp} = R_{2D} \cup R_{3D} \text{ et } R_{2D} \cap R_{3D} = \emptyset$$

Ces relations s'expriment entre des objets de l'image. Cependant nous imposons des contraintes sur les types des objets selon les relations.

- Les relations dans l'espace à trois dimensions s'expriment uniquement entre des objets qui ont une réalité dans cet espace. Pour nous il s'agira donc des objets qui ont une représentation sémantique : objets de type *élément* et *scène*.
- Les relations 2D sont contraintes par les types d'objet. Nous résumons les contraintes qui doivent être vérifiées pour qu'une relation $\Delta_{sp}$, $\Delta_{sp} \in R_{2D}$, soit possible entre deux objets.

- **disjoint**(**A**, **B**)
  - $type_{pict}(A) = point$ et $type_{pict}(B) = point$ : $A \neq B$
  - $type_{pict}(A) = point$ et $type_{pict}(B) = region$ : impossible
  - $type_{pict}(A) = region$ et $type_{pict}(B) = region$ :

## C.2 Calcul des relations spatiales

Nous définissons ici les relations spatiales entre les objets de l'image, puis par la suite nous décrivons celles qui peuvent être déduites de la relation de composition et celles qui doivent être calculées selon les informations qui sont à notre disposition.

**disjoint** : deux objets $o_1$ et $o_2$ sont disjoints si et seulement si $pixel(o_1) \cup pixel(o_2)$ forme un ensemble de pixels non connectés.

**overlap** (chevauchement) : deux objets $o_1$ et $o_2$ se chevauchent si et seulement si il existe au moins un intervalle $i \subset [1, l]$ tel que $i \sqsubseteq \pi_x(o_1)$ et $i \sqsubseteq \pi_x(o_2)$, et il existe au moins un intervalle $j \subset [1, h]$ tel que $j \sqsubseteq \pi_y(o_1)$ et $j \sqsubseteq \pi_y(o_2)$

La relation de chevauchement peut uniquement exister entre des objets de type région ou scène, c'est à dire entre des objets représentant des ensembles non connectés de pixels.

**inside** : un objet $o_1$ est à l'intérieur d'un autre objet $o_2$ si et seulement si $\pi_x(o_1) \sqsubseteq \pi_x(o_2)$ et $\pi_y(o_1) \sqsubseteq \pi_y(o_2)$.

L'objet $o_2$ représente un ensemble non connecté de pixels.

**equal** : un objet $o_1$ est égal à un objet $o_2$ si et seulement si $pixel(o_1) = pixel(o_2)$.

**covers** : un objet $o_1$ recouvre un objet $o_2$ si et seulement si $pixel(o_1) \subset pixel(o_2)$.

**covered_by** : un objet $o_1$ est recouvert par un objet $o_2$ si et seulement si $pixel(o_2) \subset pixel(o_1)$.

**north** : un objet $o_1$ est au nord d'un objet $o_2$ si et seulement si $\pi_y(o_1) > \pi_y(o_2)$.

**south** : un objet $o_1$ est au sud d'un objet $o_2$ si et seulement si $\pi_y(o_1) < \pi_y(o_2)$.

**east** : un objet $o_1$ est au nord d'un objet $o_2$ si et seulement si $\pi_x(o_1) < \pi_x(o_2)$.

**west** : un objet $o_1$ est au nord d'un objet $o_2$ si et seulement si $\pi_x(o_1) > \pi_x(o_2)$.

Afin de pouvoir déduire certaines relations spatiales selon la relation de composition entre les objets , nous allons montrer ce qu'implique l'existence d'une relation de composition entre deux objets.

Soit deux objets $o_i$ et $o_j$, pris dans l'ensemble $OS_{pic}$, et l'objet $o_i$ est le père structurel de $o_j$ : $o_i \prec_{str\_pic} o_j$.

1. L'ensemble des pixels de l'objet $o_j$ est inclus dans l'ensemble des pixels de l'objet $o_i$ :

   $$pixel(o_j) \subset pixel(o_i)$$

2. L'application des opérateurs de projections sur les dimensions X et Y vérifie les conditions suivantes :

   $$\pi_x(o_j) \sqsubseteq \pi_x(o_i) et \pi_y(o_j) \sqsubseteq \pi_y(o_i)$$

# Annexe D

# Compléments pour la résolutions de conflits des portées

## D.1 Propriétés des ensembles caractéristiques de la catégorie succ

Nous énonùons ici les caractéristiques des ensembles $p_{(i,j)}$, $p_i$ et $p_j$ pour deux éléments admettant une portée selon la relation $rel$ et la catégorie $succ$:

- **Portée commune à $o_i$ et $o_j$ notée $p_{(i,j)}$**

  $$p_{(i,j)} \subseteq Next_{rel}(o_i) \cap Next_{rel}(o_j)$$

  1. $o_j \in Next_{rel}(o_i)$ : la portée commune correspond à la portée de $o_j$.

     $$p_{(i,j)} \subseteq Next_{rel}(o_j)$$

  2. $o_i \in Next_{rel}(o_j)$ : la portée commune correspond à la portée de $o_i$.

     $$p_{(i,j)} \subseteq Next_{rel}(o_i)$$

  3. $o_i \notin Next_{rel}(o_j)$ et $o_j \notin Next_{rel}(o_i)$ : la portée commune est l'ensemble vide car les éléments $o_i$ et $o_j$ sont différents.

     $$p_{(i,j)} = \emptyset$$

- **Successeurs spécifiques à $o_i$, ensemble noté $p_i$**

  1. $o_j \in Next_{rel}(o_i)$ : les successeurs spécifiques à $o_i$ correspondent à la portée de $o_i$ sans les éléments de la portée de $o_j$.

     $$p_i \subseteq Next_{rel}(o_i) \Leftrightarrow Next_{rel}(o_j)$$

  2. $o_i \in Next_{rel}(o_j)$ : les successeurs spécifiques à $o_i$ correspondent à la portée de $o_i$.

     $$p_i \subseteq Next_{rel}(o_i)$$

3. $o_i \notin Next_{rel}(o_j)$ et $o_j \notin Next_{rel}(o_i)$ : les successeurs spécifiques à $o_i$ correspondent à la portée de $o_i$.

$$p_i \subseteq Next_{rel}(o_i)$$

– **Successeurs spécifiques à $o_j$, ensemble noté $p_j$**

1. $o_j \in Next_{rel}(o_i)$ : les successeurs spécifiques à $o_j$ correspondent à la portée de $o_j$,

$$p_j \subseteq Next_{rel}(o_j)$$

2. $o_i \in Next_{rel}(o_j)$ : les successeurs spécifiques à $o_j$ correspondent à la portée de $o_j$, sans les éléments de la portée de $o_i$.

$$p_j \subseteq Next_{rel}(o_j) \Leftrightarrow Next_{rel}(o_i)$$

3. $o_i \notin Next_{rel}(o_j)$ et $o_j \notin Next_{rel}(o_i)$ : les successeurs spécifiques à $o_j$ correspondent à la portée de $o_j$,

$$p_j \subseteq Next_{rel}(o_j)$$

## D.2 Propriétés des ensembles caractéristiques de la catégorie pred

Nous énonùons ici les caractéristiques des ensembles $p_{(i,j)}$, $p_i$ et $p_j$ pour deux éléments admettant une portée selon la relation $rel$ et la catégorie $pred$:

– **Portée commune à $o_i$ et $o_j$ notée $p_{(i,j)}$**

$$p_{(i,j)} \subseteq Prev_{rel}(o_i) \cap Prev_{rel}(o_j)$$

1. $o_j \in Prev_{rel}(o_i)$ : la portée commune correspond à la portée de $o_j$.

$$p_{(i,j)} \subseteq Prev_{rel}(o_j)$$

2. $o_i \in Prev_{rel}(o_j)$ : la portée commune correspond à la portée de $o_i$.

$$p_{(i,j)} \subseteq Prev_{rel}(o_i)$$

3. $o_i \notin Prev_{rel}(o_j)$ et $o_j \notin Prev_{rel}(o_i)$ : la portée commune est l'ensemble vide car les éléments $o_i$ et $o_j$ sont différents.

$$p_{(i,j)} = \emptyset$$

– **Prédécesseurs spécifiques à $o_i$, ensemble noté $p_i$**

1. $o_j \in Prev_{rel}(o_i)$ : les prédécesseurs spécifiques à $o_i$ correspondent à la portée de $o_i$ sans les éléments de la portée de $o_j$.

$$p_i \subseteq Prev_{rel}(o_i) \Leftrightarrow Prev_{rel}(o_j)$$

2. $o_i \in Prev_{rel}(o_j)$ : les prédécesseurs spécifiques à $o_i$ correspondent à la portée de $o_i$.

   $p_i \subseteq Prev_{rel}(o_i)$

3. $o_i \notin Prev_{rel}(o_j)$ et $o_j \notin Prev_{rel}(o_i)$ : les prédécesseurs spécifiques à $o_i$ correspondent à la portée de $o_i$.

   $p_i \subseteq Prev_{rel}(o_i)$

– **Prédécesseurs spécifiques à $o_j$, ensemble noté $p_j$**

1. $o_j \in Prev_{rel}(o_i)$ : les prédécesseurs spécifiques à $o_j$ correspondent à la portée de $o_j$,

   $p_j \subseteq Prev_{rel}(o_j)$

2. $o_i \in Prev_{rel}(o_j)$ : les prédécesseurs spécifiques à $o_j$ correspondent à la portée de $o_j$, sans les éléments de la portée de $o_i$.

   $p_j \subseteq Prev_{rel}(o_j) \Leftrightarrow Prev_{rel}(o_i)$

3. $o_i \notin Prev_{rel}(o_j)$ et $o_j \notin Prev_{rel}(o_i)$ : les prédécesseurs spécifiques à $o_j$ correspondent à la portée de $o_j$,

   $p_j \subseteq Next_{rel}(o_j)$

## D.3 Algorithmes de résolutions de conflits des attributs de composition ascendant

Nous considérons dans un premier temps la catégorie *pred* avec la relation de composition, c'est-à-dire un attribut de composition ascendant. Le point de départ du parcours correspond donc aux éléments structurels de l'ensemble $OS_\alpha$ qui ont la profondeur maximum, $N = Max$.

$$Max = \underset{o_i \in OS_\alpha}{Max} (o_i.d)$$

Nous exprimons ici les différentes conditions que nous pouvons rencontrer lors de parcours et donnons les actions corrspondant à chacune de ces conditions.

1. Deux éléments structurels de $E_\alpha$ ayant un ascendant structurel direct n'appartenant pas à $E_\alpha$.

   **Condition 1**
   *L'ascendant structurel direct $o$ des éléments $o_1$ et $o_2$ de $E_\alpha$ n'appartient pas à $E_\alpha$ et est commun aux portées des deux éléments.*

   $o_1, o_2 \in E_\alpha,\ \nexists\ o \in E_\alpha$
   *tel que* $o_1 \neq o_2$ , $o_1.d = N,\ o_2.d = N,\ (o \prec_{comp} o_1)$ *et* $(o \prec_{comp} o_2)$
   *et* $o \in P^{o_1}_{\alpha,comp}$ *et* $o \in P^{o_1}_{\alpha,comp}$

**Action 1**
*Nous résolvons le conflit lié aux portées des éléments $o_1$ et $o_2$. Cette résolution entraîne l'ajout de l'élément $o$ dans l'ensemble $E_\alpha$ et nour retirons les éléments $o_1$ et $o_2$ de cet ensemble puisque les conflits liés à leurs portées sont traités. Il ne peut plus exister de conflits sur leurs portées car les éléments plus bas dans l'arborescence ont déja été traités.*

$$Resolve_\alpha(o_1, o_2);\ delete(o_1, E_\alpha);\ delete(o_2, E_\alpha);\ add(o, E_\alpha)$$

*L'appel à la fonction $Resolve_\alpha$ sur les éléments $o_1$ et $o_2$ avec les conditions exprimées précédemment va fournir à l'élément $o$ un attribut $\alpha$ instancié par la valeur $combine_\alpha(o_1.\alpha, o_2.\alpha)$.*

2. Deux éléments structurels de $E_\alpha$ ayant un ascendant structurel direct appartenant à $E_\alpha$

**Condition 2**
*L'ascendant structurel direct $o$ des éléments $o_1$ et $o_2$ de $E_\alpha$ appartient à $E_\alpha$ et est commun aux portées des deux éléments.*

$$o_1, o_2 \in E_\alpha,\ \exists\ o \in E_\alpha$$
$$tel\ que\ o_1 \neq o_2\ , o_1.d = N,\ o_2.d = N,\ (o \prec_{comp} o_1)\ et\ (o \prec_{comp} o_2)$$
$$et\ o \in P^{o_1}_{\alpha,comp}\ et\ o \in P^{o_1}_{\alpha,comp}$$

**Action 2**
*Nous résolvons le conflit lié aux portées des éléments $o$ et $o_1$, puis celle liée aux portées des éléments $o$ et $o_2$. Les éléments $o_1$ et $o_2$ sont ensuite retirés de l'ensemble $E_\alpha$. La condition exprimée ci-dessus montre qu'il y'a inclusion entre les ensemble décrivant les portées de $o$, $o_1$ et $o_2$. Nous choisissons d'effectuer ces deux traitements plutôt que la résolution entre $o_1$ et $o_2$ pour éviter l'insertion dans $E_\alpha$ de l'élément $o$ (qui appartient déjà à cet ensemble).*

$$Resolve_\alpha(o, o_1);\ delete(o_1, E_\alpha);\ Resolve_\alpha(o, o_2);\ delete(o_2, E_\alpha);$$

*Le premier appel à la fonction $Resolve_\alpha$, $Resolve_\alpha(o, o_1)$, modifie la valeur de l'attribut $\alpha$ de $o$ si nous nous plaçons dans un contexte de dépendance des valeurs : $o.\alpha = combine_\alpha(o_1.\alpha, o.\alpha)$. Le second appel à cette fonction, $Resolve_\alpha(o, o_2)$, va de nouveau modifier la valeur de $o.\alpha$ : $o.\alpha = combine_\alpha(o_2.\alpha, o.\alpha)$.*

3. Un élément structurel de $E_\alpha$ et son plus proche ascendant structurel direct appartenant à $E_\alpha$.

**Condition 3**
*L'élément $o$ est le plus petit ascendant structurel de l'élément $o_1$ de $E_\alpha$ appartenant à $E_\alpha$ et à la portée de $o_1$.*

$$o_1 \in E_\alpha,\ \exists\ o \in E_\alpha$$
$$tel\ que\ o_1.d = N,\ o \in Prev_{comp}(o_1)$$
$$et\ o.d = \underset{o_i}{Min}(o_i.d)\ avec\ o_i \in Prev_{comp}(o_1) \cap OS_\alpha$$
$$et\ o \in P^{o_1}_{\alpha,comp}$$

**Action 3**
*L'élément o est inclus dans l'ensemble décrivant la portée de l'élément $o_1$ et o est aussi un élément de $E_\alpha$. il faut donc résoudre ce conflit puis retirer $o_1$ de $E_\alpha$ pour signifier que sa portée ne peut plus rentrer en conflit avec d'autres portées.*

$$Resolve_\alpha(o_1, o);\ delete(o_1, E_\alpha);$$

Chaque ajout dans l'ensemble $E_\alpha$ signifie qu'il y a un nouveal élément qui est introduit dans l'ensemble des éléments sources noté $OS_\alpha$.

Les appels à la fonction $Resolve_\alpha$ correspondent à des résolutions de conflits décrits dans les sections a) et b) (pages 181).

Le processus général se déroule par niveau de profondeur dans l'arborescence structurelle. Pour chaque niveau de profondeur, c'est la condition 1 qui est privilégiée, puis la condition 2 et enfin la condition 3.

$$\begin{array}{l}
Parcours_{pred,comp}(E_\alpha, N) \\
DEBUT \\
\ SI\ (N = 0\ ||E_\alpha = \emptyset)\ ALORS\ exit() \\
\ SINON \\
\quad TANTQUE\ (\exists\ o_i \in E_\alpha \mid o_i.d = N) \\
\qquad \{ \\
\qquad SI\ condition1\ ALORS\ action1 \\
\qquad SINON \\
\qquad\quad SI\ condition2\ ALORS\ action2 \\
\qquad\quad SINON\ action3 \\
\qquad \} \\
\ Parcours_{pred,comp}(E_\alpha, N \Leftrightarrow 1) \\
FIN
\end{array} \tag{D.1}$$

## D.4 Algorithmes de résolutions de conflits des attributs de composition descendant

Les attributs de composition descendant se caractérisent par la catégorie *succ* et la relation de composition. Le parcours de l'arborescence structurelle va s'établir en partant des éléments les plus hauts dans cette arborescence pour atteindre ensuite les éléments les plus bas. L'ensemble de départ est constitué des éléments structurels de $OS_\alpha$ qui sont les plus haut dans l'arborescence, c'est-à-dire qui ont la profondeur minimale.

$$Min = \underset{o_i \in OS_\alpha}{Min}\ (o_i.d)$$

Nous considérons les différents cas de figure et donnons les actions qui s'appliquent à chacun d'entre eux.

1. Un élément structurel de $E_\alpha$ ayant un descendant structurel direct appartenant à $E_\alpha$ et à l'ensemble décrivant sa portée.

   **Condition 4**
   *Le descendant structurel direct $o$ de l'élément $o_1$ de $E_\alpha$ appartient à $E_\alpha$ et appartient à l'ensemble décrivant la portée de $\alpha$ depuis $o_1$.*

   $$o_1 \in E_\alpha,\ \exists\ o \in E_\alpha$$
   $$\text{tel que } o_1.d = N,\ (o_1 \prec_{comp} o) \qquad \text{et } o \in P^{o_1}_{\alpha,comp}$$

   **Action 4**
   *Nous résolvons le conflit lié aux portées des éléments $o_1$ et $o$.*

   $$Resolve_\alpha(o_1, o);$$

   *Les éléments $o$ et $o_1$ sont des éléments structurels sources, ils appartiennent à l'ensemble $E_\alpha$. La valeur de l'attribut $\alpha$ de $o$ va prendre en compte la valeur provenant de l'attribut $\alpha$ de $o_1$ : $o.\alpha = combine_\alpha(o_1.\alpha, o.\alpha)$. L'élément $o_1$ n'est pas retiré de l'ensemble $E_\alpha$ car sa portée peut inclure d'autres descendants structurels admettant un attribut $\alpha$.*

2. Un élément structurel de $E_\alpha$ ayant au moins un descendant structurel appartenant à $E_\alpha$ et à sa portée.

   **Condition 5**
   *L'élément structurel $o_1$ admet parmi ses descendants structurels un élément $o$ qui appartient à la fois à $E_\alpha$ et à la portée de $\alpha$ depuis $o_1$, ensemble noté $P^{o_1}_{\alpha,comp}$. Par ailleurs, il n'existe pas d'élément intermédiaire, c'est-à-dire d'éléments qui appartiennent à la fois à la portée de $\alpha$ depuis $o_1$ et à $E_\alpha$, et qui soit un prédécesseur de $o$.*

   $$o_1 \in E_\alpha,\ \exists\ o \in E_\alpha$$
   $$\text{tel que } o_1.d = N,\ o \in Next_{comp}(o_1) \text{ et } o \in P^{o_1}_{\alpha,comp}$$
   $$\text{et } \nexists\ o_i \in Next_{comp}(o_1) \text{ tel que } o_i \in E_\alpha \cap P^{o_1}_{\alpha,comp} \text{ et } o_i \in Prev_{comp}(o)$$

   **Action 5**
   *Nous résolvons le conflit lié aux portées des éléments $o_1$ et $o$ : $o.\alpha = combine_\alpha(o_1.\alpha, o.\alpha)$.*

   $$Resolve_\alpha(o_1, o);$$

   *Le traitement et son résultat sont ici similaires à ce qui a été effectué dans l'action 4.*

   La condition 4 peut être supprimé puisqu'il s'agit d'un cas particulier de la condition 5 qui est plus générale. Le particularisme de la condition 4 vient du fait que nous considérons les descendants structurels directs qui vérifient évidemment la condition supplémentaire introduite dans la condition 5 sur la non-existence d'éléments intermédiaires.

3. Un élément structurel de $E_\alpha$ n'ayant pas de descendant structurel appartenant à $E_\alpha$ et à sa portée.

**Condition 6**
*L'élément structurel $o_1$ n'admet pas parmi ses descendants structurels d'éléments qui appartiennent simultanément à $E_\alpha$ et à sa portée.*

$$o_1 \in E_\alpha,\ \nexists\, o \in E_\alpha$$
$$\text{tel que } o_1.d = N,\ o \in Next_{comp}(o_1) \cap P^{o_1}_{\alpha,comp}$$

**Action 6**
*L'élément $o_1$ et la portée qui lui est associée pour l'attribut $\alpha$ ne peuvent plus rencontrer de conflit. L'élément structurel $o_1$ est donc retiré de l'ensemble $E_\alpha$.*

$$delete(o_1, E_\alpha);$$

Lors du parcours de l'arborescence structurelle, nous prenons comme élément courant un élément structurel de profondeur $N$. Tant que cet élément vérifie l'une des trois conditions que nous avons présenté ci-dessus il n'est pas retiré de l'ensemble $E_\alpha$. Le passage au niveau de profondeur suivant ( $N \Leftrightarrow 1$ se fait lorsque que tout les éléments de niveau $N$ ont été traités (parcours en largeur de l'arborescence).

```
Parcours_{succ,comp}(E_α, N)
 SI (E_α = ∅)
  ALORS   exit()
  SINON
            TANTQUE (∃ o_i ∈ E_α | o_i.d = N)
              {
              SI condition4 ALORS action4                    (D.2)
              SINON
                SI condition5 ALORS action5
                SINON action6
              }
 Parcours_{succ,comp}(E_α, N ⇔1)
FIN
```

## D.5 Algorithmes de résolutions de conflits des attributs de séquence arrière

Les attributs de séquence arrière se caractérisent par une catégorie *pred* avec la relation de séquence. La relation de séquence présente moins de cas particuliers puisqu'il ne s'agit pas

d'une arborescence du fait des propriétés de la relation $\prec_{seq}$. Nous considérons donc uniquement deux conditions particulières puisque soit un élément admet pas parmi ses prédécesseurs un élément qui rentre en conflit avec lui (3) soit il n'en admet pas (7).

**Condition 7**
*L'élément structurel $o_1$ n'admet pas parmi ses prédécesseurs d'éléments qui appartiennent à $E_\alpha$ et à sa portée.*

$$o_1 \in E_\alpha,\ \nexists\, o \in E_\alpha$$
$$\text{tel que } o_1.d = N,\ o \in Prev_{seq}(o_1) \cap P^{o_1}_{\alpha,seq}$$

**Action 7**
*Il n'existe donc pas de conflit lié à cet élément et il peut être retiré de l'ensemble $E_\alpha$ et conserver la portée qui lui a été assignée.*

$$delete(o_1, E_\alpha);$$

Afin de définir ce parcours, nous nous basons sur celui décrit en D.3 en ramenant le problème à une séquence au lieu d'une arborescence. Seule les condition 3 et 7 sont alors valables puisque les autres sont des cas particuliers provenant des caractéristiques de l'arborescence.

Le parcours se résume donc à l'algorithme suivant :

$$\begin{array}{l}
Parcours_{pred,seq}(E_\alpha, N) \\
DEBUT \\
\quad SI\ (N = 0\ ||E_\alpha = \emptyset) \\
\quad\quad ALORS \quad exit() \\
\quad\quad SINON \\
\qquad\qquad\qquad TANTQUE\ (\exists\ o_i \in E_\alpha \mid o_i.d = N) \\
\qquad\qquad\qquad\quad \{ \\
\qquad\qquad\qquad\quad SI\ condition3\ ALORS\ action3 \\
\qquad\qquad\qquad\quad SINON\ action7 \\
\qquad\qquad\qquad\quad \} \\
\quad Parcours_{pred,seq}(E_\alpha, N \Leftrightarrow 1) \\
FIN
\end{array} \tag{D.3}$$

## D.6 Algorithmes de résolutions de conflits des attributs de séquence avant

Les attributs de séquence avant se caractérisent par une catégorie *succ* avec la relation de séquence. Nous introduisons la condition et l'action suivante afin de supprimer de $E_\alpha$ les éléments ne pouvant plus admettre de conflits sur leur portée. Le traitement complet reprend les conditions 5 et 8 ainsi que les actions qui leur sont associées : soit un élément est inclus

dans la portée de son prédécesseur le plus proche dans $OS_\alpha$ (condition 5), soit il n'existe pas de conflit (condition 8).

**Condition 8**
*L'élément structurel $o_1$ n'admet pas parmi ses successeurs séquentiels d'éléments qui appartiennent à la fois à $E_\alpha$ et à sa portée.*

$$o_1 \in E_\alpha,\ \nexists\, o \in E_\alpha$$
$$\text{tel que } o_1.d = N,\ o \in Next_{seq}(o_1) \cap P^{o_1}_{\alpha,seq}$$

**Action 8**
*Il n'existe donc pas de conflit lié à cet élément et il peut être retiré de l'ensemble $E_\alpha$ et conserver la portée qui lui a été assignée.*

$$delete(o_1, E_\alpha);$$

Le parcours se résume donc à l'algorithme suivant :

$$\begin{array}{l}
Parcours_{pred,seq}(E_\alpha, N) \\
DEBUT \\
\ SI\ (N = 0\ ||E_\alpha = \emptyset) \\
\quad ALORS \quad exit() \\
\quad SINON \\
\qquad\qquad TANTQUE\ (\exists\ o_i \in E_\alpha \mid o_i.d = N) \\
\qquad\qquad\quad \{ \\
\qquad\qquad\quad SI\ condition5\ ALORS\ action5 \\
\qquad\qquad\quad SINON\ action8 \\
\qquad\qquad\quad \} \\
\ Parcours_{pred,seq}(E_\alpha, N \Leftrightarrow 1) \\
FIN
\end{array} \tag{D.4}$$