Recherche d'associations séquentielles et alignement d'ontologies biologiques

Bastien Rance

N° d'ordre : 9526

Thèse

de

L'Université Paris–Sud

présentée en vue de l'obtention du

Doctorat de l'Université Paris–Sud

Spécialité : Informatique

Par

Bastien RANCE

Recherche d'associations séquentielles et alignement d'ontologies biologiques

Soutenue le 28 septembre 2009 devant la commission d'examen :

M.	Ollivier Haemmerlé	Rapporteur
M.	Jean-Marc Petit	Rapporteur
M.	Jean-François Gibrat	Examinateur
M^{me}	Frédérique Lisacek	Examinatrice
M^{me}	Chantal Reynaud	Examinatrice
M^{me}	Christine Froidevaux	Directrice de thèse

Laboratoire de Recherche en Informatique, U.M.R. CNRS 8623,
Université Paris-Sud, 91405 Orsay Cedex, France

Je tiens tout d'abord à remercier Christine Froidevaux, qui m'a fait l'honneur de diriger mes travaux de thèse pendant ces 4 années, qui m'a enseigné la rigueur de la méthode scientifique, et qui n'a pas hésité à consacrer de longues heures à de très intéressantes discussions, aussi bien pour des brainstorming que pour des séances de formalisation.

Je remercie mes rapporteurs Ollivier Haemmerlé et Jean-Marc Petit, pour leurs commentaires, leurs remarques et leurs questions, qui m'auront permis de prendre un peu plus de recul sur le travail de ces années passées. Merci à mes examinateurs, Jean-François Gibrat avec qui j'ai eu la joie de collaborer lors de ces années de thèse, Frédérique Lisacek qui m'aura permis de découvrir la bioinformatique sous un angle que je ne connaissais pas, de découvrir le cadre de la bioinformatique à Genève et pour son regard sur l'évolution de cette thématique. Enfin à Chantal Reynaud, avec qui j'ai eu la chance de travailler dans mes activités d'enseignement.

Mes remerciements vont bien sûr aussi à Alain Denise et Michel Termier, qui ont eu la brillante idée de la création de la filière BIBS (BioInformatique et BioStatistiques) ouverte en 2002, et de m'y avoir accepté en licence, puis dans les années suivantes. La dynamique de la promo "zéro" est assurément l'un des éléments qui m'auront permis de débuter cette thèse.

J'en profite pour remercier les nombreux enseignants qui m'auront marqués tout le long de ma scolarité, dans des disciplines aussi variées que l'informatique, la biologie générale, le français, la musique, les mathématiques, le violon, et bien d'autres... et qui m'auront donné l'envie de partager mes connaissances et d'essayer de les transmettre de la façon la plus vivante possible. Merci aux étudiants, qui auront subi avec plus ou moins de joie mes expériences pédagogiques.

Merci aux étudiants de cette promo "zéro" de BIBS, un groupe de "Petits Nicolas", heureux d'être là, aussi dissipés que travailleurs, toujours à l'affût d'une session mal fermée ou d'une épaule amicale ! Combien d'étudiants auront subi les affres de nos netsend, ou vu leur fichier .login vandalisé ? La question reste entière, et par égard pour les chartes signées, peut-être vaut-il mieux ne jamais essayer d'y répondre.

Ces quatre ans n'auraient pu se terminer de façon satisfaisante sans l'accueil de cette petite communauté qu'est l'équipe bioinformatique du LRI. Les moments de pauses, entre mots croisés, rébus, jeux de mots ou devinettes ardues, mais aussi les moments de discussions scientifiques ont été des instants très agréables et enrichissants. Le souvenir d'un certain "JOBIM !" crié dans une sombre rue lyonnaise restera pour longtemps dans ma mémoire comme la marque d'une solidarité sans faille entre les membres de l'équipe !

Caché au sein de l'équipe, il y a un petit îlot que je tiens à remercier particulièrement, d'abord nommé bureau 153, puis 148. Merci à celles et ceux dont j'ai partagé le bureau., Sarah Cohen-Boulakia et Lucie Gentils d'abord, puis Frédéric Lemoine, Thomas Bourquard, Cédric Saule, et les petits nouveaux Bryan Brancotte et Zahira Alsaoui. C'est un peu à cause de Sarah que j'ai fait une thèse au sein du LRI. Successivement, exemple, enseignante, collègue puis amie, elle m'a ouvert la voie vers les thématiques de l'équipe bioinformatique. A cause des concours, dont je tairai le but, entre Sarah et Lucie l'ambiance du bureau a vite été joyeuse, facilitant largement mon intégration dans les locaux. Un merci particulier à Frédéric Lemoine, camarade et ami depuis la licence BIBS, binôme de temps en temps quand nous étions sur les bancs des salles de TD, puis plus souvent du côté du tableau à l'IUT d'Orsay. Soutien indispensable pendant la thèse, jamais à court d'humour, d'une perspicacité rare, avec qui j'ai échangé des points de vue sur des sujets aussi variés que la politique, l'intérêt du patron de conception "fabrique" ou les aspirateurs USB (au passage, je ne suis toujours pas convaincu...).

La période de la rédaction a été plus aisée grâce à la présence d'Annelyse Thévenin dans les locaux un certain nombre de week-ends et à notre amicale competition.

Certains dans l'équipe assurent plus que d'autres, il s'agit bien sûr du groupe des grimpeurs :

Jérôme Azé, et Damien de Vienne, merci ! Jérôme en particulier, pour m'avoir permis par ses conseils calmes et discrets, mais toujours avisés, de franchir bien des difficultés (parfois en tête) sur les rochers comme dans mon travail de thèse.
Une thèse se réalise aussi en dehors des moments de travail. Pour tous les moments de détente, de fraîcheur, de suée sur les parois, de musique... Je voudrais remercier pêle-mèle : - la bande de Fleming. J'espère être un digne successeur de mes illustres prédécesseurs les Docteurs Sylvestre et Sapede ! - les amis du DEUG Bio (je continue à vous appeler comme ça, alors que d'une part c'était un DEUG de Sciences de la vie, et que d'autre part, notre amitié n'est plus vraiment liée au DEUG en question, et qu'en plus une grande partie du groupe n'est pas issue du DEUG en question !) ; - la musique occupe une place importante dans ma vie, à cette liste de remerciements s'ajoutent les membres de l'Orchestre Moderne, de l'orchestre de Morsang-sur-Orge, bien sûr Geneviève Billaud mon professeur de violon, le quatuor Ouarézon ; - les exilés d'Allemagne, d'Angleterre ou de Suisse ; - les amis de colo (pas celles de mon enfance, celle en tant qu'animateur) ; - le groupe du club d'escalade.
Enfin, je finirai avec ma famille. Pour leur soutien indéfectible, leur amour, leur éducation, et leur résistance surhumaine à mon orthographe défaillante, je remercie de tout c$\frac{1}{2}$ur mes parents Kinou et Yvon. Enfin merci à mon frère Thibaud, célébré indirectement par le nom d'un des systèmes présentés dans cette thèse, compagnon de musique, de jeux (sur ordinateur de préférence), et confident.

Table des matières

Introduction

L'amélioration des techniques d'analyse à haut débit et l'accroissement constant des connaissances en biologie ont conduit à une explosion du volume des données biologiques disponibles et accessibles aux chercheurs sur internet. À titre d'exemple, les méthodes modernes de séquençage d'ADN peuvent produire en une seule expérience une quantité de résultats aussi importante en taille que le génome humain (3 milliards de paires de bases). Il y a encore peu de temps, il fallait plusieurs années à un consortium mondial pour obtenir une telle quantité de résultats.

Malheureusement, sans une expertise humaine, ces informations sont vides de sens. Pour donner une signification biologique à ces données, pouvoir les exploiter et en tirer des connaissances biologiques nouvelles, il faut les analyser à l'aide de diverses méthodes. Ces données doivent d'abord être assemblées en unités génétiques, en chromosomes par exemple, puis les gènes doivent être localisés sur ces chromosomes et associés aux produits des gènes (les protéines). Enfin, il faut associer aux protéines leur rôle biologique dans l'organisme. C'est cette dernière étape qui nous intéresse ici plus particulièrement : l'**annotation fonctionnelle**.

Nous nous sommes intéressés dans un premier temps à l'architecture en domaines des protéines qui a une influence sur leur rôle et leur fonction biologique. Les séquences protéiques sont habituellement comparées entre elles à l'aide d'algorithmes d'alignement. Étant donnée une certaine fonction protéique, les acides aminés des séquences provenant de plusieurs organismes sont mis en correspondance un à un. Les régions les plus régulières, c'est-à-dire les plus conservées à travers les espèces, définissent souvent des sites caractéristiques ou des domaines. De nombreuses sources de données de familles de protéines et de domaines protéiques ont été créées. Chacune d'elles repose sur un processus spécifique de sélection des membres de la famille et de définition de la famille. Les sources de données plus générales telles que Pfam [14] ont été conçues au départ autour de deux principes : (i) chercher à couvrir un maximum d'espèces et de séquences connues (ii) en supposant initialement l'indépendance entre les familles. Il en découle que le nombre de familles protéiques a considérablement augmenté au cours du temps.

La nécessité d'explorer ces architectures a conduit la majorité des sources de données de familles protéiques les plus célèbres à introduire des outils pour visualiser l'appartenance à de multiples familles. Une telle multiplicité reflète la variabilité des architectures en domaines des protéines. Quelques ressources sont conçues pour permettre des requêtes avancées ou la comparaison de ces architectures. Ce type de recherche est motivé par l'observation de co-occurrences dans les domaines des protéines en lien avec une propriété fonctionnelle. Par exemple, deux domaines structuraux peuvent co-occurrer pour maintenir une structure 3D, et conséquemment une fonction biologique. L'exploration systématique des relations possibles entre les domaines de protéines en terme d'architecture de domaines peut ainsi révéler des propriétés intéressantes. Aujourd'hui l'exploration des sources de données est guidée par l'intuition de l'utilisateur, mais

certaines régularités peuvent échapper à l'oeil pourtant entraîné du spécialiste forcé de parcourir des pages et des pages de représentations graphiques d'architectures. Nous présentons dans cette thèse une méthode d'exploration qui a conduit à l'outil SNK qui permet d'analyser systématiquement les combinaisons de domaines, en recherchant des règles d'association séquentielles particulières : les pépites séquentielles de connaissances. Nous avons réalisé à l'aide de cet outil une étude de deux familles protéiques, basée sur leur architecture en domaines.

Au cours de ce travail, nous avons dû utiliser les annotations que nous avons associées aux architectures en domaines des protéines. Mais les données de qualité, annotées à la main, étaient trop dispersées et dans des formats trop hétérogènes, de ce fait inutilisables. Cette étude a clairement mis en évidence le besoin d'une standardisation de l'annotation fonctionnelle.

L'association d'une fonction à une protéine est un processus complexe, entre autres parce que la fonction est un concept multi-facettes : pour une même protéine, il existe au moins une fonction moléculaire (son activité biochimique), une fonction cellulaire (son rôle dans les voies de signalisation de la cellule) et une fonction phénotypique (son influence sur l'ensemble de l'organisme). Qui plus est, il n'est pas rare qu'une même fonction soit décrite de différentes façons, ce qui conduit à de très nombreux synonymes. Par exemple, les termes "Transforming growth factor beta-1-induced transcript 1 protein", "Hydrogen peroxide-inducible clone 5 protein", et "Androgen receptor-associated protein of 55 kDa" décrivent la même fonction protéique.

Afin d'organiser les connaissances dans le champ dense et complexe de l'annotation fonctionnelle, les biologistes ont conçu des schémas de classifications biologiques. Les premiers schémas standardisés pour l'annotation fonctionnelle étaient de simples hiérarchies, dans lesquelles la spécificité de la description d'une fonction s'accroissait au fur et à mesure que l'on quittait la racine de la hiérarchie pour ses feuilles [96]. Essentiellement conçues pour l'annotation d'organismes précis, *Escherischia coli* pour MultiFun [95], et *Bacillus subtilis* pour Subtilist [83], ces hiérarchies étaient en partie biaisées par les caractéristiques des organismes qu'elles annotaient, en particulier parce que les deux organismes annotés étaient des bactéries. De plus, la conception de ces hiérarchies dépendait fortement des centres d'intérêt des biologistes qui les avaient conçues. Des hiérarchies plus générales ont alors été développées, qui pouvaient décrire les fonctions des protéines pour les organismes procaryotes aussi bien que pour les eucaryotes. Funcat [99], créée par le MIPS (Munich Information center for Protein Sequences), est un exemple d'une telle hiérarchie.

En parallèle des classifications hiérarchiques, différents schémas ont été proposés. Nous citerons par exemple, Gene Ontology [111], développée au départ pour comparer des annotations d'organismes multicellulaires tels que le ver *Caenorhabditis elegans*, la mouche du vinaigre *Drosophila melanogaster*, et d'autres (souris...). Organisée en graphe acyclique orienté, avec des relations un peu plus complexes que le simple héritage, GO est l'une des classifications les plus utilisées de nos jours.

Différents organismes ont été annotés manuellement en utilisant des classifications diverses. Comme nous devons comparer les annotations de différents organismes annotés avec différentes classifications, nous devons être capables d'établir des correspondances entre les composants de ces différentes classifications. Il nous faut construire un **mapping**.

La comparaison d'ontologies biologiques pour l'annotation fonctionnelle n'est pas immédiate puisque chacune d'entre elles reflète l'expertise et l'intérêt des scientifiques qui l'ont conçue. De ce fait, selon l'ontologie, certains aspects des fonctions protéiques sont plus détaillés que d'autres [99]. Ainsi, certaines ontologies proposent une description très fine des fonctions des produits

des gènes (par exemple GO), alors que d'autres sont plus concises (par exemple FunCat).

En informatique, de nombreuses techniques de mise en correspondances (mapping) d'ontologies ont été proposées durant les dix dernières années. Récemment, plusieurs études et livres ont entrepris de classifier les différentes techniques [34, 54] et un workshop a été organisé spécialement sur cette thématique [53].

Malheureusement, la spécificité des ontologies biologiques que nous souhaitons utiliser rend les approches classiques mal adaptées. En effet, celles avec lesquelles nous travaillons ont un relativement petit nombre de concepts, une structure simple (une hiérarchie avec seulement un type de relation), et le plus souvent, leurs concepts ne partagent pas d'instances communes.

Nous introduisons dans cette thèse O'Browser, un système qui utilise fortement les connaissances du domaine pour trouver un mapping fiable et utilisable par l'expert. Nous introduisons de nouvelles stratégies pour découvrir et sélectionner les correspondances candidates et pour combiner divers critères permettant d'établir ces correspondances.

Cette thèse est organisée de la façon suivante : tout d'abord nous présentons des notions de base de biologie moléculaire et un rapide tour d'horizon de quelques-unes des techniques de séquençage (chapitre 1). La première partie contiendra un état de l'art sur la recherche de règles d'association séquentielles (chapitre 2). Puis notre première contribution portera sur la définition de la notion de pépites séquentielles de connaissance, et la proposition d'un algorithme de recherche (SNK) pour trouver de telles pépites avec l'application à l'architecture en domaines protéiques (chapitre 3). La seconde partie sera consacrée aux ontologies biologiques et aux méthodes d'alignement d'ontologies biologiques ou non (chapitre 4). Le chapitre 5 présentera ensuite notre contribution à l'alignement d'ontologies biologiques : le système O'Browser. Enfin, nous conclurons et présenterons les perspectives de ce travail.

Chapitre 1

Protéines et annotation fonctionnelle

Sommaire

1.1 De l'ADN aux protéines

Rendu populaire par ses applications policières grâce à Hollywood, l'ADN est l'une des molécules centrales de la biologie. Siège du patrimoine génétique, cette molécule recèle toutes les informations nécessaires au développement et au maintien de la vie.

L'**ADN** (acide désoxyribo nucléique) est constitué d'une chaîne de nucléotides (parfois appelées bases azotées) dont la longueur peut varier de quelques centaines de bases à plusieurs dizaines de milliers ou millions. Dans la cellule, l'ADN est organisé. Il prend certaines formes dont la plus célèbre est la double hélice. En effet les bases azotées, au nombre de quatre : adénine (A), cytosine (C), guanine (G), et thymine (T), peuvent s'apparier deux à deux (formant une paire de bases). Il en résulte une macromolécule organisée sous la forme d'une double hélice constituée de deux chaînes d'ADN (voir figure 1.1).

Pour passer de la molécule support de l'information, l'ADN, à la protéine active dans la cellule, une autre macromolécule biologique joue un rôle très important : l'**ARN** (acide ribo nucléique).

Les scientifiques ont longtemps minimisé le rôle de l'ARN. On sait aujourd'hui que cette molécule peut occuper un très grand nombre de rôles dans le développement et le maintien de la vie, puisqu'elle peut jouer à la fois un rôle de support de l'information génétique et un rôle actif de catalyseur dans les réactions biologiques. Du fait de ces rôles multiples, il existe une très grande variabilité dans les molécules d'ARN. Nous ne nous intéresserons ici qu'à l'ARN messager (ARNm) qui permet de transporter l'information génétique de l'ADN à l'endroit de la cellule où seront synthétisées les protéines. L'ARNm est une copie d'une portion d'un des brins d'ADN :

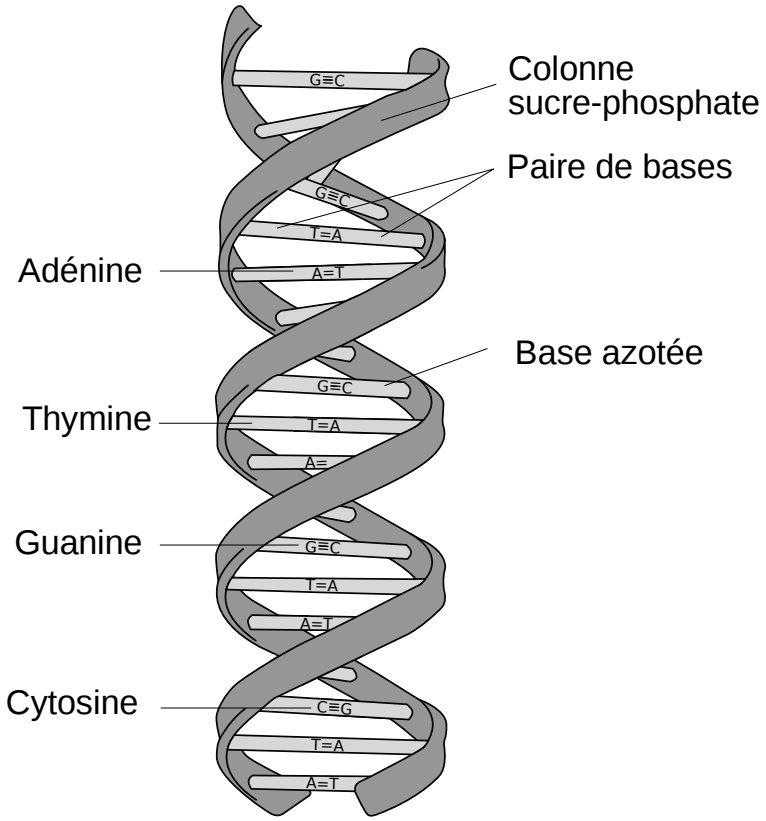


FIG. 1.1 – *Deux brins d'ADN appariés*

il est donc composé d'une chaîne de nucléotides (les mêmes que l'ADN sauf pour la thymine qui est remplacée par l'uracile (U)). La synthèse de l'ARNm à partir de l'ADN se nomme la transcription.

Chez les eucaryotes, l'ARNm a subi des transformations pour pouvoir être traduit en protéine. L'ADN est transcrit en un ARN pré-messager, qui va "maturer" pour donner l'ARNm, traduit lui-même en protéine. L'un des principaux événements de la maturation est l'**épissage**, qui enlève les portions de l'ARN pré-messager qui ne seront pas traduites en protéine. C'est le reflet de la présence dans le gène d'**introns** et d'**exons**. Les introns sont les parties du gène qui seront enlevées lors de l'épissage tandis que l'exon contient l'information génétique utilisée pour fabriquer la protéine.

Troisième grande catégorie de macromolécules biologiques, les **protéines** sont connues de tous, scientifiques ou non. Leur impact dans l'alimentation, que ce soit pour les sportifs, les régimes ou même leur présence sur les informations nutritionnelles de tout emballage de nourriture, les a rendues célèbres bien au-delà des spécialistes. Les protéines sont l'un des constituants essentiels de la cellule, puisqu'elles assurent à la fois le fonctionnement et la structure cellulaire. Le terme protéine vient du grec *prôtos* qui signifie *premier*, *essentiel* et a été introduit pour la première fois en 1839 par G.J. Mulder [84] qui a détecté leur rôle premier dans la cellule.

Les protéines sont consistuées par un enchaînement linéaire d'**acides aminés**. Elles sont constituées de séquences plus ou moins longues de ces acides aminés, et selon le nombre pourront être appelés protéines (pour les plus grandes séquences), polypeptides ou peptides (pour les plus petites). Les rôles des protéines peuvent être très variés. Elles interviennent dans le maintien de la structure cellulaire, dans la catalyse de réactions biologiques, dans la régulation de l'expression des gènes, etc. Le ou les rôles d'une protéine sont appelés sa ou ses fonctions. La détermination de la fonction est l'annotation fonctionnelle, nous y reviendrons un peu plus loin dans ce chapitre.

L'étape de synthèse des protéines à partir de l'ARN messager se nomme la traduction. Elle permet de fabriquer une (parfois plusieurs, chez les virus par exemple) protéine constituée d'une chaîne d'acides aminés à partir d'une séquence d'ARNm. Les acides aminés les plus courants sont au nombre de 20. L'étape de traduction de l'ARN en protéine permet donc le passage d'une information codée dans un alphabet à 4 lettres (A, C, G et U) à un alphabet à 20 lettres. C'est Crick et collègues [25] en 1961 qui ont émis l'hypothèse d'une information codée par des triplets dans l'ADN, confirmée depuis. Le **code génétique** (tableau 1.1) permet de passer de 64 triplets à 20 acides aminés. Il faut noter qu'un triplet ne correspond qu'à un seul acide aminé, alors qu'un acide aminé correspond à plusieurs triplets. C'est la dégénérescence du code génétique.

Il y a 6 façons de lire une séquence d'ADN puisque les nucléotides sont lus 3 par 3. En commençant à un nucléotide (n), ou en décalant d'un nucléotide ($n+1$) ou de deux ($n+2$), le décalage d'un nucléotide supplémentaire nous ramène dans le premier cas. On a la même chose avec le brin complémentaire. On parle de 6 phases de lecture de l'ADN. Les gènes peuvent être répartis sur ces 6 phases sans préférences.

1.1.1 Séquençage

L'accès à l'information génétique est un challenge ancien aujourd'hui élucidé : l'une des grandes étapes fut la découverte de la structure de l'ADN en 1953 par Watson, Crick et Wilkins grâce entre autres aux travaux de Rosalind Franklin. Il faut attendre 1975 pour que les premières techniques permettant le séquençage de cette macromolécule fassent leur apparition avec les

	U		C		A		G		
U	UUU	phénylalanine	UCU	sérine	UAU	tyrosine	UGU	cystéine	U
	UUC		UCC		UAC		UGC		C
	UUA	leucine	UCA		UAA	stop	UGA	stop/sélénocystéine	A
	UUG		UCG		UAG		UGG	tryptophane	G
C	CUU		CCU	proline	CAU	histidine	CGU	arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	isoleucine	ACU	thréonine	AAU	asparagine	AGU	sérine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	lysine	AGA	arginine	A
	AUG	méthionine/start	ACG		AAG		AGG		G
G	GUU	valine	GCU	alanine	GAU	acideaspartique	GGU	glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	acideglutamique	GGA		A
	GUG		GCG		GAG		GGG		G

TAB. 1.1 – *Code génétique*

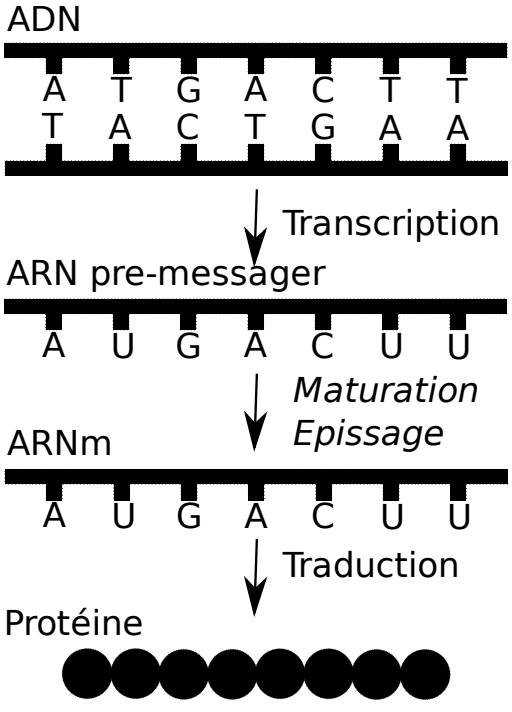
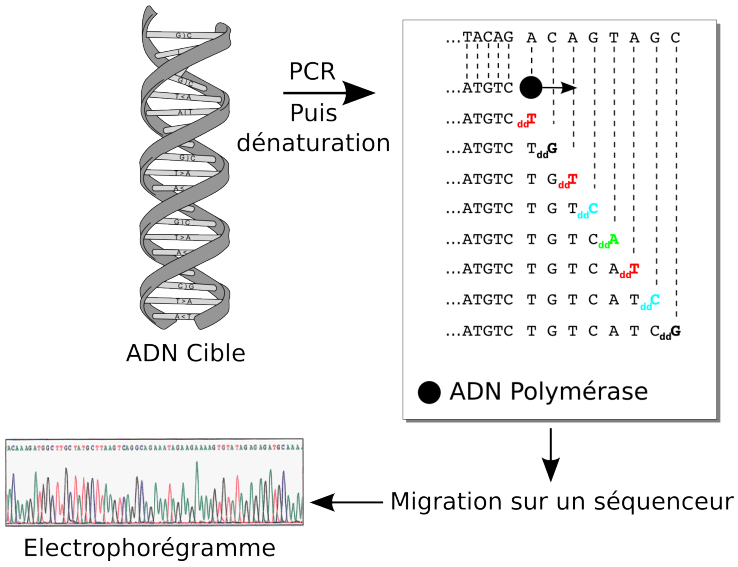


FIG. 1.2 – *Le dogme central de la biologie, passage de l'ADN aux protéines*

FIG. 1.3 – *Méthode de Sanger*



travaux de Sanger et la méthode "plus and minus" [102]. C'est cette méthode qui permettra notamment le séquençage de la première séquence : le bactériophage (virus de bactérie) φX174 en 1977.

Cette même année deux nouvelles méthodes voient le jour, développées indépendamment par Maxam et Gilbert d'une part et l'équipe de Sanger d'autre part. Ces travaux leur vaudront le prix Nobel de chimie en 1980.

Technique de Maxam-Gilbert Pour cette technique [74], la molécule d'ADN est clivée - découpée - au niveau de certaines bases spécifiques (C, G, A+G et C+T). L'extrémité des séquences est marquée radioactivement. Les portions d'ADN sont ensuite séparées par électrophorèse, c'est-à-dire par séparation des molécules dans un champ électrique. Le résultat est révélé par radiographie et permet de "lire" la séquence. Cette méthode n'est plus utilisée de nos jours.

Les méthodes suivantes nécessitent des quantités importantes de la séquence pour permettre un séquençage complet. Pour cela les scientifiques peuvent utiliser la PCR (Polymerase Chain Reaction). Il s'agit d'un procédé permettant l'amplification d'une région spécifique d'une molécule d'ADN ou d'ARN (l'amplification est l'augmentation du nombre de copies de cette région). On réalise une succession de phases de rechauffement et refroidissement de la solution contenant la séquence nucléotidique. Les deux brins sont séparés en chauffant, c'est la dénaturation. Puis l'ajout d'une enzyme l'ADN polymérase en présence de désoxyribo-nucléotides (dNTP) permet la synthèse de nouveaux brins d'ADN en utilisant l'un des deux brins originaux comme une matrice. Dans le cas de la méthode de Sanger que nous présenterons dans le prochain paragraphe, avant le développement de la PCR, les biologistes faisaient appel à des techniques différentes utilisant par exemple des bactéries comme vecteurs de clonage de la séquence.

Méthode de Sanger La méthode de Sanger [103] se déroule de la façon suivante : les deux brins d'ADN de la double hélice sont séparés en chauffant le mélange. Une enzyme,

l'ADN polymérase est ajoutée en présence de désoxyribo-nucléotides (dNTP) et de didésoxyribo-nucléotides (ddNTP) marqués chimiquement par une molécule fluorophore (voir figure 1.3). Les dNTPs peuvent être utilisés par la polymérase pour allonger la séquence, tandis que les ddNTPs bloquent la polymérisation. Les séquences produites se terminent ainsi par un ddNTP. Les quatre nucléotides A, C, G et T sont traités à part, et des séquences arrêtées à chaque occurrence du nucléotide dans la séquence devraient être obtenues. Après migration de ces séquences par électrophorèse, les résultats vont pouvoir être lus en repérant les ddNTP fluorophores à l'aide d'un laser. Cette méthode permet le séquençage de portions d'ADN de taille relativement réduite.

D'autres techniques plus récentes ont été développées et ont permis notamment le séquençage de génomes complets.

Séquençage de génomes complets

Technique du "Shotgun" La première étape consiste à découper un génome entier en petits fragments de manière aléatoire. Ces fragments sont amplifiés par PCR et séquencés par la méthode de Sanger. Les séquences obtenues sont analysées pour reconstituer le génome initial. Pour obtenir une bonne couverture, et permettre une reconstitution de qualité, cette méthode nécessite de séquencer l'équivalent d'au moins 5 fois le génome initial.

Pyroséquençage Cette technique [97] a tendance à remplacer progressivement la technique de Sanger. L'ADN est dénaturé, puis l'ADN polymérase est utilisée pour synthétiser un nouveau brin d'ADN à partir des brins matrices. C'est au cours de cette synthèse que le séquençage va être réalisé. Des nucléotides sont ajoutés au milieu de réaction, un par un. Un système de réaction physicochimique permet de détecter via des caméras CCD lorsque le nucléotide allonge effectivement la séquence (voir figure 1.4). Le séquençage est effectué en temps réel, au fur et à mesure de l'ajout des nucléotides à la séquence. Cette technique ne permet de séquencer que des séquences courtes (400 nucléotides de long), il faut ensuite être capable d'assembler ces différents fragments en unités génétiques (chromosomes...). C'est là l'une des étapes critiques du séquençage. L'existence d'automates capables d'un séquençage massif et parallèle a conduit au développement de cette technique à une échelle plus importante.

Une description plus complète des méthodes de séquençage pourra être trouvée dans [64]. La taille de fragments séquencés et le nombre de fragments séquencés en une expérience sont les éléments critiques. Les méthodes en cours de développement travaillent actuellement à l'amélioration de ces points.

Grâce à ces méthodes et à leur amélioration constante, le nombre de génomes séquencés ne cesse d'augmenter. À ce jour, le nombre de génomes complets recensés par le NCBI (National Center for Biotechnology Information) est de 1049, et ce nombre continue à croître de façon importante. L'accès à de telles quantités de données permet d'envisager des recherches jusqu'alors difficiles, voire impossibles, mais demande aussi beaucoup de moyens d'expertise et d'analyse. Dans la prochaine section, nous verrons une technique bio-informatique qui peut être utilisée pour analyser en partie les résultats de ces séquençages.

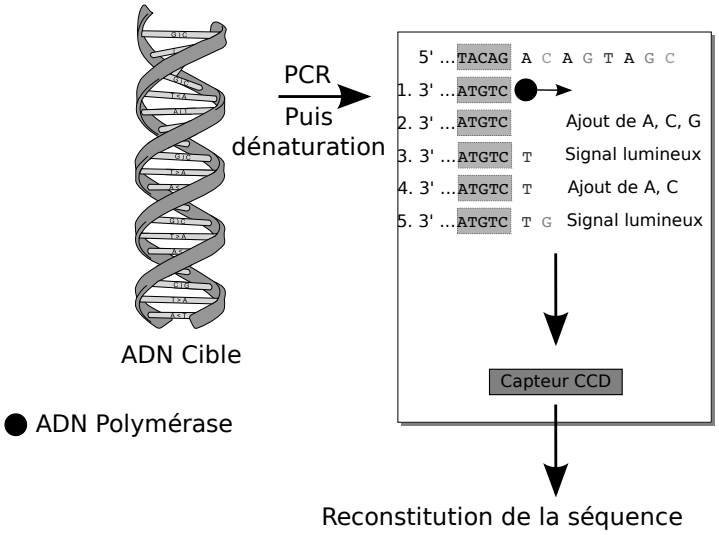


FIG. 1.4 – *Pyroséquençage*

1.1.2 Comparaisons d'objets biologiques

Dans la section précédente, nous avons vu succinctement comment l'information génétique est stockée et utilisée par les organismes. Cette information est transmise d'un ancêtre à ses descendants, et peut être soumise à des mécanismes qui peuvent la modifier : mutations ponctuelles, réorganisation partielle, délétion d'une partie de l'information... Ces modifications peuvent supprimer des fonctionnalités de la cellule, les modifier, ou en créer de nouvelles. C'est l'ensemble de ces phénomènes associés à la sélection naturelle qui forment l'évolution [26].

En comparant les gènes ou les protéines présents de nos jours dans les organismes, il est possible d'essayer de reconstruire leur histoire évolutive passée. Pour cela on cherche à identifier les protéines homologues, c'est-à-dire deux protéines descendant d'une même protéine ancestrale. L'homologie peut être décomposée en plusieurs cas, dont deux nous intéressent particulièrement : l'orthologie et la paralogie. Deux protéines sont dites orthologues quand elles descendent d'une même protéine ancestrale et sont issues d'un événement de spéciation. Deux protéines sont paralogues quand elles sont issues d'une duplication ancestrale. Dans les deux cas, les protéines peuvent garder des ressemblances aussi bien au niveau de leurs rôles biologiques qu'au niveau de leurs séquences nucléotidiques.

Bien que ce ne soit pas toujours le cas, il est admis que deux protéines orthologues ont de grandes chances d'avoir des fonctions similaires.

Pour identifier des proximités entre des séquences biologiques, la méthode la plus communément utilisée aujourd'hui est la comparaison de séquences. L'hypothèse est que deux séquences sont d'autant plus proches biologiquement et évolutivement qu'elles se ressemblent syntaxiquement. Les figures 1.6 et 1.7 présentent deux alignements : l'alignement d'une protéine humaine (*Homo sapiens*) avec une protéine de la souris (*Mus musculus*), et l'alignement de cette même protéine humaine avec une espèce évolutivement moins proche de l'Homme que la souris, un poisson *Danio rerio*. La ressemblance entre les séquences est illustrée par les rectangles gris qui encadrent

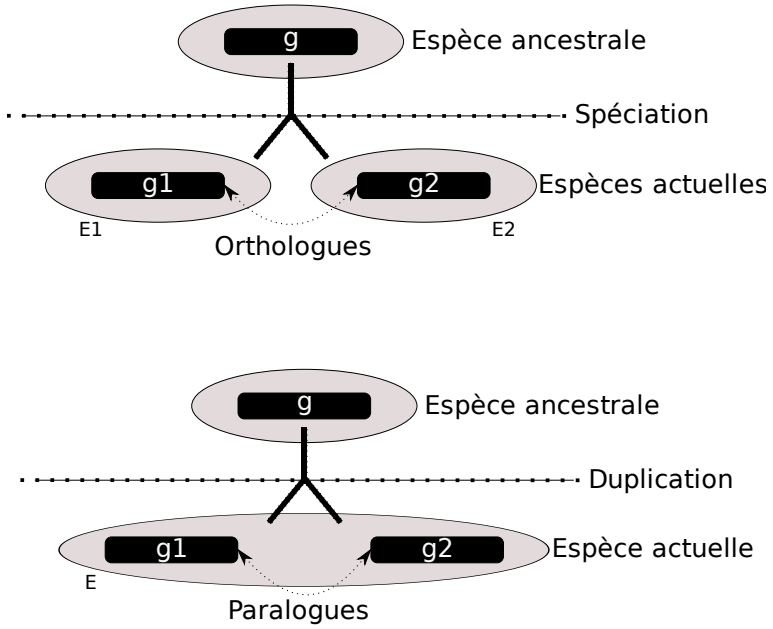


FIG. 1.5 – *Homologie, paralogie*

l'alignement. Plus l'alignement est fort (et donc plus les séquences se ressemblent) et plus les rectangles gris sont étendus. Il est flagrant sur cet exemple que les séquences de l'Homme et de la souris sont plus ressemblantes que les séquences de l'Homme et de Danio rerio. Dans la pratique pour la constitution d'arbre d'espèces, ou pour déterminer la proximité entre deux espèces, les scientifiques ne se contentent pas d'un alignement simple comme présenté ici, mais recherchent beaucoup plus d'informations, afin d'éviter les obstacles qui pourraient être posés par les alignements ou des phénomènes biologiques particuliers comme l'existence de transferts horizontaux (transfert de matériel génétique par des mécanismes non héréditaires entre espèces vivantes), la fusion ou la fission de gènes par exemple.

Malgré ces restrictions, il est parfois possible d'inférer la fonction d'une protéine grâce aux

```
Q9Y5Y5   MEKLRLLGLRYQEYVTRHPAATAQLETAVRGFSYLLAGRFADSHELSELVYSASNLLVLL  60    PEX16_HUMAN
Q91XC9   MEKLRLLSLRYQEYVTRHPAATAQLETAVRGLSYLLAGRFSDSHELSELVYSASNLLVLL  60    PEX16_MOUSE
         *******.**********************:********:******************

Q9Y5Y5   NDGILRKELRKKLPVSLSQQKLLTWLSVLECVEVFMEMGAAKVWGEVGRWLVIALIQLAK  120   PEX16_HUMAN
Q91XC9   NDGILRKELRKKLPVSLSQQKLLTWLSVLECVEVFMEMGAAKVWGEVGRWLVIALIQLAK  120   PEX16_MOUSE
         ************************************************************

Q9Y5Y5   AVLRMLLLLWFKAGLQTSPPIVPLDRETQAQPPDGDHSPGNHEQSYVGKRSNRVVRTLQN  180   PEX16_HUMAN
Q91XC9   AVLRMLLLIWFKAGIQTSPPIVPLDRETQAQPLDGDHNPGSQEPSYVGKRSHRVVRTLQN  180   PEX16_MOUSE
         ********:*****:**************** ****.**.:* *******:********

Q9Y5Y5   TPSLHSRHWGAPQQREGRQQQHHEELSATPTPLGLQETIAEFLYIARPLLHLLSLGLWGQ  240   PEX16_HUMAN
Q91XC9   SPSLHSRYWGAPQQREIRQKQQQEELSTPPTPLGLQETIAESLYIARPLLHLLSLGLWGQ  240   PEX16_MOUSE
         :******:******** **:*:::****:.************ ******************

Q9Y5Y5   RSWKPWLLAGVVDVTSLSLLSDRKGLTRRERRELRRRTILLLYYLLRSPFYDRFSEARIL  300   PEX16_HUMAN
Q91XC9   RSWTPWLLSGVVDMTSLSLLSDRKNLTRRERLELRRRTILLLYYLLRSPFYDRFSEAKIL  300   PEX16_MOUSE
         ***.****:****:**********.****** *************************:**

Q9Y5Y5   FLLQLLADHVPGVGLVTRPLMDYLPTWQKIYFYSWG  336   PEX16_HUMAN
Q91XC9   FLLQLLTDHIPGVGLVARPLMDYLPSWQKIYFYSWG  336   PEX16_MOUSE
         ******:**:******:********:**********
```

FIG. 1.6 – *Alignement de deux séquences proches (Peroxisomal membrane protein PEX16 de Homo sapiens et Mus musculus)*

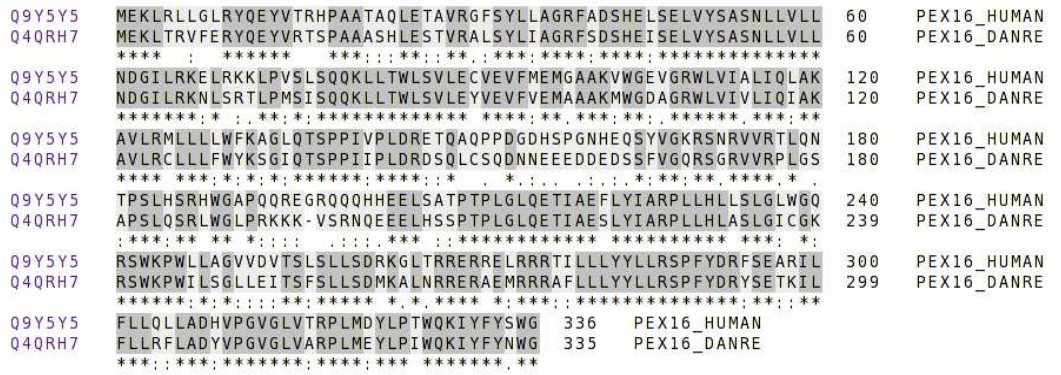


FIG. 1.7 – *Alignement de deux séquences moins proches (Peroxisomal membrane protein PEX16 de Homo sapiens et Danio rerio)*

fonctions des protéines qui lui ressemblent beaucoup (au sens de la proximité des séquences).

Un outil de recherche de séquences similaires très utilisé est BLAST [10]. Il s'agit d'un logiciel d'alignement de séquences utilisant une heuristique permettant une grande rapidité dans les réponses. Étant donné une séquence d'intérêt, il permet de trouver dans une base de données de séquences, les séquences les plus proches syntaxiquement. L'algorithme commence en découpant la séquence donnée dans la requête en sous-chaînes de longueur fixée (souvent 3 dans les cas d'une séquence d'acides aminés et 11 dans le cas de séquences nucléiques). Pour sous-chaîne m, il identifie la liste l des mots de même longueur qui lui sont proches dans l'ensemble de la base de donnée comparée. BLAST identifie ensuite les occurrences exactes des mots de la liste l dans les séquences de la base de donnée, puis pour chaque occurrence essaie d'étendre l'alignement dans les deux directions. Si l'extension du "noyaux" conduit à un alignement dont la qualité diminue trop, BLAST abandonne cette occurrence et passe à la suivante.

Deux valeurs sont associées à un résultat d'alignement BLAST. La première indique le pourcentage de paires de bases conservées (100 % indiquant que les deux protéines sont identiques dans leur composition en acides aminés). L'autre valeur est la e-value, qui indique le nombre de séquences qui peuvent être trouvées par hasard dans la base de données ayant servi à comparer la séquence initiale. Plus ce nombre est petit moins le risque est élevé que l'alignement soit dû au hasard et plus l'alignement est pertinent (par exemple des valeurs de 10^{-4} et inférieures sont souvent jugées "bonnes").

Il existe plusieurs versions de BLAST, toutes basées sur le même fonctionnement, mais portant des noms différents selon les objets biologiques qu'elles comparent ; dans le cas de comparaison protéine-protéine, il est commun d'utiliser la version BLASTp.

1.2 Annotation structurale

Une fois la séquence du génome obtenue, il est nécessaire de donner du sens à cette information. La première étape est souvent l'**annotation structurale** de la séquence. Il s'agit le plus souvent de positionner dans le génome les gènes et leurs produits (transcrits, protéines) ainsi que d'autres objets biologiques : motifs de régulation, éléments transposables, etc., mais aussi les gènes codant

pour des ARNt, ARNr, et bien d'autres encore.

L'identification des positions se fait par des méthodes bioinformatiques. Il existe différentes méthodes : la méthode expérimentale, les méthodes comparatives, les méthodes *ab initio* et les méthodes intégratives qui combinent les précédentes. Dans ce paragraphe nous allons détailler succinctement les méthodes citées.

Méthodes expérimentales et comparatives Expérimentalement, un brin d'ADN, copie des brins d'ARNm qui seront traduits en protéines, est produit (on crée une banque d'ADN complémentaire : l'ADNc). Il est aligné sur la séquence complète du génome et permet ainsi d'identifier les gènes par comparaisons des séquences. Cette méthode est la plus sûre car expérimentale, mais se heurte malgré tout à des problèmes en pratique, en particulier en raison de l'existence de petits exons et au fait que l'identification complète du gène n'est pas garantie.

L'annotateur peut comparer la séquence génomique qu'il étudie avec 3 types d'objets biologiques :
- La séquence génomique est comparée aux ESTs dont on dispose. C'est l'étape détaillée ci-dessus.
- La séquence est traduite sur les 6 phases de lecture, et les résultats obtenus sont comparés aux protéines des sources de données publiques. L'identification d'une protéine permet d'identifier la portion de séquence la codant.
- La séquence génomique d'intérêt est comparée aux séquences génomiques d'espèces proches. On espère pouvoir identifier des gènes proches, et ainsi les positionner.

Méthodes *ab initio* On cherche à reconnaître les particularités communes à tous les gènes d'un génome, puis on cherche à les retrouver sur l'ensemble de la séquence. Pour cela on utilise un modèle statistique, ou autre, de ce qu'est un gène.

Une présentation très complète et détaillée de ces méthodes est exposée dans [73].

1.3 Annotation fonctionnelle

La section précédente a montré comment les progrès techniques ont permis d'accéder à une information de plus en plus précise sur les séquences biologiques (ADN, ARN et protéiques). Ces données seules ne sont pas très informatives. Une étape capitale du post-séquençage d'un génome est l'**annotation fonctionnelle**. Il s'agit de la tâche qui consiste à associer à une protéine sa ou ses fonctions biologiques.

L'annotation peut être réalisée expérimentalement ou *in silico*. Expérimentalement, les biologistes réalisent des manipulations, par exemple des mutagenèses au sein des protéines pour en perturber l'expression, et peuvent ainsi caractériser la fonction en observant les fonctionnalités perdues par l'organisme. Cette technique a bien sûr des limites, il n'est pas possible par exemple de réaliser ce type d'expérimentation sur des protéines essentielles à la survie de l'organisme. D'autres techniques expérimentales existent donc. Elles ont en commun de caractériser la ou les fonctions d'une protéine avec une grande précision, mais au prix d'un temps long et d'un coût financier parfois important. Avec l'essor des techniques de séquençage à haut débit, de nouvelles techniques bio-informatiques ont fait leur apparition et se sont rapidement montrées

indispensables, avec toutefois une limite importante qui réside dans le fait qu'elles ne puissent identifier que des éléments déjà connus.

Aujourd'hui, le plus souvent, l'annotation fonctionnelle est réalisée *in silico*, par comparaison de séquences. Le bioinformaticien recherche dans les sources de données publiques des protéines proches. Si les fonctions de ces protéines sont déjà connues, alors l'annotateur peut, selon certaines conditions, propager cette information à la protéine de fonction inconnue.

De nombreuses informations entrent en compte dans la détermination de la fonction, nous n'avons détaillé ici que les aspects liés aux comparaisons de séquences, mais les paramètres peuvent être très variés : charge électrique de la protéine (point isoélectrique), hydrophobicité, présence de signatures spécifiques. Enfin, si les acides aminés sont les briques élémentaires des protéines, il est néanmoins possible d'identifier des blocs conservés entre les différentes protéines de différents organismes, les domaines protéiques. Nous y reviendrons dans la prochaine section (section 1.4). L'organisation de ces différents domaines peut aussi être une information pertinente pour associer à une protéine une fonction biologique. Information de qualité, la conservation de l'ordre des gènes - **synténie** - peut être également un excellent indicateur pour les scientifiques. Si l'ordre des gènes autour d'un gène d'intérêt a été conservé au cours de l'évolution, il est possible de comparer ce gène à ceux qui occupent la même place dans les autres organismes. Si leur fonction est connue, il est envisageable de la transférer à la protéine d'intérêt.

Cette étape est réalisée de manière semi-automatique et peut être longue et constitue encore un goulet d'étranglement. A titre d'exemple, le protéome humain est complétement annoté depuis la fin de l'année 2008, il contient encore entre 30 et 35 % de protéines de fonction inconnue, dont 10 à 15 % pour lesquelles aucune similarité avec des protéines connues n'a été identifiée. Mis a part quelques organismes modèles, ces proportions se retrouvent dans un grand nombre d'organismes séquencés et annotés.

De nombreuses recherches sont en cours pour automatiser au mieux l'annotation (voir par exemple [11]).

Nous verrons dans la partie 4, que le choix des termes décrivant la fonction des protéines est crucial, et est une question importante de l'annotation fonctionnelle.

1.4 Domaines protéiques

Sous l'appellation **domaine protéique**, nous regrouperons des concepts biologiquement différents : les domaines protéiques, les motifs protéiques et les zones de faible complexité. La définition des domaines n'est d'ailleurs pas homogène entre les différentes banques de données. Le point commun de ces différentes notions est qu'elles concernent une sous-partie de la séquence protéique, que nous appellerons domaine protéique.

Les domaines protéiques nous permettent de décrire une protéine à un nouveau niveau de granularité. Elle est décrite non plus comme une chaîne d'acides aminés, mais comme une séquence de domaines. Il est prouvé aujourd'hui que la constitution en domaines d'une protéine a une influence très forte sur sa fonction [37]. Dans le chapitre 2, nous explorerons l'intérêt que l'ordre de ces domaines peut avoir pour la fonction.

Habituellement, la notion de domaine protéique est associée à une partie de la séquence qui forme une structure compacte, indépendamment du reste de la chaîne d'acides aminés. La plupart des

grosses protéines sont composées de plusieurs domaines, et ceux-ci sont souvent associés à des fonctions biologiques précises. La comparaison de protéines entre elles permet d'identifier ces domaines, grâce à leur conservation dans l'histoire évolutive des protéines. Une source de donnée biologique contenant des informations sur les domaines est Pfam [14]. Dans Pfam les domaines protéiques sont de deux types : des domaines détectés par un alignement multiple vérifié par un expert, ainsi que de domaine dont la fonction biologique est connue, regroupés sous le nom de Pfam-A, et des domaines détectés uniquement par alignement de séquences, nommés Pfam-B.

Les **motifs protéiques** sont d'une autre nature, ils sont dus à la conservation d'une fonction et peuvent être beaucoup plus dégénérés. Il s'agit d'une organisation spatiale de la chaîne d'acide aminé qui va conduire à la présence d'un motif dans la chaîne primaire. On trouve par exemple dans ces motifs les "doigts de zinc". Le motif va pouvoir être caractérisé par une séquence consensus : par exemple `C-x(2-4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H` qui signifie que le motif est caractérisé par la présence d'une cystéine (C), puis de 2 à 4 acides aminés quelconques (x(2,4)) puis à nouveau d'une cystéine... Une source de donnée biologique contenant des motifs est Prosite [51].

Enfin, nous utiliserons une dernière information, les **régions de faible complexité**, qui ne sont ni des domaines protéiques au sens strict, ni des motifs protéiques, mais des zones de la protéine avec des répétitions de fragments très courts. Ces régions peuvent provoquer des erreurs dans les alignements de séquences et sont souvent filtrées [116].

Les domaines que nous avons présenté ici sont ceux qui nous intéresserons dans le reste de ce manuscrit. D'autres types de domaine pourraient être envisagés et utilisés.

Dans le reste du manuscrit et en particulier dans la première partie, nous considérerons souvent les protéines comme une séquence de ces domaines, motifs ou régions. Nous les nommerons globalement "domaines protéiques".

1.5 Projet RAFALE

Une partie du travail de cette thèse s'inscrit dans le cadre du projet RAFALE (Règles pour l'Annotation Fonctionnelle semi-Automatisée de la LEvure). Il s'agit d'un projet ACI IMP Bio, ayant pour but d'assister les biologistes dans les différentes tâches liées à l'annotation, en utilisant notamment des techniques d'apprentissages.

Les laboratoires impliqués dans ce projet sont : l'IBBMC (Institut de Biochimie et de Biophysique Moléculaire et Cellulaire), CNRS Université Paris-Sud XI, l'unité MIG (Mathématique Informatique et Génome), de l'INRA Jouy-en-Josas, et le LRI (Laboratoire de Recherche en Informatique), CNRS Université Paris-Sud XI. Le projet a un collaborateur scientifique, l'IGM (Institut de Génétique et Microbiologie), CNRS Université Paris-Sud XI.

L'annotation fonctionnelle des gènes d'organismes procaryotes ou eucaryotes est un aspect qui intéresse particulièrement l'équipe MIG de l'INRA. Dans ce but, ils ont conçu et developpé une plateforme dédiée à l'annotation : la plateforme AGMIAL [21].

La plateforme AGMIAL utilise des données variées pour l'annotation de gènes : les relations d'homologies, la structure 3D des protéines, les alignements multiples, les domaines protéiques, les motifs (domaines transmembranaires...), le voisinage de gènes, les profils phylogénétiques... Ces données proviennent de sources diverses telles que UniProt, KEGG ou Genome Review et sont

intégrées au sein de deux modules PAM (Protein Annotation Manager) et CAM (Contig Analysis Manager) ainsi que dans l'entrepôt Microbiogenomics [66]. Les bases de données contiennent aussi des données privées de l'équipe.

L'annotateur dispose ainsi d'une quantité importante d'information et peut attribuer à une protéine inconnus sa ou ses fonctions à l'aide des informations à sa disposition, en particulier les protéines proches et le voisinage de gènes. Il est invité à saisir l'annotation d'une part dans un champ libre à sa disposition, et d'autre part à remplir un autre champ à l'aide d'un vocabulaire contrôlé et structuré : Subtilist (voir section 4.1.2). Le système ne permet l'attribution que d'une seule annotation Subtilist par protéine.

Le projet RAFALE a pour objectif d'assister les biologistes pour l'annotation fonctionnelle en concevant et en mettant au point un système à base de règles qui reflète les connaissances des experts et qui permette l'annotation des données génomiques par une utilisation optimale des informations disponibles. Le système est semi-automatique dans le sens où le processus est collaboratif : les annotations sont suggérées par l'application de règles qui reflètent des annotations connues pour d'autres protéines, mais elles sont finalement validées par des biologistes. Deux principaux cadres de travail ont été adoptés pour ce projet, une approche par arbre de décision du premier ordre et une approche par arbre de décision attribut-valeur multilabel.

Les travaux présentés dans cette thèse se placent dans le contexte de ce projet à plusieurs niveaux. Dans un premier temps, nous souhaitons établir des liens entre l'architecture des domaines protéiques et une propriété de la protéine. Cette propriété peut aussi bien être sa fonction biologique, que l'appartenance à une famille protéique. Les informations apportées par l'organisation des domaines peuvent donner des indications supplémentaires à l'annotateur pour mener sa tâche à bien.

Dans un second temps, c'est au niveau des ontologies biologiques utilisées pour l'annotation que nous interviendrons. Les données annotées manuellement sont de très grande valeur. Reflet des connaissances de l'expert, elles sont crédibles et utilisables. L'écueil à leur utilisation réside seulement dans la faible quantité de ces données. De plus les différentes équipes d'annotateurs dans le monde n'utilisent pas les mêmes vocabulaires contrôlés pour l'annotation pour des raisons que nous développerons plus tard. Le développement de méthodes permettant l'intégration des annotations manuelles hétérogènes est alors très intéressant, aussi bien dans le but d'augmenter les jeux d'apprentissage des algorithmes automatiques, que pour donner plus d'information à l'expert humain. C'est ce second problème que nous étudierons dans la seconde partie de cette thèse.

Première partie

Partie 1 : Règles d'association pour l'exploration fonctionnelle de familles protéiques

Chapitre 2

Recherche de pépites de connaissance

Sommaire

La recherche de régularités dans de très grandes quantités de données a connu son essor au début des années 90. Avec l'amélioration des technologies de code-barre, il a été possible de réunir de grandes quantités d'information sur les achats de clients dans des bases de données. L'identification de groupes d'objets achetés souvent simultanément présentait un grand intérêt, dans les secteurs de la grande distribution, pour l'optimisation des plans d'organisation des magasins par exemple. Nous présenterons les formalisations utilisées pour répondre à cette question dans la section 2.1 : les motifs fréquents et les règles d'associations.

Les scientifiques se sont ensuite intéressés à identifier des motifs dans des séquences de transactions, permettant ainsi de considérer les transactions non plus indépendamment les unes des autres, mais avec une donnée temporelle, en considérant les transactions les unes après les autres. Chaque client peut être identifié par une carte de fidélité, et les achats successifs sont stockés dans une base de données. Une solution pour formaliser cette question est présentée dans la section 2.2 : les motifs séquentiels fréquents.

2.1 Motifs fréquents et règles d'association

Nous reprenons les définitions et les notations de [6].

Définition 2.1.1 (Item et itemset) Soit $\mathcal{I} = i_1, i_2, ..., i_m$ un ensemble de littéraux, appelés *items*.
Nous appelons un *itemset*, un ensemble d'items T tel que $T \subseteq \mathcal{I}$.

Définition 2.1.2 Soit $\mathcal{D}$, une base de données, constituée d'un ensemble de n enregistrements. Chaque enregistrement est constitué d'un itemset. Un enregistrement T *contient* un itemset X, si $X \subseteq T$.

Définition 2.1.3 (Motif) Un *motif* M est un ensemble d'items tel que $M \subseteq \mathcal{I}$. Nous dirons qu'un motif M est *contenu* par un enregistrement T si $M \subseteq T$.

Exemple :
Les notions de cette section seront illustrées sur la base de données $\mathcal{D}_E$ suivante :

Enregistrement
A B C D E
A B C E
C
B C F

$\mathcal{D}_E$ contient 4 enregistrements.
Le motif $\{A, F\}$ n'est contenu par aucun enregistrement de la base de données $\mathcal{D}_E$.
Le motif $\{B, C\}$ est contenu par 3 enregistrements de la base de données $\mathcal{D}_E$.

Étant donné un motif fréquent, il peut être intéressant de chercher des implications du type "si X est présent dans la base alors Y est présent dans le même enregistrement", avec X et Y des itemsets. Une méthode pour générer de telles implications est de considérer toutes les permutations possibles entre les items du motif pour que les items soient attribués successivement à l'itemset X ou Y.

Définition 2.1.4 (Règle d'association) Une *règle d'association* est une implication de la forme $X \rightarrow Y$, avec $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, et $X \cap Y = \emptyset$.
X est appelé l'antécédent de la règle et Y son conséquent.

Les motifs ou les règles d'association qu'il est possible d'identifier pouvant être très nombreux, il est nécessaire de pouvoir les comparer et éventuellement selon certains critères les ordonner. Le premier critère utilisé est la fréquence d'apparition du motif ou de la règle dans la base de données.

Supposons que soit stockée dans une base de données l'architecture en domaine de protéines (par exemple dans Pfam), supposons que dans 70 % des protéines d'une famille protéique donnée (l'architecture d'une protéine représente alors un enregistrement), les domaines PLDc et low_complexity soient observés simultanément (chacun des domaines représente un item), cette co-occurrence ne nous donne aucune information fonctionnelle sur les protéines, mais nous indique un intérêt potentiel de l'étude du motif {PLDc, low_complexity}.

Notation Notons n_X (resp. n_Y) le nombre d'enregistrements de $\mathcal{D}$ contenant l'itemset X (resp. Y), et n_{XY} le nombre d'enregistrements contenant $X \cup Y$. Le nombre d'enregistrements de la base de données est noté n.

Nous avons vu qu'un indicateur de l'intérêt d'un motif, ou d'une règle, pouvait être sa fréquence dans la base étudiée. Nous introduisons les notions de support d'un motif ou d'une règle, et de motif fréquent.

Définition 2.1.5 (Support) Soit X un motif et $\mathcal{D}$ une base de données contenant n enregistrements :
$\text{support}_{\mathcal{D}}(X) = \frac{n_X}{n}$
Soit $X \rightarrow Y$ une règle d'association :
$\text{support}_{\mathcal{D}}(X \rightarrow Y) = \frac{n_{XY}}{n}$

Un **motif fréquent** est un motif dont la valeur de support est supérieure ou égale à un seuil *min_supp*.

Pour les règles d'association, un critère permettant d'ordonner les règles les unes par rapport aux autres et d'extraire les plus pertinentes d'entre elles, est la confiance.
D'autres critères peuvent être définis, mais la confiance est le critère le plus utilisé.

Définition 2.1.6 (Confiance) $\text{confiance}_{\mathcal{D}}(X \rightarrow Y) = \frac{n_{XY}}{n_X}$

Exemple :
Soit la règle r_1 extraite de $\mathcal{D}$: $ABC \rightarrow E$
Soit la règle r_2 extraite de $\mathcal{D}$: $BC \rightarrow E$
Nous pouvons calculer les valeurs suivantes :
$\text{support}_{\mathcal{D}}(r_1) = 2/4 = 0.5$
$\text{confiance}_{\mathcal{D}}(r_1) = 2/2 = 1$
$\text{support}_{\mathcal{D}}(r_2) = 2/4 = 0.5$
$\text{confiance}_{\mathcal{D}}(r_2) = 2/3 = 0.66$
Alors que r_1 et r_2 ont même support, on voit que r_1 est plus pertinente que r_2 car de confiance égale à 1.

Dans le cadre de l'étude fonctionnelle de familles protéiques, nous pourrions par exemple nous intéresser à :
- Toutes les règles d'association concluant sur une classe fonctionnelle donnée. Cela permettrait d'aller vers l'automatisation de l'annotation fonctionnelle.
- Toutes les règles d'association ayant un domaine protéique donné dans leur antécédent, et ainsi étudier le rôle de ce domaine pour différentes fonctions biologiques.

Pour passer des motifs fréquents aux règles d'association, [7] propose la méthode suivante :

1. Trouver tous les ensembles d'items (itemsets) dont le support est supérieur à un seuil minimum.
2. Utiliser les motifs fréquents l pour générer les règles d'association : pour chaque itemset l, trouver tous les sous-ensembles stricts non-vides a de l. Pour chaque sous-ensemble a, retourner une règle de la forme $a \rightarrow (l - a)$.

2.2 Motifs séquentiels fréquents

Les motifs séquentiels fréquents sont une extension des motifs fréquents.

Les notions suivantes sont empruntées à [8].

Définition 2.2.1 (Séquence, motif séquentiel) Une *séquence* s dans I est une liste ordonnée d'itemsets, notée $\langle E_1, E_2, ..., E_l \rangle$, avec $E_i \subseteq \mathcal{I}, 1 \leq i \leq l$. Il faut noter qu'un itemset peut avoir de multiples occurrences dans une séquence.
Un motif séquentiel est une *séquence*.

Définition 2.2.2 (Base de données séquentielles) Une *base de données séquentielles* notée $\mathcal{DS}$ est un ensemble de séquences.

Définition 2.2.3 (Règle d'association séquentielle) Une *règle d'association séquentielle* est une implication de la forme $r : W \rightarrow Y$ où W est une séquence contenant au moins un itemset et Y un itemset. W est appelé l'*antécédent* de la règle (noté ant(r)) et Y son *conséquent* (noté cons(r)).

Définition 2.2.4 (Taille) La *taille* d'une séquence s est le nombre d'itemsets qui la composent et est notée $|s|$. On parle alors de k-séquence. Nous dirons que la taille d'une règle r, notée $|r|$, est la taille de la séquence composant son antécédent.

Définition 2.2.5 (Sous-séquence) Une séquence $s = \langle E_1, E_2, ..., E_n \rangle$ est appelée *sous-séquence* d'une autre séquence $s' = \langle F_1, F_2, ..., F_m \rangle$, noté $s \sqsubseteq s'$, si et seulement si il existe des entiers $j_1, ..., j_n$, tels que $1 \leq j_1 < j_2 < ... < j_n \leq m$ et $E_1 \subseteq F_{j_1}$, $E_2 \subseteq F_{j_2}$, ... , $E_n \subseteq F_{j_n}$, où $\subseteq$ représente l'inclusion classique entre deux ensembles. Nous disons que s' *contient* s.
De plus si s et s' sont deux séquences distinctes telles que $s \sqsubseteq s'$, nous notons $s \sqsubset s'$.

Exemples :
Soit $\alpha' = \langle a, b, f, c, e, f, g \rangle$ une séquence.
$\alpha = \langle b, c, f, g \rangle$ est une sous-séquence de α'.
Soit $\beta' = \langle (abc), (ade), (bde), (bdf), (abc) \rangle$ une séquence.
$\beta = \langle (ab), (a), (bde), (bc) \rangle$ est une sous-séquence de β'.

Définition 2.2.6 (Support) Le *support* d'un motif séquentiel m dans une base de données séquentielles $\mathcal{DS}$ est défini comme le nombre de séquences de $\mathcal{DS}$ qui contiennent le motif. Formellement, nous avons :

$$\text{support}_{\mathcal{DS}}(m) = |\{\langle s \rangle \in \mathcal{DS} \; t.q. \; (m \sqsubseteq s)\}|$$

Il est important de noter que les itemsets dans m n'ont pas besoin d'être consécutifs dans une séquence de $\mathcal{DS}$ pour être contenus par cette séquence.

La section suivante présente les algorithmes classiques qui permettent de rechercher de tels itemsets ou règles d'association.

2.3 Algorithmes de recherche de motifs séquentiels fréquents

Dans cette section, nous présentons les principaux algorithmes de recherche de motifs fréquents et de motifs séquentiels fréquents. Dans un premier temps, nous nous intéresserons à deux algorithmes de recherche de motifs fréquents et règles d'association : Apriori et FP-growth, puis nous étudierons trois algorithmes de recherche de motifs séquentiels fréquents.

2.3.1 Recherche de motifs fréquents

Le problème de la recherche de motifs fréquents et de règles d'association dans des grandes bases de données, a été introduit par [7]. De nombreux algorithmes ont été proposés par la suite : [104, 90, 114, 119, 20, 47]. Parmi ces algorithmes, seuls trois ont été implémentés indépendamment des auteurs et mis à disposition publiquement (par exemple dans [18] ou [19]) : Apriori, Eclat et FP-Growth. Dans cette section nous présenterons le principe de ces 3 algorithmes.

Apriori

Apriori a été introduit en 1993 par Agrawal et co-auteurs [7] et est le premier algorithme de recherche de règles d'association proposé pour des grandes bases de données. Il cherche des règles ou des itemsets respectant deux contraintes : une condition de fréquence minimale (ce sont des itemsets fréquents), et une contrainte de confiance minimale. Apriori utilise une stratégie "bottom up", c'est-à-dire qu'il construit successivement les itemsets fréquents de taille k (en commençant par les itemsets de taille 1), puis étend les noyaux ainsi établis à l'aide des itemsets fréquents restants. Il s'agit de l'étape de génération des candidats. Il construit alors les candidats itemsets fréquents de taille $k+1$. A cette étape les itemsets non fréquents de taille $k+1$ sont élagués.

L'étape de génération des candidats produit potentiellement une grande quantité de candidats inutiles. Nous présentons maintenant deux algorithmes qui s'affranchissent en partie de cette étape.

Eclat

L'algorithme Eclat [119] partitionne l'espace de recherche pour pouvoir paralléliser certaines tâches. Les données sont dans un premier temps stockées dans une base de données. L'algorithme évalue d'abord la fréquence des itemsets de taille 2, puis partitionne la base. Il transforme ensuite la base par une verticalisation (on ordonne non plus les items par identifiant d'enregistrement, mais les enregistrements par items), puis divise l'espace en utilisant les items. Les motifs fréquents sont ensuite identifiés indépendamment sur chacune des partitions ainsi formées. L'algorithme est fait pour être utilisé en programmation parallèle, puisqu'il affecte une "sous-base" à un processeur de façon à donner les tâches les plus longues aux processeurs disponibles. D'autre part, il charge en mémoire les données afin de les traiter au plus vite.

FP-Growth

L'algorithme FP-Growth [47] utilise un arbre préfixe pour représenter de façon condensée les données. L'algorithme adopte une stratégie de type diviser pour régner. Il utilise le principe de l'accroissement du préfixe pour trouver les motifs fréquents. Nous ne détaillerons pas plus les principes de FP-Growth dans cette partie, car nous développerons des notions très proches appliquées aux motifs séquentiels dans la section suivante, avec PrefixSpan.

D'après les mesures réalisées par Christian Borgelt [19] avec ces implémentations de Apriori, Eclat et FP-growth, ce dernier est clairement le plus rapide sur l'ensemble des benchmarks de test et ce quels que soient les paramètres utilisés.

2.3.2 Recherche de motifs séquentiels fréquents

Une des extensions de la recherche de motifs fréquents présentée dans la section précédente est la recherche de motifs fréquents séquentiels. Ce sujet a fait l'objet de nombreux travaux [8, 107, 46, 91, 118].

Nous présentons plus en détail dans cette section trois algorithmes de recherche de motifs séquentiels, le premier pour des raisons historiques : GSP [8] qui est le premier algorithme à avoir été conçu pour rechercher ce genre de motifs. Nous présenterons aussi parmi les plus récents, deux autres algorithmes, SPADE [118] et PrefixSpan [91], qui utilisent des structures de données différentes.

Ces trois méthodes permettent la découverte de motifs séquentiels fréquents, mais pas de règles d'association séquentielles. En effet la difficulté due à l'explosion combinatoire du nombre de règles qu'il est possible de générer à partir d'un motif séquentiel empêche un passage simple de l'un à l'autre comme dans le cas des motifs fréquents et des règles d'association.

GSP

Le premier algorithme de recherche de motifs séquentiels est une adaptation de Apriori, nommée GSP. L'algorithme utilise une notion nouvelle, la **jointure séquentielle** de deux séquences.

Jointure séquentielle Nous définissons la jointure séquentielle d'après Agrawal *et al.* [8] repris dans Masseglia *et al.*[72] pour deux séquences X_i et X_j :
$jointure(X_i, X_j)$: on ordonne les items au sein des itemsets par ordre alphanumérique. La jointure séquentielle est définie si la sous-séquence obtenue en supprimant le premier élément (plus petit item du premier itemset) de X_i est la même que la sous-séquence obtenue en supprimant le dernier élément (le plus grand item du dernier itemset) de X_j. La séquence candidate obtenue par $jointure(X_i, X_j)$ est la séquence X_i étendue avec le dernier item de X_j.
L'item ajouté fait partie du dernier itemset de X_i s'il était dans un itemset de taille supérieure à 1 dans X_j et devient un nouvel itemset s'il était dans un itemset de taille 1 dans X_j.

Exemple 1 : tout d'abord un exemple trivial.

$$X_i = < (A)(B)(C) >$$
$$X_j = < (B)(C)(D) >$$
$$jointure(X_i, X_j) = < (A)(B)(C)(D) >$$

Exemple 2 : cas où le dernier item de la deuxième séquence appartient à un itemset de taille supérieure à 1.

$$X_i = < (AB)(C) >$$
$$X_j = < (B)(CD) >$$
$$jointure(X_i, X_j) = < (AB)(CD) >$$

Exemple 3 : cas où le dernier item de la deuxième séquence appartient à un itemset de taille 1.

$$X_i = < (\text{AB})(\text{C}) >$$
$$X_j = < (\text{B})(\text{C})(\text{E}) >$$
$$jointure(X_i, X_j) = < (\text{AB})(\text{C})(\text{E}) >$$

Contre-exemple 1 : jointure trivialement non définie.

$$X_i = < (\text{B})(\text{C})(\text{E}) >$$
$$X_j = < (\text{AB})(\text{C}) >$$
$$jointure(X_i, X_j) \text{ non définie}$$

Contre-exemple 2 : les sous-séquences de taille n-1 ne se contiennent pas mutuellement.

$$X_i = < (\text{A})(\text{BC}) >$$
$$X_j = < (\text{B})(\text{C})(\text{D}) >$$
$$jointure(X_i, X_j) \text{ non définie}$$

Propriété : la jointure n'est pas commutative.

Exemple :

$$X_i = < (\text{A})(\text{B})(\text{C}) >$$
$$X_j = < (\text{B})(\text{C})(\text{D}) >$$
$$jointure(X_i, X_j) = < (\text{A})(\text{B})(\text{C})(\text{D}) >$$

Contre-exemple :

$$X_j = < (\text{B})(\text{C})(\text{D}) >$$
$$X_i = < (\text{A})(\text{B})(\text{C}) >$$
$$jointure(X_j, X_i) \text{ non définie}$$

On peut envisager d'autres jointures, moins strictes par exemple, où il suffirait d'avoir une sous-séquence commune plus petite que précédemment pour pouvoir réaliser la jointure.
L'avantage de la jointure précédente est qu'elle génère des motifs de taille n ou $n+1$ à partir de deux séquences de taille n.

Algorithme L'algorithme GSP peut être décomposé en deux étapes.

1. **Génération des candidats :** étant donné un ensemble $\mathcal{E}$ de séquences fréquentes (c'est-à-dire dont le nombre d'occurrences dans une base de données est au moins égal à un seuil déterminé par l'utilisateur) de taille $n-1$, on génère les candidats de taille n par jointure de $\mathcal{E}$ sur lui-même. Pour augmenter la vitesse d'exécution de la phase suivante, les séquences de taille n générées sont stockées dans une table de hachage.
2. **Évaluation du support :** pour chaque séquence candidate, on calcule le support (en utilisant la table de hachage précédemment créée). Les séquences non fréquentes sont éliminées.

Limites Les algorithmes dérivés de Apriori [8] tels que GSP [108] présentent des faiblesses :
– le nombre de candidats générés peut être très grand ($n^2 + \frac{n(n+1)}{2}$ à chaque génération, n étant le nombre de séquences de la base à la génération précédente).
– La recherche de l'ensemble des séquences fréquentes peut nécessiter un grand nombre d'accès à la base de données.
– Du fait des deux points précédents, il est très difficile, voire impossible, de trouver de longs motifs séquentiels.
Pour résoudre ces problèmes et pour permettre une recherche plus rapide et plus efficace, de nouveaux algorithmes ont été développés.

Algorithme 1: GSP

Données : $\mathcal{DS}$ Une base de données séquentielles.
$\mathcal{F}$ un ensemble de séquences fréquentes.
min_supp un seuil de support

retourner *L'ensemble des séquences fréquentes*
début
  $\mathcal{F}_1$ = les 1-séquences fréquentes
  $k = 2$
  tant que $\mathcal{F}_{k-1} \neq \varnothing$ **faire**
    C_k = L'ensemble des k-séquences candidates obtenues par jointure des séquences de taille $k-1$.
    pour *toutes les séquences* α *de la base de données* $\mathcal{DS}$ **faire**
      Évaluer le support de α dans $\mathcal{DS}$
      $\mathcal{F}_k = \{\alpha \in C_k / \text{support}_{\mathcal{DS}}(\alpha) \geq min\_sup\}$
    finPour
    $k = k + 1$
  finTq
fin
retourner *L'ensemble des séquences* $\cup_k \mathcal{F}_k$

PrefixSpan, Pei et al. 2001

Pei *et al.* [91] proposent PrefixSpan un algorithme permettant la recherche de motifs séquentiels dans une base de données par Préfixe-Projection. La préfixe-projection est une opération qui étant donné une séquence et un préfixe calcule le plus grand suffixe possible. Par exemple, soit une séquence *abcd*. Considérons le préfixe *b*, la préfixe-projection de *b* est *cd*. Dans le cas d'une séquence d'itemsets : $a(abc)(ac)d(cf)$, considérant le préfixe $a(ab)$, la préfixe-projection est $(\_c)(ac)d(cf)$.

Préfixe-Projection

Algorithme PrefixSpan

- **Étape 1 : Trouver les items fréquents.** Une passe sur la base de données séquentielles permet d'évaluer le support de chaque motif de taille 1 de la base (dans la suite de cette section, nous parlerons d'évaluer le support d'un item).
- **Étape 2 : Diviser l'espace de recherche.** On divise l'espace de recherche en fonction du nombre de préfixes différents trouvés dans la base.
- **Étape 3 : Trouver les sous-ensembles de motifs séquentiels.** Les sous-ensembles de motifs séquentiels peuvent être trouvés en construisant les préfixes-projections des bases obtenues et en réappliquant l'algorithme de manière récursive.

La preuve de la complétude de l'algorithme est donnée dans Pei *et al.* [91].

Nous ne donnerons pas plus de précisions sur l'algorithme car ces notions seront reprises et étendues dans le chapitre suivant. Nous illustrons le comportement de l'algorithme PrefixSpan sur un exemple.

Soit une base de données séquentielles $\mathcal{DS}$. Posons min_supp = 2 ;

Sequence_ID	Sequence
10	$< (a)(abc)(ac)(d)(cf) >$
20	$< (ad)(c)(bc)(ae) >$
30	$< (ef)(ab)(df)(c)(b) >$
40	$< (e)(g)(af)(c)(b)(c) >$

Étape 1 : tout d'abord, cherchons les items de la base et calculons leur support. On obtient :

item	a	b	c	d	e	f	g
support	4	4	4	3	3	3	1

On peut créer la liste des items fréquents classés par ordre décroissant de support :

$$a : 4,\ b : 4,\ c : 4,\ d : 3,\ e : 3,\ f : 3$$

On cherche maintenant à créer les {préfixe}-projections pour l'ensemble des items fréquents.

préfixe	{préfixe}-projection
$< a >$	$< (abc)(ac)(d)(cf) >$ $< (\_d)(c)(bc)(ae) >$ $< (\_b)(df)(c)(b) >$ $< (\_f)(c)(b)(c) >$
$< b >$	$< (\_c)(ac)(d)(cf) >$ $< (\_c)(ae) >$ $< (df)(c)(b) >$ $< (c) >$
...	...

Étape 2, on identifie les motifs séquentiels fréquents.
Puis on relance récursivement sur la nouvelle base créée par {a}-projection.

$$\text{f\_list} = ab : 4,\ ac : 4,\ aa : 2,\ af : 2.$$

Appel récursif à PrefixSpan :
Calcul des {préfixe de taille 2}-projections :

préfixe	{préfixe}-projection
$< aa >$	$< (\_bc)(ac)(d)(cf) >$
$< ab >$	$< (\_c)(ac)(d)(cf) >$ $< (\_c)(a) >$ $< (c) >$
...	...

Étape 3, récursivement, on trouve l'ensemble des motifs séquentiels fréquents de la base de données initiale.

SPADE, Zaki 2001

SPADE est présenté dans [118]. Cet algorithme cherche à réduire l'espace de recherche en regroupant les séquences par catégories. L'algorithme regroupe les motifs séquentiels fréquents qui présentent des préfixes communs, ce qui permet de décomposer le problème en sous-problèmes traités en mémoire.

Le calcul des candidats de la génération $n+1$ passe par une inversion de la base de données, qui la transforme d'un format vertical à un format horizontal.

L'algorithme gère les catégories de la façon suivante : deux k-séquences appartiennent à la même catégorie si elles possèdent un préfixe commun de taille $k-1$. Plus formellement, soit $\mathcal{P}_{k-1}(\alpha)$ la séquence de taille $k-1$ préfixe de la séquence α. Comme α est fréquente, $\mathcal{P}_{k-1}(\alpha) \in F_{k-1}$, avec F_{k-1} l'ensemble des séquences fréquentes de taille $k-1$. Une catégorie ou classe d'équivalence est définie de la manière suivante :

$$[\rho \in F_{k-1}] = \{\alpha \in F_k / \mathcal{P}_{k-1}(\alpha) = \rho\}$$

Les candidats de la génération $n+1$ sont générés par auto-jointure de chaque classe d'équivalence.

Performances des algorithmes

[72] explique qu'il n'est pas possible de déterminer lequel de ces algorithmes est le meilleur. Les performances dépendent étroitement du type des données étudiées. L'algorithme GSP est très efficace pour les grandes bases de données avec des séquences moyennement longues, alors que les algorithmes SPADE ou PrefixSpan sont performants dans le cas où un très grand nombre de candidats de même taille sont générés.

2.4 Mesures de qualité

L'identification des séquences d'itemsets qui sont fortement corrélées et la construction de règles d'association pertinentes avec ces itemsets sont des tâches difficiles. Les mesures d'intérêt aident à estimer l'importance d'une règle : elles peuvent être utilisées pour élaguer les règles de faible utilité, ou pour ranger et sélectionner les règles d'intérêt. La sélection d'une bonne mesure permet de réduire les coûts de recherche aussi bien en temps qu'en espace durant le processus de découverte de ces règles [110, 41].

Dans cette section, on considérera les mesures d'intérêt d'une règle d'association, séquentielle ou non, identifiée dans une base de données (séquentielle si la règle est séquentielle) $\mathcal{D}$.

$$r : A \rightarrow B$$

Toutes les mesures d'intérêt ne capturent pas les mêmes genres d'association. Par exemple, en utilisant une approche basée sur la confiance et le support, une règle $A \rightarrow B$ peut être considérée comme importante, même si B est souvent trouvé sans A. La distribution des exemples de $\overline{A}$ entre B et $\overline{B}$ n'est pas prise en compte.

Une mesure de qualité est une fonction mathématique qui associe à une règle d'association dans une base de données une valeur réelle. Pour cela elle utilise des valeurs de $p(A)$, $p(B)$, $p(AB)$,

	A et B incompatibles	A et B indépendants	A implique B
$f(A,B) =$	-1 ↗	0 ↗	1

TAB. 2.1 – *Tableau de variation d'une bonne mesure de qualité selon Zhang.*

$p(\overline{A})$, $p(\overline{B})$, ... dans la base de données. On peut avoir des mesures de qualité de différentes sortes.
Notons f une mesure de qualité, r une règle d'association et $\mathcal{D}$ une base de données.
Généralement,

$$f : r, \mathcal{D} \rightarrow \mathbb{R}$$

Selon Zhang [120], on se limite à :

$$f : r, \mathcal{D} \rightarrow [-1; 1]$$

Les tableaux ci-dessous présentent différentes mesures d'intérêt identifiées dans [65] et [62] et comparent leurs valeurs dans 3 situations remarquables identifiées dans [62] :
- L'incompatibilité : A et B ne sont jamais trouvés simultanément.
- L'indépendance : $\frac{p(A)p(B)}{p(AB)} = 1$
- Le cas de règles logiques : chaque fois que A est trouvé, B est présent.

Piatetsky [92] propose des propriétés (normalisées par Zhang [120]) qu'une "bonne mesure" doit satisfaire. La valeur de la mesure doit être égale à :
- -1, en cas d'incompatibilité.
- 0, en cas d'indépendance.
- 1, au cas où la règle r est logique (toujours vérifiée).

Ces propriétés sont exprimées sous la forme du tableau de variation 2.1.

Le tableau 2.2 présente les mesures de qualité "basiques", le minimum en cas d'incompatibilité est souvent égal à 0 et le maximum peut être au plus 1. Le tableau 2.3 présente les mesures dont le domaine de variation est plus grand que les mesures précédentes : la moindre-contradiction, la confiance-relative pondérée et la mesure Z varient entre une borne positive et une borne négative. Le multiplicateur de cote donne un poids très important à l'implication logique mais différencie peu les autres cas.

[61] et [41] suggèrent un certain nombre de propriétés clés à examiner pour guider la sélection de la mesure la plus adaptée pour étudier au mieux les données. La compréhension par l'utilisateur de la signification de la mesure d'intérêt est un autre critère important.

2.5 Pépites de connaissance

Une des difficultés dans l'utilisation des règles d'association réside dans leur validation et leur expertise. Selon les jeux de données étudiés le nombre de règles trouvées par les algorithmes peut

Nom	formule	incomp.	indép.	règle logique
Confiance	$p(B/A)$	0	$p(B)$	1
Précision Négative	$p(\overline{B}/\overline{A})$	$1/p(\overline{A})$	$p(\overline{B})$	$p(\overline{B})/p(\overline{A})$
Sensibilité	$p(A/B)$	0	$p(A)$	$p(A)/p(B)$
Spécificité	$p(\overline{A}/\overline{B})$	0	$p(\overline{A})$	1
Couverture	$p(A)$	$p(A)$	$p(A)$	$p(A)$
Support	$p(AB)$	0	$p(A)p(B)$	$p(A)$

TAB. 2.2 – *Les mesures d'intérêt classiques pour une règle* $A \rightarrow B$

Nom	formule	incomp.	indép.	règles logiques
Moindre-contradiction	$\frac{p(AB)-p(A\overline{B})}{p(B)}$	$-\frac{p(A)}{p(B)}$	$2p(A) - \frac{p(A)}{p(B)}$	$\frac{p(A)}{p(B)}$
Confiance-relative pondérée	$p(A)(p(A/B) - p(B))$	$-p(B)p(A)$	0	$p(A)(1 - p(B))$
Multiplicateur de cote	$\frac{p(AB)p(\overline{B})}{p(A\overline{B})p(B)}$	0	1	$+\infty$
Zhang	$\frac{p(AB)-p(A)p(B)}{\max(p(AB)p(\overline{B});p(B)p(A\overline{B}))}$	-1	0	1
ZC	$\frac{p(AB)-p(A)p(B)}{p(A\overline{B})p(B)}$	-1	0	$+\infty$

TAB. 2.3 – *Mesures d'intérêt pour une règle* $A \rightarrow B$

être élevé, et ce d'autant plus que les conditions de support sont abaissées. Sans même chercher des règles rares, le fait de passer d'un seuil de 80 % par exemple à 50 % pour le support pourra augmenter considérablement le nombre de règles obtenues. Il est difficile de proposer à l'expert les règles résultantes telles quelles.

Pour pallier cette possible explosion du nombre de règles transmises à l'expert, une solution est l'utilisation de mesures de qualité qui permettra de classer et/ou filtrer les règles pour n'en garder que les plus intéressantes. La plus répandue et utilisée classiquement est la confiance introduite dans la section précédente. Ce problème est d'autant plus important dans le domaine d'application qui nous concerne, la bioinformatique, que des événements rares peuvent être très intéressants. Nous allons donc souvent être amenés à utiliser des seuils de support faible.

Les pépites de connaissance [12] ont été introduites comme des règles d'association de bonne qualité et de support potentiellement faible. On sélectionnera des règles en fonction de ces critères pour ne proposer à l'expert que les plus pertinentes. La qualité de la règle est évaluée grâce à l'utilisation de mesures de qualité.

La condition de support minimal sert alors à éviter l'apprentissage par cœur, où les règles obtenues ne feraient que décrire les données enregistrement par enregistrement sans en extraire un comportement ou des implications globales. Il est aussi possible de limiter le nombre de règles obtenues en ne conservant que les meilleures parmi celles dépassant certains seuils [12] ou en choisissant de ne proposer à l'expert que les règles "les plus petites" comme nous le ferons dans la section suivante.

Chapitre 3

SNK : un algorithme de recherche de pépites séquentielles de connaissance

Sommaire

3.1 Pépites séquentielles de connaissance : c-SNoKs et s-SNoKs

Dans le chapitre précédent, les notions de règles d'association, d'itemsets séquentiels fréquents et de pépites de connaissance ont été présentées.

Dans le cadre d'une collaboration avec Frédérique Lisacek de l'Institut Suisse de Bioinformatique (SIB), nous nous sommes intéressés à l'étude de familles protéiques en relation avec leur architecture en domaines protéiques. Dans le chapitre 1 nous avons vu qu'il était possible de représenter les protéines comme des séquences de domaines. Nous souhaitons maintenant pouvoir identifier des régularités entre des motifs séquentiels et des propriétés de ces protéines. Frédérique Lisacek avait émis l'hypothèse que l'ordre d'apparition de ces domaines pouvait jouer un rôle fonctionnellement important. Nous avons donc développé un algorithme capable de tester cette hypothèse.

Le domaine d'application étant la biologie, il est très possible de rencontrer des motifs pertinents, mais peu représentés dans les données. La notion d'itemsets séquentiels fréquents ne pouvait donc pas être utilisée telle quelle. C'est pourquoi nous avons introduit la notion de Pépite

Séquentielle de Connaissance [38] (Sequential Nugget of Knowledge : SNoKs), comme une règle d'association séquentielle avec des valeurs de support et de mesure de qualité supérieures à deux seuils définis par un utilisateur pour une base de données et une mesure de qualité donnée. Nous nous intéresserons aux pépites séquentielles de connaissance vérifiant une propriété de minimalité, c'est-à-dire que nous ne souhaitons pas obtenir toutes les règles, mais seulement les plus générales.

Nous introduisons ensuite deux notions dérivées des Pépites Séquentielles de Connaissance : les pépites séquentielles de connaissance contraintes de taille k (Constrained SNoK : k-c-SNoK) et les pépites séquentielles de connaissance relâchées de taille q (Soft SNoK : q-s-SNoK).

k-c-SNoKs Une k-c-SNoK est une pépite séquentielle de connaissance r de taille supérieure ou égale à un *seuil de taille* k défini par l'utilisateur et minimale.

Dans une k-c-SNoK, le membre gauche contient au moins k itemsets. La sélection de r comme une règle valide suit le principe suivant : il n'existe pas de règle r' avec le même conséquent qui satisfait les conditions de support, mesure de qualité et taille, et dont l'ensemble des items dans le membre gauche est strictement inclus dans l'ensemble des items de r.

Ce seuil de taille peut être très utile quand les règles les plus petites ne sont pas significatives. Si k est égal à 1, une 1-c-SNoK correspond à un SNoK basique comme introduit plus haut.

Exemple :
Considérons deux règles respectant les seuils suivants : 0.05 pour le support, 0.9 pour la mesure de qualité, et 2 pour le seuil de taille.

```
R1 : PLDc, Pfam-B_115, Pfam-B_6054 -> Function1, (0.29,0.93)
R2 : PLDc, Pfam-B_6054 -> Function1, (0.31,1.00)
```

La règle R1 se traduit par : les domaines PLDc suivis de Pfam-B_115 puis de Pfam-B_6054 dans cet ordre strictement sont trouvés en association avec la cible Function1 dans 29 % des n-uplets de la base. La force de l'association est indiquée par la mesure de qualité et vaut ici 0.93 (valeur maximum à 1). R_1 et R_2 satisfont toutes les deux les conditions de support et de mesure de qualité et sont de taille supérieure ou égale à 2.
R_1 n'est pas un 2-c-SNoK parce que (i) le membre gauche de la règle R_2 est strictement inclus dans le membre gauche de la règle R_1 et (ii) tous les domaines présents dans le membre gauche de la règle R_2 apparaissent dans le même ordre dans le membre gauche de R_1. R_2 est de taille 2, il ne peut pas exister d'autres règles plus petites contenant R_2 et respectant les conditions de seuils, R_2 est un 2-c-SNoK.

q-s-SNoKs Les pépites séquentielles de connaissance relâchées (q-s-SNoK) sont des pépites séquentielles de connaissance de taille inférieure ou égale à un seuil q de taille maximum défini par l'utilisateur, et qui satisfont les conditions de support et de mesure de qualité.

Exemple :
Considérons deux règles respectant les seuils suivants : 0.05 pour le support, 0.9 pour la mesure de qualité, et 2 pour le seuil de taille.
`R1 : PLDc, Pfam-B_115, Pfam-B_6054 -> Function1, (0.29,0.93)`
`R2 : PLDc, Pfam-B_6054 -> Function1, (0.31,1.00)`

R_2 est une 2-s-SNoK parce qu'elle vérifie les conditions de seuil. En particulier, sa taille est inférieure ou égale à 2. En revanche R_1 est de taille 3 et n'est donc pas une 2-s-SNoK.

3.2 Définitions mathématiques et propriétés

Nous souhaitons découvrir des dépendances entre la description d'objets en terme de séquence d'items, et un item cible spécifique.

Notation Notons IDT l'ensemble des identifiants des objets et T l'ensemble des items cibles. Soit I l'ensemble de tous les items (attributs booléens). Les ensembles I et T sont supposés disjoints. Rappelons qu'un *itemset* est un sous-ensemble de I.

Définition 3.2.1 (Concaténation) Définissons $s = \langle E_1, E_2, ..., E_n \rangle$ et $s' = \langle F_1, F_2, ..., F_m \rangle$ deux séquences dans I. Nous notons $s \cdot s'$ la séquence résultant de la concaténation de s avec s' : $s \cdot s' = \langle E_1, E_2, ..., E_n, F_1, F_2, ..., F_m \rangle$.

Exemple :
Soit $\alpha = \langle (abe), (bd), (cf) \rangle$ et $\beta = \langle (abc), (e) \rangle$. La concaténation de α avec β est : $\alpha \cdot \beta = \langle (abe), (bd), (cf), (abc), (e) \rangle$

Définition 3.2.2 (Base de données séquentielles catégorisées) .
Une *base de données séquentielles catégorisées* est un ensemble CSD de n-uplets $\langle sid, s, tg \rangle$, $sid \in IDT$, $tg \in T$, où sid est l'identifiant de l'objet, s la séquence d'itemsets de I le décrivant et tg l'item cible qui lui est associé. Un n-uplet $\langle sid, s, tg \rangle$ *contient* une séquence s' si et seulement si s' est une sous-séquence de s.

Toutes les notions de cette section seront illustrées sur l'exemple suivant :

Exemple :

CSD :

	id	seq	cible
α_1	$\langle id_1,$	$\langle a, b, f, c, e, f, g \rangle$	$, tg_1 \rangle$
α_2	$\langle id_2,$	$\langle a, e, b, h, c, f, g \rangle$	$, tg_1 \rangle$
α_3	$\langle id_3,$	$\langle c, e, a, b, e, g, f \rangle$	$, tg_2 \rangle$
α_4	$\langle id_4,$	$\langle c, e, a, b, e, g, f, a, e, b, f, d \rangle$	$, tg_2 \rangle$

CSD est une base de données séquentielles catégorisées contenant 4 n-uplets.

Dans CSD la séquence $\langle b, e, f \rangle$ est une sous-séquence de $\langle a, b, f, c, e, f, g \rangle$ et α_1 contient la séquence $\langle b, e, f \rangle$.

Une règle d'association séquentielle est une combinaison des règles d'association classiques et des patterns séquentiels. Formellement on a :

Définition 3.2.3 (Règle d'association séquentielle) Une *règle d'association séquentielle* r sur CSD est une implication de la forme $ANT \rightarrow CONS$, où ANT est une séquence d'itemsets de $\mathcal{I}$ et $CONS$ est un élément de T. Nous appelons ANT (resp. $CONS$) l'*antécédent* (resp. *conséquent*) de r et nous le notons ant(r) (resp. cons(r)).

Définition 3.2.4 (Support) Le *support* d'une règle d'association séquentielle r sur une base de données séquentielles catégorisées CSD est défini comme le nombre de n-uplets de CSD qui contiennent à la fois son antécédent et son conséquent. Formellement, nous avons :

$$\text{support}_{CSD}(ANT \rightarrow CONS) = |\{\langle sid, s, tg\rangle \in CSD \ t.q. \ (ANT \sqsubseteq s) \wedge (CONS = tg)\}|$$

Il est important de noter que les itemsets dans ANT n'ont pas besoin d'être consécutifs pour être contenus par un n-uplet de CSD.

Exemple :
$\text{support}_{CSD}(\langle a, b, f\rangle \rightarrow tg_1) = 2$

Définition 3.2.5 (Confiance) La *confiance* d'une règle d'association séquentielle r dans une base de données séquentielles catégorisées CSD mesure parmi les n-uplets de CSD qui contiennent son antécédent combien contiennent son conséquent.

$$\text{confiance}_{CSD}(ANT \rightarrow CONS) = \frac{|\{\langle sid, s, tg\rangle \in CSD \ t.q. \ (ANT \sqsubseteq s) \wedge (CONS = tg)\}|}{|\{\langle sid, s, tg\rangle \in CSD \ t.q. \ ANT \sqsubseteq s\}|}$$

Exemple :
$\text{confiance}_{CSD}(\langle a, b, f\rangle \rightarrow tg_1) = 0.5$
$\text{confiance}_{CSD}(\langle a, b, f, g\rangle \rightarrow tg_1) = 1$

Définition 3.2.6 (Généralisation) Une règle d'association séquentielle r_1 *contient* une autre règle r_2, notée $r_2 \preceq r_1$, si et seulement si $\text{cons}(r_1) = \text{cons}(r_2)$ et $\text{ant}(r_2) \sqsubseteq \text{ant}(r_1)$. Nous disons aussi que r_2 est *plus générale* que r_1. Si $r_1 \neq r_2$ et $r_2 \preceq r_1$, nous notons $r_2 \prec r_1$.

Exemple :
Soit $r_1 = \langle a, b, f, g\rangle \rightarrow tg_1$ et $r_2 = \langle a, b, f\rangle \rightarrow tg_1$; alors $r_2 \prec r_1$.
Soit $r_1 = \langle a, b, f, g\rangle \rightarrow tg_1$ et $r_2 = \langle a, b, f\rangle \rightarrow tg_2$ avec $tg_1 \neq tg_2$; alors r_2 et r_1 ne sont pas plus générales l'une que l'autre parce que $\text{cons}(r_1) \neq \text{cons}(r_2)$.

Nous introduisons maintenant formellement la notion principale de ce chapitre, c'est-à-dire les *Pépites Séquentielles de Connaissance*. Il s'agit de règles d'association séquentielles avec un support éventuellement bas, mais d'une haute qualité. Un support minimum n'est requis que pour éviter de découvrir des associations fortes n'impliquant que peu d'objets et qui pourraient provenir du bruit des données. Une *pépite séquentielle de connaissance* est définie comme une règle d'association séquentielle r dans CSD telle que ni son support, ni sa qualité ne sont inférieurs à leurs seuils respectifs.

Dans le cas des données que nous étudions, il est difficile de connaître exactement le sens d'un itemset. Un itemset AB peut indiquer une disjonction : le domaine A apparaît ou le domaine B. Mais il peut aussi indiquer la présence simultanée de deux domaines. Nous avons donc fait le choix de ne considérer que des séquences d'items (c'est-à-dire séquences d'itemsets constitués de singletons), qui suffisent à représenter les architectures en domaines. De plus, dans les applications que nous avons prévues, les objets sont décrits par des séquences d'items, de telle sorte que les séquences d'itemsets introduites dans le chapitre précédent sont inutilement compliquées. De ce fait, dans le reste de cette section, nous ne considérerons que des séquences où les itemsets ne sont constitués que d'un simple item. La définition de la sous-séquence peut être réécrite sous une forme plus simple en remplaçant les inclusions par des égalités.

Définition 3.2.7 (Sous-séquence) Une séquence $s = \langle E_1, E_2, ..., E_n \rangle$ est appelée *sous-séquence* d'une autre séquence $s' = \langle F_1, F_2, ..., F_m \rangle$, noté $s \sqsubseteq s'$, si et seulement si il existe des entiers $j_1, ..., j_n$, tels que $1 \leq j_1 < j_2 < ... < j_n \leq m$ et $E_1 = F_{j_1}$, $E_2 = F_{j_2}$, ... , $E_n = F_{j_n}$. Nous disons que s' *contient* s.

Dans les définitions suivantes, on considère une base de données séquentielles catégorisées CSD, un seuil de support min_supp, un seuil de mesure de qualité min_meas et une mesure de qualité IM.

Définition 3.2.8 (Pépite séquentielle de connaissance) On appelle pépite séquentielle de connaissance, une règle d'association séquentielle r telle que :
- $\text{support}_{CSD}(r) \geq min\_supp$
- $\text{measure}_{CSD,IM}(r) \geq min\_meas$

Définition 3.2.9 (Pépite séquentielle de connaissance minimale) .
On appelle pépite séquentielle de connaissance minimale, notée SNoK, une pépite séquentielle de connaissance r telle que :
- $\nexists r'$ une pépite séquentielle de connaissance avec $r' \prec r$.

Définition 3.2.10 (k-Pépite séquentielle de connaissance contrainte) .
On appelle pépite séquentielle de connaissance contrainte de longueur k notée k-c-SNoK, ou c-SNoK de longueur k, une pépite séquentielle de connaissance r telle que :
- $|r| \geq k$
- $\nexists r'$ une pépite séquentielle de connaissance, telle que
  * $|r'| \geq k$
  * $r' \prec r$

Les SNoKs sont des c-SNoKs de longueur 1.

Définition 3.2.11 (q-Pépite séquentielle de connaissance relâchée) .
On appelle pépite séquentielle de connaissance relâchée notée q-s-SNoK, ou s-SNoK de longueur q, une pépite séquentielle de connaissance r telle que :
- $|r| \leq q$

Remarque : une q-s-SNoK n'est pas forcément minimale.

Postfixe-projection : La méthode que nous proposons pour rechercher les règles d'associations séquentielles de connaissance s'inspire de l'approche de [91] pour les patterns séquentiels.

Nous projetons récursivement la base de données séquentielles catégorisées dans une base de données séquentielles catégorisées plus petite, et générons ainsi des bases projetées en agrandissant les préfixes.

Nous introduisons maintenant formellement les notions de préfixe et de postfixe-projection.

Pour ce qui suit, CSD est une base de données séquentielles catégorisées, et
$\alpha = \langle sid_1, \langle e_1 ... e_n \rangle, c_1 \rangle$ est un n-uplet de CSD.

Nous reprenons les notions définies par Pei [91].

Définition 3.2.12 (Préfixe) Soit $s' = \langle e'_1, ..., e'_m \rangle$ une séquence avec $m \leq n$. s' est appelé un *préfixe* de α si et seulement si $\forall i, 1 \leq i \leq m, e'_i = e_i$.

Exemple :
La séquence $\langle a, b, f \rangle$ est un préfixe de α_1 ($\langle id1, \langle a, b, f, c, e, f, g \rangle, tg_1 \rangle$).

Définition 3.2.13 (Suffixe) Étant donnée une séquence $\alpha = \langle e_1 e_2 ... e_n \rangle$.
Soit $\beta = \langle e_1 e_2 ... e_{m-1} e_m \rangle$ $(m \leq n)$ une séquence préfixe de α. La séquence $\gamma = \langle e_{m+1} ... e_n \rangle$ est un *suffixe* de α en considérant le préfixe β, noté $\gamma = \alpha / \beta$.

Nous avons la propriété : $\alpha = \beta \cdot \gamma$. Il faut noter que si β n'est pas une sous-séquence de α, le suffixe de α en considérant β est vide.

Notation Soit $\alpha = \langle sid, s, tg \rangle$ un n-uplet de CSD. Nous notons id, seq et target les méthodes qui renvoient respectivement l'identifiant, la séquence et la cible de α :
$\mathrm{id}(\alpha) = sid$, $\mathrm{seq}(\alpha) = s$ et $\mathrm{target}(\alpha) = tg$.

La notion de s'-projection correspond au n-uplet avec pour séquence la plus longue sous-séquence de s ayant s' comme préfixe.

Définition 3.2.14 Soit α un n-uplet et s' une séquence telle que $s' \sqsubseteq seq(\alpha)$. Un n-uplet $\alpha' = \langle id(\alpha'), seq(\alpha'), target(\alpha') \rangle$ est une *s'-projection* de α si et seulement si

1. $\mathrm{id}(\alpha') = \mathrm{id}(\alpha)$
2. $\mathrm{seq}(\alpha') \sqsubseteq \mathrm{seq}(\alpha)$
3. $\mathrm{target}(\alpha') = \mathrm{target}(\alpha)$
4. s' est un *préfixe* de α'
5. $\nexists \alpha''$ un n-uplet de CSD tel que $\mathrm{seq}(\alpha') \sqsubset \mathrm{seq}(\alpha'')$ et $\mathrm{seq}(\alpha'') \sqsubseteq \mathrm{seq}(\alpha)$ et s' est un préfixe de α''.

Remarque : Avec une telle définition, seule la sous-séquence de $seq(\alpha)$ préfixée par la première occurrence de s' sera considérée pour α'.

Exemple :
Rappelons $\alpha_1 = \langle id1, \langle a, b, f, c, e, f, g\rangle, tg_1\rangle$
$\langle id_1, \langle a, b, f, c, e, f, g\rangle, tg_1\rangle$ est une abf-projection de α_1.
$\langle id_1, \langle a, b, f, g\rangle, tg_1\rangle$ ne l'est pas parce que la condition (5) n'est pas satisfaite.
De même :
$\langle id_4, \langle a, b, f, a, e, b, f, d\rangle, tg_2\rangle$ est une abf-projection de α_4, alors que $\langle id_4, \langle a, b, f, d\rangle, tg_2\rangle$ ne l'est pas à cause de (5).

La s'-projection de α, si elle existe (i.e. si s' peut être un préfixe d'un n-uplet dont la séquence est contenue dans celle de α) est unique. C'est *la* s'-projection de α.

Définition 3.2.15 (Postfixe) Soit α un n-uplet de CSD et $s = \langle e_1, ..., e_n\rangle$ une séquence de I. Soit $\alpha' = \langle id_1, \langle e_1, ..., e_n, e_{n+1}, ..., e_{n+p}\rangle, tg_1\rangle$ une s-projection de α, telle que $s \sqsubseteq \text{seq}(\alpha)$.

Alors $\gamma = \langle id_1, \langle e_{n+1}, ..., e_{n+p}\rangle, tg_1\rangle$ est un *s-postfixe* de α'. Si $p > 0$, alors le s-postfixe a une séquence de taille > 0 : il est dit non vide et est noté par α/s.

Propriété 3.2.1 γ satisfait : $seq(\alpha') = s \cdot seq(\gamma)$, c'est-à-dire $seq(\gamma)$ est le suffixe de $seq(\alpha')$.

Définition 3.2.16 La *base de données s-projetée*, notée $s - \text{postfix}(CSD)$, est définie comme suit :
$s{-}\text{postfix}(CSD) = \{(\alpha/s),\ \alpha \in CSD\}$

Exemple :

$abf{-}\text{postfix}(CSD)$:

id	seq	target
$\langle id_1,$	$\langle c, e, f, g\rangle$	$, tg_1\rangle$
$\langle id_2,$	$\langle g\rangle$	$, tg_1\rangle$
$\langle id_4,$	$\langle a, e, b, f, d\rangle$	$, tg_2\rangle$

Le principe de l'algorithme récursif que nous présenterons au paragraphe 3.3 est basé sur les propriétés suivantes :

Propriété 3.2.2 Soit CSD une base de données séquentielles catégorisées. Soit s_1 et s_2 deux séquences de I, et r une règle d'association séquentielle. Alors :

(i) $s_2{-}\text{postfix}(s_1{-}\text{postfix}(CSD)) = s_1 \cdot s_2{-}\text{postfix}(CSD)$
(ii) $\text{support}_{s_1 \cdot s_2 - \text{postfix}(CSD)}(r) = \text{support}_{CSD}((s_1 \cdot s_2 \cdot \text{ant}(r)) \rightarrow \text{cons}(r))$
(iii) $\text{support}_{CSD}(r) \geq \text{support}_{s_1 - \text{postfix}(CSD)}(r)$

Preuve 3.2.1 (i) $s_2{-}\text{postfix}(s_1{-}\text{postfix}(CSD)) = s_1 \cdot s_2{-}\text{postfix}(CSD)$
I) Montrons que $s_2 - \text{postfix}(s_1 - \text{postfix}(CSD)) \subseteq s_1 \cdot s_2 - \text{postfix}(CSD)$.
Soit $\gamma_2 \in s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$. Montrons que $\gamma_2 \in s_1 \cdot s_2 - \text{postfix}(CSD)$.
Si $\gamma_2 \in s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$ alors par définition :
s_2 est une séquence ;
il existe un n-uplet $\alpha_2 \in s_1 - \text{postfix}(CSD)$;
il existe un n-uplet α_2' tel que α_2' est une $s_2 -$ projection de α_2 et $\gamma_2 = \alpha_2/s_2$
Mais,
α_2' est une $s_2 -$ projection de α_2 signifie que :

- s_2 est un préfixe de α'_2, plus précisément, $\text{seq}(\alpha'_2) = s_2 \cdot \text{seq}(\gamma_2)$ $(\diamond)$
- $\text{id}(\alpha_2) = \text{id}(\alpha'_2) = \text{id}(\gamma_2)$ (∇)
- $\text{target}(\alpha_2) = \text{target}(\alpha'_2) = \text{target}(\gamma_2)$ $(\heartsuit)$
- $\text{seq}(\alpha'_2) \sqsubseteq \text{seq}(\alpha_2)$ $(\square)$

et il n'existe aucun n-uplet α''_2 tel que $\text{seq}(\alpha'_2) \sqsubset \text{seq}(\alpha''_2) \sqsubseteq \text{seq}(\alpha_2)$ et s_2 est un préfixe de α''_2.

Maintenant $\alpha_2 \in s_1 - \text{postfix}(CSD)$ signifie que :
s_1 est une séquence ;
il existe un n-uplet $\alpha_1 \in CSD$;
il existe α'_1 un n-uplet tel que α'_1 est une $s_1 -$ projection de α_1, et $\alpha_2 = \alpha_1/s_1$
Mais,
$\alpha'_1 = s_1 - \text{projection}(\alpha_1)$ signifie :

- s_1 est un préfixe de α'_1, plus précisément, $\text{seq}(\alpha'_1) = s_1 \cdot \text{seq}(\alpha_2)$ $(\diamond\diamond)$
- $\text{id}(\alpha_1) = \text{id}(\alpha'_1) = \text{id}(\alpha_2)$ $(\nabla\nabla)$
- $\text{target}(\alpha_1) = \text{target}(\alpha'_1) = \text{target}(\alpha_2)$ $(\heartsuit\heartsuit)$
- $\text{seq}(\alpha'_1) \sqsubseteq \text{seq}(\alpha_1)$ $(\square\square)$

et il n'existe pas de n-uplet α''_1 tel que $\text{seq}(\alpha'_1) \sqsubset \text{seq}(\alpha''_1) \sqsubseteq \text{seq}(\alpha_1)$ et s_1 est un préfixe de α''_1.

Considérons $\alpha_1 \in CSD$, et montrons que $\gamma_2 = \alpha_1/s_1 \cdot s_2$.
Considérons $\delta' = \langle id(\gamma_2), \langle s_1 \cdot s_2 \cdot seq(\gamma_2)\rangle, target(\gamma_2)\rangle$.
Montrons que δ' est une $s_1 \cdot s_2-$projection de α_1

- $\text{id}(\alpha_1) = \text{id}(\alpha_2)(\nabla\nabla)$ et $\text{id}(\alpha_2) = \text{id}(\gamma_2)(\nabla)$. Pour cette raison, $\text{id}(\delta') = \text{id}(\alpha_1)$
- $s_2 \cdot \text{seq}(\gamma_2) \sqsubseteq \text{seq}(\alpha_2)(\diamond)$ et $(\square)$
  et $s_1 \cdot \text{seq}(\alpha_2) \sqsubseteq \text{seq}(\alpha_1)(\diamond\diamond)$ et $(\square\square)$
  Pour cette raison, $s_1 \cdot s_2 \cdot \text{seq}(\gamma_2) \sqsubseteq \text{seq}(\alpha_1)$, i.e., $\text{seq}(\delta') \sqsubseteq \text{seq}(\alpha_1)$
- $\text{target}(\alpha_1) = \text{target}(\alpha_2)(\heartsuit\heartsuit)$ et $\text{target}(\alpha_2) = \text{target}(\gamma_2)(\heartsuit)$. Pour cette raison $\text{target}(\delta') = \text{target}(\alpha_1)$
- $s_1 \cdot s_2$ est clairement un préfixe de δ'.
- Nous raisonnons par l'absurde. Supposons qu'il existe un n-uplet δ'' tel que
  $\text{seq}(\delta'') = s_1 \cdot s_2 \cdot \text{seq}(\varepsilon) \sqsubseteq \text{seq}(\alpha_1)$
  Supposons que $\text{seq}(\gamma_2) \not\sqsubseteq \text{seq}(\varepsilon)$.

Nécessairement, il existe dans $\text{seq}(\alpha_1)$ une autre occurrence de s_2 à gauche de celle de δ'. Mais comme α'_2 est une $s_2 -$ projection de α_2, cette occurrence de gauche de s_2 doit commencer dans α_1 avant le début de α_2 (propriété (5) de la s_2-projection). Par conséquent, il doit exister dans α_1 une autre occurrence de s_1 à gauche de celle de δ'. C'est une contradiction avec le fait que α_2 est une $s_1 -$ projection de α_1. Nous pouvons en conclure qu'il n'existe pas un tel δ''.

Ainsi, $\delta' = s_1 \cdot s_2 - \text{projection}(\alpha_1)$ et $\gamma_2 = \alpha_1/s_1 \cdot s_2$. Comme $\alpha_1 \in CSD, \gamma_2 \in s_1 \cdot s_2 \cdot \text{postfix}(CSD)$. $(\square)$

II) Montrons maintenant que $s_1 \cdot s_2 - \text{postfix}(CSD) \subseteq s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$
Soit $\gamma_4 \in s_1 \cdot s_2 - \text{postfix}(CSD)$. Montrons que $\gamma_4 \in s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$.

Si $\gamma_4 \in s_1 \cdot s_2 - \text{postfix}(CSD)$ par définition il existe $\alpha_4 \in CSD$ tel que $\gamma_4 = \alpha_4/s_1 \cdot s_2$.

Soit $\beta = \alpha_4/s_1$ alors clairement $\beta \in s_1 - \text{postfix}(CSD)$.
Soit $\delta = \beta/s_2$ alors clairement $\delta \in s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$.
Montrons que $\delta = \alpha_4/s_1 \cdot s_2$.
Nous avons $\text{id}(\delta) = \text{id}(\beta) = \text{id}(\alpha_4)$ et $\text{target}(\delta) = \text{target}(\beta) = \text{target}(\alpha_4)$.

Soit $\delta' = \langle \text{id}(\alpha_4), s_1 \cdot s_2 \cdot \text{seq}(\delta), \text{target}(\alpha_4) \rangle$
$\text{seq}(\delta') = s_1 \cdot s_2 \cdot \text{seq}(\delta)$
Trivialement δ est un $s_1 \cdot s_2-$postfixe de δ'
Montrons que δ' est une $s_1 \cdot s_2-$projection de α_4 :

- $\text{id}(\delta') = \text{id}(\alpha_4)$
- Est-ce que $\text{seq}(\delta') \sqsubseteq \text{seq}(\alpha_4)$?
  $\beta = \alpha_4/s_1$ implique que $s_1 \cdot \text{seq}(\beta) \sqsubseteq \text{seq}(\alpha_4)$
  $\delta = \beta/s_2$ implique que $s_2 \cdot \text{seq}(\delta) \sqsubseteq \text{seq}(\beta)$
  Par conséquent $s_1 \cdot s_2 \cdot \text{seq}(\delta) \sqsubseteq \text{seq}(\alpha_4)$ et donc
  $\text{seq}(\delta') \sqsubseteq \text{seq}(\alpha_4)$
- $\text{target}(\delta') = \text{target}(\alpha_4)$
- $s_1 \cdot s_2$ est clairement un préfixe de δ'
- Montrons qu'il n'existe aucun n-uplet δ'' tel que $\text{seq}(\delta'')$ ait $s_1 \cdot s_2$ comme préfixe et $\text{seq}(\delta') \sqsubset \text{seq}(\delta'') \sqsubseteq \text{seq}(\alpha_4)$.

Maintenant puisque $\beta = \alpha_4/s_1$, β est la plus grande sous-séquence de α_4 après s_1 et puisque $\delta = \beta/s_2$, δ est la plus longue sous-séquence de β après s_2. Par conséquent, δ est la plus longue sous-séquence de α_4 après $s_1 \cdot s_2$. Il ne peut donc pas exister δ'' tel que $\text{seq}(\delta') \sqsubset \text{seq}(\delta'') \sqsubseteq \text{seq}(\alpha_4)$.
Nous pouvons conclure que $\delta = \alpha_4/s_1 \cdot s_2 = \gamma_4$ et donc que $\gamma_4 \in s_2 - \text{postfix}(s_1 - \text{postfix}(CSD))$.

Preuve 3.2.2 Montrons que $\text{support}_{s_1 \cdot s_2 - \text{postfix}(CSD)}(r) = \text{support}_{CSD}((s_1 \cdot s_2 \cdot \text{ant}(r)) \rightarrow \text{cons}(r))$.

(A) D'abord prouvons que si $\gamma = \langle id, s, tg \rangle \in s_1 \cdot s_2 - \text{postfix}(CSD)$ est tel que $\text{ant}(r) \sqsubseteq s$ et $\text{cons}(r) = tg$ alors il existe $\alpha \in CSD$ tel que $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq \text{seq}(\alpha)$ et $\text{cons}(r) = \text{target}(\alpha)$.
Si $\gamma \in s_1 \cdot s_2 - \text{postfix}(CSD)$ alors il existe $\alpha \in CSD$ tel que $\text{target}(\gamma) = \text{target}(\alpha)$ et $s_1 \cdot s_2 \cdot \text{seq}(\gamma) \sqsubseteq \text{seq}(\alpha)$.
Pour cette raison $\text{target}(\alpha) = \text{cons}(r) = tg$.
Comme $\text{ant}(r) \sqsubseteq \text{seq}(\gamma)$, $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq s_1 \cdot s_2 \cdot \text{seq}(\gamma)$ et $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq \text{seq}(\alpha)$. Ainsi, $\alpha \in CSD$ et α supporte la règle $s_1 \cdot s_2 \cdot \text{ant}(r)$.

(B) Considérons maintenant $\alpha = \langle id, s, tg \rangle \in CSD$ tel que $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq s$ et $\text{cons}(r) = tg$.
Soit $\gamma = s_1 \cdot s_2 - \text{postfix}(\alpha)$, montrons que γ supporte r, i.e. $\text{ant}(r) \sqsubseteq \text{seq}(\gamma)$ et $\text{cons}(r) = \text{target}(\gamma)$.
Or $\text{target}(\gamma) = \text{target}(\alpha) = tg = \text{cons}(r)$.
$\gamma = s_1 \cdot s_2 - \text{postfix}(\alpha)$ signifie qu'il existe $\alpha' \in s_1 \cdot s_2 - \text{projection}(\alpha)$ tel que :
$\text{seq}(\alpha') = s_1 \cdot s_2 \cdot \text{seq}(\gamma)$
$\text{seq}(\alpha') \sqsubseteq \text{seq}(\alpha)$
et $\nexists \alpha''$ un n-uplet tel que $\text{seq}(\alpha') \sqsubset \text{seq}(\alpha'')$ et $\text{seq}(\alpha'') \sqsubseteq \text{seq}(\alpha)$ et $s_1 \cdot s_2$ est un préfixe de α''.
Nous avons $\text{seq}(\alpha') \sqsubseteq \text{seq}(\alpha) = s$, i.e., $s_1 \cdot s_2 \cdot \text{seq}(\gamma) \sqsubseteq s$.
Mais $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq s$ par hypothèse.

Comme seq(γ) est la plus longue sous-séquence de seq(α) ayant $s_1 \cdot s_2$ comme préfixe, $s_1 \cdot s_2 \cdot \text{ant}(r) \sqsubseteq \text{seq}(\alpha)$ implique que $\text{ant}(r) \sqsubseteq \text{seq}(\gamma)$. Par conséquent, γ supporte r.

En conclusion, pour chaque $\gamma \in s_1 \cdot s_2 - \text{postfix}(CSD)$ tel que γ supporte r, il existe $\alpha \in CSD$ qui supporte $s_1 \cdot s_2 \cdot \text{ant}(r) \rightarrow \text{cons}(r)$ et inversement.
La propriété (ii) est donc établie.

Preuve 3.2.3 Montrons que pour tout $\gamma \in s_1 - \text{postfix}(CSD)$ qui supporte r, il existe $\alpha \in CSD$ qui supporte r.

$\gamma = \langle id, s, tg \rangle \in s_1 - \text{postfix}(CSD)$ est tel que $\text{ant}(r) \sqsubseteq s$ et $\text{cons}(r) = tg$. Alors, il existe $\alpha \in CSD$ tel que $\text{target}(\gamma) = \text{target}(\alpha)$ et $s_1 \cdot \text{seq}(\gamma) \sqsubseteq \text{seq}(\alpha)$. On a donc $s_1 \cdot s \sqsubseteq \text{seq}(\alpha)$ et par suite $s_1 \cdot \text{ant}(r) \sqsubseteq \text{seq}(\alpha)$. Par conséquent, $\alpha \in CSD$ et α supporte la règle $s_1.\text{ant}(r) \rightarrow \text{cons}(r))$.
En conclusion, pour chaque $\gamma \in s_1 - \text{postfix}(CSD)$ tel que γ supporte r, il existe $\alpha \in CSD$ qui supporte $s_1 \cdot \text{ant}(r) \rightarrow \text{cons}(r)$.
La propriété (iii) est donc établie.

3.3 L'algorithme SNK

3.3.1 Présentation

L'algorithme SNK recherche des c-SNoKs et des s-SNoKs dans une base de données séquentielles catégorisées. Il utilise les propriétés (i), (ii) et (iii) (propriétés 3.2.2). Étant donnée une propriété ciblée (par exemple, une fonction protéique), SNK construit les règles séquentielles qui concluent sur cette propriété cible récursivement, en agrandissant la taille des règles et en testant la minimalité de chaque c-SNoK à chaque appel récursif (cf. définition 3.2.10).

Pour commencer, SNK essaie de générer les règles d'associations séquentielles de taille 1 commençant par les items présents dans les séquences. Puis, les items qui n'ont pas permis de construire une règle sont utilisés comme des noyaux (ils serviront de base pour construire les règles à la prochaine itération), et SNK essaie de générer des SNoKs de taille 2. Cette étape est répétée récursivement.

Pseudo-code

L'extension de cet algorithme pour la recherche de c-SNoKs et de s-SNoKs est triviale.

3.3.2 Propriétés de l'algorithme

Complexité en temps
La complexité en temps de SNK est liée au nombre d'items cibles y de T présents dans CSD, et pour tout y dans ST, au nombre d'appels récursifs de SNKrec. Par conséquent, nous mesurons la complexité en estimant le nombre de tests (**si** x-postfix(S) $\neq \emptyset$) réalisés par SNKrec pour un y donné. Le pire cas pour SNKrec a lieu quand toutes les règles générées ont un bon support

Algorithme 2: Méthode SNK

Résultats : $RESULTS$ l'ensemble de toutes les pépites séquentielles de connaissances les plus générales ;
Méthode utilisée : SNKrec
Paramètres :
Entrées : CSD une base de données séquentielles catégorisées ;
min_supp un seuil de support ;
IM une mesure de qualité ;
min_meas un seuil de mesure d'intérêt ; k en entier de valeur l ou q selon que le cas c-SNoKs ou s-SNoKs.
début
  $RESULTS \leftarrow \emptyset$;
  $ST \leftarrow$ l'ensemble de toutes les cibles de T présentes dans CSD ;
  pour chaque $y \in ST$ **faire**
    //On cherche les pépites de connaissances ciblées sur y
    $S_y \leftarrow$ l'ensemble de tous les n-uplets de CSD ayant y pour cible ;
    SNKrec(S_y,y,min_supp,IM,min_meas,$\langle\rangle$,k,$RESULTS$)
  finPourChaque
fin

mais une mauvaise mesure de qualité, conduisant à un nombre maximum d'appels récursifs.
Considérons l'arbre des appels récursifs de SNKrec. Soit S la base projetée au i-ème niveau et soit rs_y la profondeur de cet arbre (c'est la longueur de la plus longue séquence d'un n-uplet de CSD). Soit $rs_{y,i}$ la longueur de la plus grande séquence d'un n-uplet de S. Clairement $rs_{y,i} \leq rs_y - i$. Soit l_i le cardinal maximum de l'ensemble SI considéré au i-ème niveau. Une borne supérieure pour le nombre de tests au i-ème niveau est donc $\prod_{j=0}^{i-1} l_j$. Soit $\sigma_{y,i}$ le nombre maximum de n-uplets pouvant être trouvés au i-ème niveau dans n'importe quelle base projetée S.

Alors la construction de $x - \text{postfix}(S)$ au i-ème niveau nécessite $\mathcal{O}(\sigma_{y,i} rs_{y,i})$ opérations. Finalement, une borne supérieure pour le nombre total d'opérations réalisées par SNK pour construire les bases projetées est :

$$\mathcal{O}(\sum_{y \in ST} \sum_{i=1}^{rs_y v} (\prod_{j=0}^{i-1} l_j) \text{Max}((\sigma_{y,i} rs_{y,i}), (|CSD| C_{rs_{y,0}}^{\text{int}(sr_{y,0}/2)})))$$

A chacun de ces tests correspond un calcul du support ${}_S((p \cdot x) \rightarrow y)$ qui est supérieur ou égal à $\sigma_{y,i} rs_{y,i}$. Le calcul de mesure ${}_{IM,CSD}((p \cdot x) \rightarrow y)$ est simple : c'est le résultat de divers calculs de probabilités effectués une fois pour toutes avant l'exécution de SNK. À chaque appel récursif de l'algorithme, une base de données séquentielles catégorisées peut produire un nombre de règles égal au plus au nombre de combinaisons de sous-séquences de la plus longue séquence, multiplié par le nombre de séquences. À chaque appel récursif, le coût est au plus soit le coût d'une postfix-projection, soit le coût de la méthode add_rule. Avec notre approche de recherche en profondeur, toutes les bases projetées n'ont pas besoin d'être stockées en mémoire contrairement aux approches en largeur. De plus, le calcul des différentes projections peut être réalisé indépendamment.
Cette analyse montre que la complexité en temps théorique est très élevée dans le pire cas.

Algorithme 3: Méthode SNKrec cas des k-c-SNoKs

Spécification : SNKrec(S,y,min_supp,IM,min_meas,p,k,$RESULT$) génère des règles r de la forme $(p \cdot x) \rightarrow y$ avec $|p \cdot x| \geq k$, où x est un item apparaissant dans S et p le préfixe utilisé ;
Méthodes utilisées : add_rule ; //add_rule(r,RES) ajoute la règle r à RES sauf si r est moins générale ou égale à une règle de RES et enlève de RES toutes règles moins générales que r.
support ; // support$_S(r)$ évalue le support de r dans S.
measure ; // measure$_{IM,CSD}$(r) évalue la valeur de r pour IM dans CSD
Paramètres :
Entrées : S l'ensemble des n-uplets ayant y comme cible ;
min_supp, IM, min_meas ;
p une séquence utilisée comme préfixe ;
k un entier définissant la taille minimum des k-c-SNoKs.
Entrées/Sorties : $RESULTS$

```
début
    SI ← l'ensemble de tous les items de I apparaissant comme élément de S ;
    pour chaque x ∈ SI faire
        si support_S(x → y) ≥ min_supp alors
            si |p · x| ≥ k et measure_{IM,CSD}((p · x) → y) ≥ min_meas alors
                RESULTS ← add_rule((p · x) → y,k,RESULTS)
            sinon
                si x − postfix(S) ≠ ∅ alors
                    SNKrec(x−postfix(S),y,min_supp,IM,min_meas,p·x,k,RESULTS)
                finSi
            finSi
        finSi
    finPourChaque
fin
```

Cependant, en pratique, pour les applications prévues, l'algorithme SNK reste efficace du fait de la diminution très rapide de la taille des bases projetées et de la faible longueur des n-uplets.

Correction et complétude

La correction et la complétude de SNK sont présentées ici dans le cas des 1-c-SNoKs. Elles se démontrent de façon analogue pour les k-c-SNoKs et les q-s-SNoKs.

Théorème 1 (Correction) Soit CSD une base de données séquentielles catégorisées d'enregistrements. Soit $RESULTS$ l'ensemble de toutes les pépites séquentielles de connaissance renvoyées par SNK($CSD, min\_supp, IM, min\_meas$) pour une mesure d'intérêt IM. Alors :
(1) La valeur du support dans CSD de chaque $r \in RESULTS$ est supérieure ou égale à min_supp et sa valeur de mesure de qualité dans CSD est supérieure ou égale à min_meas.
(2) Pour tout $r \in RESULTS$ il n'existe pas de règle r' sur CSD telle que support$_{CSD}(r') \geq min_supp$, measure$_{IM,CSD}(r') \geq min_meas$ et $r' \prec r$.

Algorithme 4: Méthode SNKrec cas des q-s-SNoKs

Spécification : **SNKrec**(S,y,min_supp,IM,min_meas,p,q,$RESULT$) génère des règles r de la forme $(p \cdot x) \rightarrow y$ avec $|p \cdot x| \leq q$ et $q \geq 1$, où x est un item apparaissant dans S et p le préfixe utilisé ; Met à jour $RESULTS$ avec r ;
Méthodes utilisées :
support ; support$_S(r)$ évalue le support de r dans S.
measure ; measure$_{IM,CSD}$(r) évalue la valeur de r pour IM dans CSD

Paramètres :
Entrées : S l'ensemble des n-uplets ayant y comme cible ;
min_supp, IM, min_meas ;
p une séquence utilisée comme préfixe ;
q un entier, définissant la taille maximale des q-s-SNoKs.
Entrées/Sorties : $RESULTS$ l'ensemble des pépites séquentielles de connaissance relâchées de taille inférieure ou égale à q.

```
début
    SI ← l'ensemble de tous les items de I apparaissant comme élément de S ;
    pour chaque x ∈ SI faire
        si support_S(x → y) ≥ min_supp alors
            si measure_{IM,CSD}((p · x) → y) ≥ min_meas alors
                RESULTS ← RESULTS ∪ {(p · x) → y}
            sinon
                si x − postfix(S) ≠ ∅ et |p · x| < q alors
                    SNKrec(x − postfix(S),y,min_supp,IM,min_meas,p·x,q,RESULTS)
                finSi
            finSi
        finSi
    finPourChaque
fin
```

Preuve 3.3.1 (Correction) .
Dans la méthode SNKrec appelée pour $S, y, min\_supp, IM, min\_meas$ et p, la méthode add_rule est appelée pour la règle $r = (p \cdot x) \rightarrow y$ si :

(a) support$_s(x \rightarrow y) \geq min_supp$ et

(b) measure$_{IM,CSD}(p \cdot x \rightarrow y) \geq min_meas$

(1) Soit r une règle dans $RESULTS$. Elle a été ajoutée à $RESULTS$ (et non détruite) par la méthode add_rule appelée par SNKrec pour certaines valeurs des paramètres S et p. Soit k le nombre d'appels récursifs nécessaires.
Si $k = 1$ alors r est de la forme $x \rightarrow y$ et un seul appel est nécessaire (la méthode étant elle-même appelée par la méthode principale SNK). Par conséquent, p est un préfixe vide et $S = S_y$, l'ensemble des n-uplets de CSD ayant y comme cible.
Si $k > 1$, alors r est de la forme $(x_1, x_2, ..., x_{k-1}, x_k) \rightarrow y$, puisque chaque appel récursif ajoute un item dans l'antécédent de la règle.

Algorithme 5: Méthode add_rule

Spécification : add_rule(r,k,RES) ajoute la règle r à RES sauf si r est moins générale ou égale à une autre règle de RES et enlève de RES toutes les règles moins générales que r

Paramètres :

Entrées : r une règle ;

k la taille minimale de la règle (on cherche des k-c-SNoKs) Entrées/Sorties : RES un ensemble de n règles $\{r_1, ..., r_n\}$ de longueur supérieure ou égale à k telles qu'il n'existe pas $\rho_1, \rho_2 \in RES$ avec $\rho_1 \prec \rho_2$;

début

max_found ← false ;

$i \leftarrow 1$;

tant que $i \leq n$ ***et*** $max\_found = false$ **faire**

si $r \prec r_i$ *et* $|r_i| \geq k$ **alors**

$RES \leftarrow RES \setminus \{r_i\}$

sinon

si $r_i \preceq r$ **alors**

$max\_found \leftarrow true$

finSi

finSi

$i = i + 1$

finTq

si $max\_found = false$ **alors**

$RES \leftarrow RES \cup \{r\}$

finSi

fin

Les appels récursifs successifs de SNKrec ont été réalisés avec les valeurs suivantes de S : $S_1, S_2, ..., S_k$ avec $S_1 = S_y$, $S_2 = x_1 - \text{postfix}(S_1)$, $S_3 = x_2 - \text{postfix}(S_2)$, ..., $S_k = x_{k-1} - \text{postfix}(S_{k-1})$. Les valeurs successives de p sont : $\emptyset, x_1, x_1 \cdot x_2, ..., x_1 \cdot x_2 \cdot ... \cdot x_{k-1}$.
Par conséquent $S_k = x_{k-1} - \text{postfix}(x_{k-2} - \text{postfix}(...(x_1 - \text{postfix}(S_y))...))$.
Par la propriété 3.2.2 (i), $S_k = x_1 \cdot x_2 \cdot ... \cdot x_{k-1} - \text{postfix}(S_y)$
Maintenant, par la propriété 3.2.2 (ii),
$\text{support}_{x_1 \cdot x_2 \cdot ... \cdot x_{k-1} - \text{postfix}(S_y)}(x_k \rightarrow y) = \text{support}_{S_y}((x_1 \cdot ... \cdot x_{k-1}) \cdot x_k \rightarrow y)$
Puisque seuls les n-uplets de CSD ayant y comme cible sont utiles pour le calcul de support des règles ayant y comme conséquent, nous avons :

$$\text{support}_{x_1 \cdot x_2 \cdot ... \cdot x_{k-1} - \text{postfix}(S_y)}(x_k \rightarrow y) = \text{support}_{CSD}(r)$$

Mais puisqu'au k^{eme} appel de SNKrec, la règle r est obtenue par add_rule, nous avons : $\text{support}_{S_{k-1}}(x_k \rightarrow y) \geq min\_supp$.
Par conséquent :

$$\text{support}_{CSD}(r) \geq min\_supp$$

(2) Raisonnons par l'absurde. Suposons qu'il existe une règle r' définie sur CSD, avec $r' \prec r$ et telle que $\text{support}_{CSD}(r') \geq min\_supp$ et $\text{measure}_{IM,CSD}(r') \geq min\_meas$. Alors $r' = x_{p_1} \cdot ... \cdot x_{p_q} \rightarrow y$ avec $p_1, ..., p_q \in \{1, ..., k\}$, $p_1 < ... < p_q$ et $q < k$.

Il existe une succession d'appels récursifs qui contruisent le début de ant(r').

Supposons que r' est complètement construite : alors au q^{eme} appel de SNKrec, add_rule aurait été appelée et aurait éliminé la règle r puisque $r' \prec r$. C'est impossible. Par conséquent, le processus de construction de r' a été stoppé aux j^{eme} appel récursif de SNKrec, avec $j < q$.
Soit $r'' = x_{p_1} \cdot x_{p_2} \cdot ... \cdot x_{p_j} \rightarrow y$.
De la même façon que dans (1), nous pouvons prouver que $\text{support}_{CSD}(r'') = \text{support}_{S_j}(x_{p_j} \rightarrow y)$ où S_j est la valeur du paramètre S au j^{eme} appel.
Mais puisque $r'' \prec r' \prec r$, nous avons $\text{support}_{CSD}(r'') \geq \text{support}_{CSD}(r) \geq min\_supp$, par définition du support.

Maintenant, puisqu'aucun autre appel récursif n'a été fait, $\text{measure}_{IM,CSD}(r'') \geq min\_meas$.
Donc SNKrec aurait dû appeler la méthode add_rule qui aurait dû ajouter r'' à $RESULTS$.
Comme $r'' \prec r$, r aurait dû être enlevée de $RESULTS$, une contradiction.
En conclusion, il n'existe pas de telle règle r'.

Théorème 2 (Complétude) Soit CSD une base de données séquentielles catégorisées, IM une mesure de qualité, min_supp un seuil de support et min_meas un seuil de mesure de qualité pour IM.

Soit $Q_{CSD,min\_supp,IM,min\_meas}$ = { règles r sur CSD satisfaisant $\text{support}_{CSD}(r) \geq min\_supp$, $\text{measure}_{IM,CSD}(r) \geq min\_meas$ et telles que $\nexists\ r'$ sur CSD avec $\text{support}_{CSD}(r') \geq min\_supp$, $\text{measure}_{IM,CSD}(r') \geq min\_meas$ et $r' \prec r$}.

Alors n'importe quelle règle de $Q_{CSD,min\_supp,IM,min\_meas}$ peut être obtenue par SNK(CSD, min_supp, IM, min_meas).

Preuve 3.3.2 (Complétude) Notons Q l'ensemble $Q_{CSD,min\_supp,IM,min\_meas}$, pour simplifier. Soit $r \in Q$. Montrons que r peut être obtenue par SNK. Supposons que r est de la forme $x_1 \cdot ... \cdot x_k \rightarrow y$.

Nous montrons que r est retournée par SNK après k appels récursifs de SNKrec.
Cas de base : $k = 1$. Alors SNK appelle SNKrec avec $S = S_y$ et $p = \emptyset$. Comme $r \in Q$, $\text{support}_{CSD}(x_1 \rightarrow y) \geq min\_supp$. Mais comme montré précédemment $\text{support}_{CSD}(x_1 \rightarrow y) = \text{support}_{S_y}(x_1 \rightarrow y)$. Puisque $r \in Q$, $\text{measure}_{IM,CSD}(x_1 \rightarrow y) \geq min\_meas$.
Par conséquent, au premier appel de SNKrec, la méthode add_rule est appelée. Puisque $r \in Q$, il n'existe pas de règle $r' \in RESULTS$ contenue dans r. Par conséquent r est ajoutée à $RESULTS$ et y restera jusqu'à la fin de l'algorithme puisqu'il n'existe pas de règle plus générale.

Cas général : $k > 1$. Montrons que SNKrec réalise successivement k appels récursifs pour retourner r.
Soit $S_1 = S_y, S_2 = x_1 - \text{postfix}(S_1), ..., S_k = x_{k-1} - \text{postfix}(S_{k-1})$ les valeurs successives du paramètre S pour les k premiers appels de SNKrec.
Soit $r'_j = x_1 \cdot ... \cdot x_j \rightarrow y$ pour $1 \leq j \leq k-1$.

Puisque $\text{support}_{CSD}(r) \geq min\_supp$, on a : $\text{support}_{S_y}(r) \geq min\_supp$ (a)

Par la propriété 3.2.2 (i) et (ii), on a :
$\text{support}_{S_j}(x_j \rightarrow y) = \text{support}_{S_y}(x_1 \cdot ... \cdot x_j \rightarrow y)$, $1 \leq j \leq k-1$
Mais $\text{support}_{S_y}(x_1 \cdot ... \cdot x_j \rightarrow y) \geq \text{support}_{S_y}(x_1 \cdot ... \cdot x_k \rightarrow y) \geq min\_supp$ car $j < k$ et $\text{support}_{Sy}(x_1 \cdot ... \cdot x_k \rightarrow y) \geq min\_supp$.
Nous avons donc :
$\text{support}_{CSD}(r'_j) = \text{support}_{S_y}(r'_j) \geq min\_supp$.
D'où $r'_j \prec r$ pour $1 \leq j \leq k-1$ (b),
Et enfin r'_j est une règle définie sur CSD (c).
De (a), (b), (c) et du fait que $r \in Q$, nous pouvons déduire que $\text{measure}_{IM,CSD}(r'_j) < min\_meas$.
Par conséquent, SNKrec sera appelé une fois de plus et ce avec le paramètre S_{j+1}.

Nous avons montré que k appels récursifs seront effectués, conduisant à la règle r au k^{eme} appel. Lors de ce dernier appel de SNKrec, SNKrec appellera la méthode add_rule qui testera si r peut être ajoutée à $RESULTS$. Comme $r \in Q$, il ne peut pas exister une règle $r' \in RESULTS$ telle que $r' \prec r$, puisque toute règle de $RESULTS$ satisfait les conditions de seuil pour le support et la mesure de qualité (cf. partie correction).

En conclusion r sera ajoutée à $RESULTS$, sans possibilité de l'y enlever ultérieurement, et r sera renvoyée par la méthode principale SNK.

Les théorèmes 1 et 2 conduisent à la spécification de la sortie de SNK suivante :

Théorème 3 (Correction et complétude de SNK) .
Soit CSD une base de données séquentielles catégorisées et IM une mesure de qualité. Étant donnés un seuil de support et un seuil de mesure de qualité, SNK retourne exactement toutes les pépites séquentielles de connaissances minimales sur CSD pour IM.

3.4 Recherche de pépites séquentielles de connaissance : l'outil SNK

Nous avons développé un prototype de SNK utilisable en ligne à l'adresse http://www.lri.fr/~rance/SNK. L'application est programmée en Java et est principalement dédiée à une étude des domaines protéiques sans que ce soit exclusif.

Dans le cas d'un usage général, SNK considère des séquences annotées par l'utilisateur en employant un format spécifique. Il peut choisir de modifier les réglages par défaut. Il peut ainsi changer les valeurs de seuil de support, de mesure de qualité, de seuil de mesure de qualité, le type de pépites recherchées (SNoKs, k-c-SNoKs ou q-s-SNoKs), et le cas échéant définir la valeur de k ou de q.

Dans le cas spécifique de l'étude des domaines protéiques, SNK est doté de fonctionnalités supplémentaires. L'outil permet de prendre en entrée le résultat d'une requête sur la source de données des domaines protéiques Pfam. Il est ensuite capable d'en extraire les séquences au format adéquat. L'utilisateur n'a plus alors qu'à régler les paramètres à sa convenance. Les résultats obtenus peuvent être analysés plus facilement avec DeeVee, un visualisateur couplé à SNK et présenté dans la section suivante.

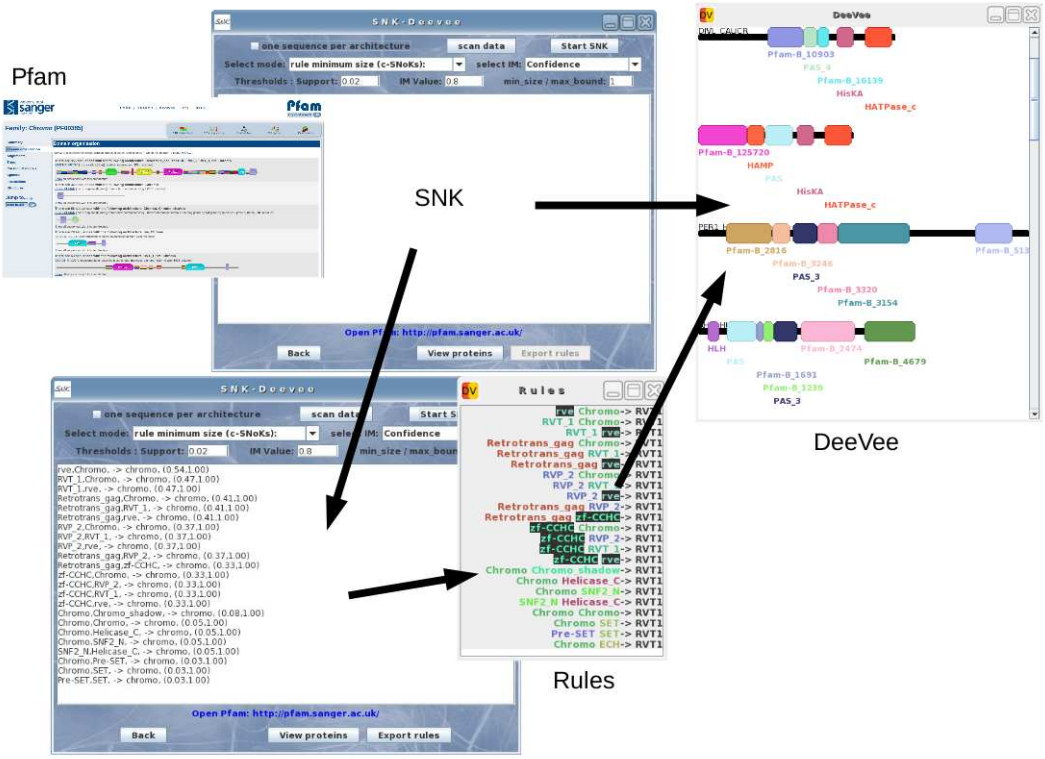


FIG. 3.1 – *Schéma global du fonctionnement de SNK*

Quelques fonctionnalités annexes sont disponibles dans SNK, notamment pour que l'outil aille chercher automatiquement les architectures des protéines à partir d'une liste d'identifiants Swiss-Prot. Il peut aussi utiliser la fonction de remplacement pour modifier les architectures en cours d'étude (par exemple remplacer tous les domaines RVT_1 et RVT_2 par RVT).

Le manuel d'utilisation complet est disponible en ligne sur le site de SNK : http://www.lri.fr/~rance/SNK.

3.5 Aide à l'analyse des résultats : l'outil DeeVee

DeeVee est un visualisateur de domaines protéiques facile d'utilisation. Il prend en entrée un ensemble de protéines extraites de Pfam (bouton "View proteins" de SNK). Les pépites séquentielles de connaissance trouvées avec SNK peuvent facilement être visualisées avec DeeVee (bouton "Export rules" de SNK). DeeVee présente deux fenêtres : (1) le visualisateur qui affiche l'architecture en domaines protéiques et (2) la fenêtre des règles qui affiche les règles trouvées par SNK et permet une sélection de toutes les protéines correspondant à une certaine règle. DeeVee facilite l'analyse des pépites obtenues en utilisant SNK sur un ensemble de protéines.

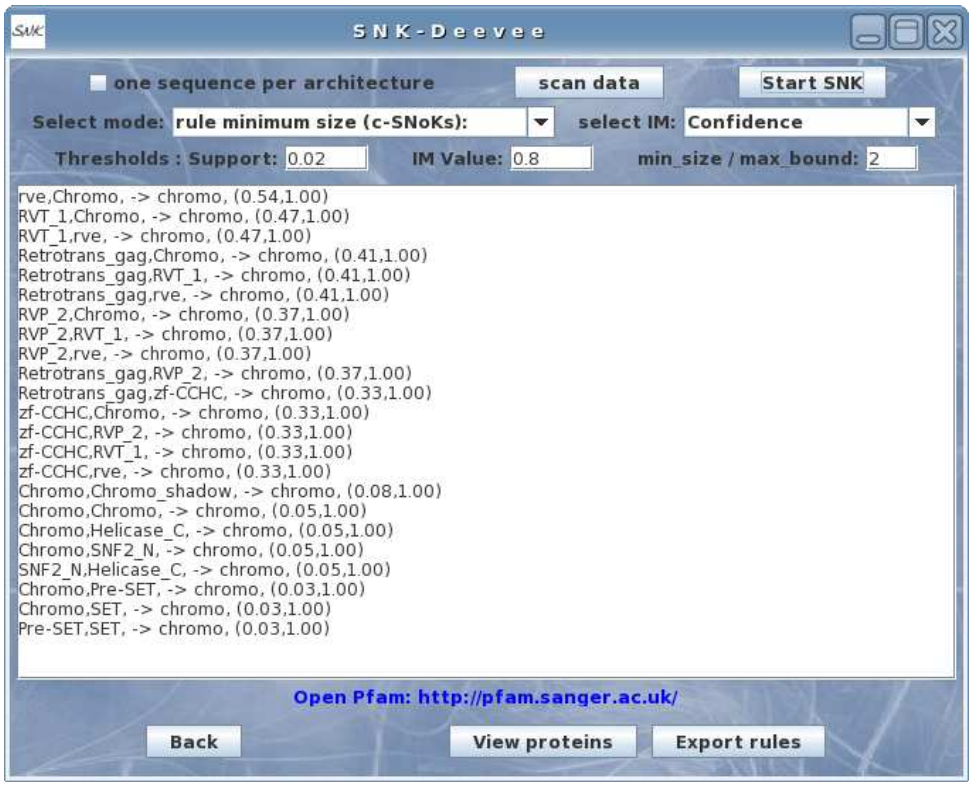


FIG. 3.2 – *Capture d'écran de l'outil SNK*

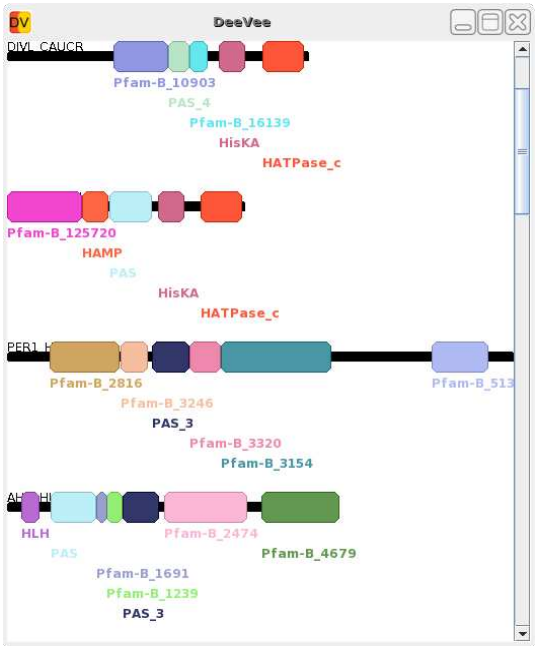


FIG. 3.3 – *Capture d'écran du logiciel DeeVee*

3.6 Exemples d'utilisation

Nous montrons maintenant l'aide qu'apporte SNK pour l'étude de familles protéiques en en explorant deux.

3.6.1 Exploration de la famille des Phospholipases D

Nous avons d'abord choisi une famille protéique bactérienne. Chaque protéine est décrite par la séquence de ses domaines. Nous appelons indifféremment domaine une partie fonctionnelle, ou une partie bien conservée de la séquence d'acides aminés. Nous considérons la famille des Phospholipases D (PLD) qui est présente dans tout l'arbre du vivant, des virus aux eucaryotes, et est impliquée dans de nombreux processus cellulaires [75]. Ces protéines sont groupées simplement parce qu'elles partagent un motif PLDc en tandem. Elles contiennent aussi un large éventail d'autres motifs. Dans [87], nous avions décrit une régularité très surprenante concernant la taille de la partie C-terminale des protéines de cette famille. Plus précisément, la distance entre la fin du second motif PLD et l'extrémité C-terminale de la protéine (la plus à droite) était corrélée aux fonctions connues des protéines.

Par conséquent, les protéines peuvent être regroupées en utilisant cette distance comme critère de classification. Dans le reste de cette section nous nous référerons aux longueurs de ces régions comme la *longueur C-terminale*. Cette longueur peut valoir : 40, 60, 72, 82, 100 acides aminés, les protéines sont ainsi regroupées en 5 classes. Chaque classe est homogène d'un point de vue fonctionnel. Nous avons utilisé SNK pour rechercher des relations potentielles entre l'architecture en domaine, la longueur C-terminale, et la fonction. Nous avons considéré toutes les protéines bactériennes de la source de données UniProtKB [13] contenant deux motifs PLDc. L'ensemble de protéines correspondant présente une variété de combinaisons de domaines, impliquant des signatures d'autres familles protéiques aussi bien que des régions de faible complexité (séquence faiblement informative [116]). Le nombre total de protéines est de 676.

Dans un premier temps, nous avons recherché un lien possible entre les régions de faible complexité et la longueur C-terminale. Dans ce premier test, les protéines étaient décrites comme une succession de motifs PLDc, de régions de faible complexité, et d'une longueur C-terminale. Nous avons étudié l'ensemble des protéines des classes "72" et "82" en utilisant la mesure de qualité de Zhang. Cette mesure permet de prendre en compte non seulement les co-occurrences de A et B pour une règle $A \rightarrow B$, mais aussi la présence de $\overline{A}$ avec B (cf 2.3), tout en restant facilement interprétable. Dans l'ensemble de ce premier exemple, nous recherchons des 1-c-SNoKs.

SNK a été utilisé avec un support de seuil très faible (min_supp=0.022) et avec un seuil de mesure de qualité assez élevé (min_meas=0.8). Ces seuils ont été définis dans un premier temps grâce aux nombres d'occurrences, d'où la précision du seuil de support. Parmi les 20 pépites séquentielles de connaissances obtenues, 3 sont particulièrement intéressantes. Dans les règles présentées ci-dessous, les régions de faible complexité sont notées lc, et les valeurs entre parenthèses sont respectivement la valeur de support et la valeur de la mesure de Zhang.

```
(1a) lc,PLDc,PLDc -> 82, (0.40,0.80),
(1b) lc,lc,PLDc -> 82, (0.25,0.68),
(1c) PLDc,PLDc,lc -> 72, (0.20,0.90)
```

Les règles d'association séquentielles retournées par SNK sont de haute qualité. Les règles (1a) et (1c) mettent en lumière l'importance de l'ordre des modules pour l'attribution à une classe. La localisation des régions de faible complexité est fortement liée à la longueur C-terminale. Selon que *lc* se trouve d'un côté ou de l'autre du double motif PLDc, la règle permet de conclure que la protéine appartient à l'une ou l'autre des classes. Des règles d'association simples n'auraient pas permis d'exprimer une telle distinction. On voit tout l'intérêt de SNK.

Dans un second temps, les informations sur les signatures des familles de protéines ont été complétées avec les données de Pfam-A [14] et Pfam-B. On calcule de nouvelles règles avec les seuils min_supp=0.01 et min_meas=0.9.
`(2a) PLDc,Pfam-B_115,Pfam-B_2786 -> 40, (0.01,0.90)` et
`(2b) PLDc,Pfam-B_115,Pfam-B_6054 -> 40, (0.01,0.93)`
Parmi toutes les règles obtenues, (2a) et (2b) sont les seules règles où le motif PLDc précède le domaine Pfam-B_115 et semblent donc caractériser la classe 40. Dans toutes les autres règles où Pfam-B_115 apparaît conjointement avec PLDc, il précède le motif.

Nous avons effectué une expérience supplémentaire avec l'ensemble de données initiales, mais en prenant pour cible de SNK la fonction des protéines (soit diacyltransferase, cardiosyntase, transphosphatidylase ou phospholipase non-spécialisé D). Parmi les 9 règles obtenues avec les seuils min_supp=0.01 et min_meas=0.9, l'une est fortement associée à la fonction cardiosyntase :
`(3a) Pfam-B_1038,lc,Pfam-B_115,Pfam-B_2786 -> cardiosynthase, (0.01,1.00)`
Cette règle semble très similaire à une règle générée avec le critère de longueur (pour les mêmes seuils)
`(2c) lc,Pfam-B_115,lc,Pfam-B_2786 -> 60, (0.02,0.94)`
De même :
`(3b) lc,Pfam-B_5151 -> diacyltransferase, (0.01,0.91)`
correspond fortement à la fonction diacyltransferase alors que
`(3c) Pfam-B_5151 -> 72, (0.08,1.00)`
était précédemment générée pour le critère de longueur.

Cela généralise les corrélations suggérées dans [87] entre les longueurs 60 et 72 respectivement et les fonctions cardiosyntase et diacyltransferase. Les autres règles générées avec les fonctions protéiques comme cibles sont potentiellement erronées du fait d'erreur dans l'attribution automatique des fonctions pour ces protéines. Une correction automatique des erreurs en utilisant les règles générées avec le critère de longueur pourrait être envisagée.

3.6.2 Exploration de la famille Chromo

Le domaine chromatin organisation modifier : Nous illustrons maintenant l'utilisation de SNK par l'étude de protéines contenant le domaine protéique chromo (CHRromatin Organisation MOdifier), identifié dans la source Pfam par la clé PF00385. Dans cette étude, la mesure de qualité choisie est la confiance, en raison de la facilité d'interprétation des résultats obtenus avec cette mesure.
D'après Pfam (version 23.0, juillet 2008), la famille protéique correspondante, Chromo_fam, contient 1975 séquences organisées en 171 architectures. Le commentaire de Pfam sur la famille est le suivant :

"Proteins that contain a chromo domain appear to fall into 3 classes. The first class includes proteins having an N-terminal chromo domain followed by a region termed the chromo shadow domain (7667093), eg. Drosophila and human heterochromatin protein Su(var)205 (HP1). The second class includes proteins with a single chromo domain, eg. Drosophila protein Polycomb (Pc) ; mammalian modifier 3 ; human Mi-2 autoantigenand and several yeast and Caenorhabditis elegans hypothetical proteins. In the third class paired tandem chromo domains are found, eg. in mammalian DNA-binding/helicase proteins CHD-1 to CHD-4 and yeast protein CHD1."

Dans un premier temps, nous avons utilisé SNK pour identifier des co-occurrences de domaines ou de motifs dans la famille Chromo_fam qui pourraient confirmer le découpage en trois classes énoncé plus haut. Nous appellerons ces classes respectivement "shadow", "tandem" ou "single". Les règles générées avec "shadow", "tandem" ou "single" comme propriété cible nous permettent d'illustrer les combinaisons de domaines.

Le seuil utilisé pour la mesure de qualité est de 0.8, celui du support de 0.015. Ces seuils permettent de bien s'assurer de la qualité des résultats obtenus avec une bonne valeur pour la mesure de qualité, tout en restant dans des seuils de support très bas (1,5%) autorisant la découverte éventuelle de pépite de connaissance.

La liste de tous les 2-c-SNoKs, ainsi que leurs valeurs de support et de mesure d'intérêt entre parenthèses sont données en annexe 5.6.2. Parmi ces règles, la plus pertinente qui conclut sur *shadow* est :

```
Chromo,Chromo_shadow, -> shadow, (0.05,1.00)
```

Cette règle est typiquement une pépite séquentielle de connaissance. Elle n'est vérifiée que par quelques protéines (5 % des protéines de la famille Chromo_fam) où le domaine Chromo est suivi par Chromo_shadow, mais la qualité de l'association est très élevée (la valeur de la confiance est de 1).

Dans une seconde étape, nous avons généré des règles de longueur 3. Nous avons essayé d'identifier des architectures caractéristiques pour les deux sous-familles *tandem* et *single*. La liste suivante décrit quelques c-SNoKs de longueur 3 obtenues pour les protéines de ces deux sous-familles. La liste complète des résultats est donnée en annexe 5.6.2. Les règles sont ordonnées par valeur décroissante de support.

```
R1: RVT_1,rve,Chromo, ->  single, (0.41,1.00)
R2: Retrotrans_gag,RVT_1,Chromo, ->  single, (0.31,1.00)
R3: Retrotrans_gag,rve,Chromo, ->  single, (0.30,1.00)
R4: RVP_2,RVT_1,Chromo, ->  single, (0.29,1.00)
R6: RVP_2,rve,Chromo, ->  single, (0.28,1.00)
R8: Retrotrans_gag,RVP_2,Chromo, ->  single, (0.27,1.00)
R10: zf-CCHC,RVT_1,Chromo, ->  single, (0.26,1.00)
R12: zf-CCHC,RVP_2,Chromo, ->  single, (0.25,1.00)
R13: zf-CCHC,rve,Chromo, ->  single, (0.25,1.00)
R15: Retrotrans_gag,zf-CCHC,Chromo, ->  single, (0.25,1.00)
R21: Chromo,Chromo,SNF2_N, ->  tandem, (0.12,1.00)
R22: Chromo,Chromo,Helicase_C, ->  tandem, (0.12,1.00)
```

```
R28: Chromo,Pre-SET,SET, ->  single, (0.02,1.00)
R30: PHD,Chromo,Chromo, ->  tandem, (0.02,1.00)}
```

Parmi les 32 règles obtenues, 3 ont pour cible *tandem* : les règles R21, R22 et R30. Ces règles ont un support faible, ce qui n'est pas étonnant puisque seules quelques protéines appartiennent à cette sous-famille, mais les associations restent très fortes. Il est intéressant de noter que l'occurrence double de Chromo dans les règles est limitée à la présence d'un domaine Helicase C (PF00271), SNF2 family N-terminal (PF00176), ou PHD (PF00628). Dans le cas particulier du domaine PHD, les occurrences répétées de Chromo ont pu ne pas être détectées dans Pfam du fait de leur proximité avec le seuil de détection. En effet si le domaine est trop dégradé ou trop différent du modèle de référence, les outils de détection de Pfam ont pu manquer le domaine. Il manque dans ce cas des occurrences du domaine Chromo, seulement en raison de leur non-détection et non en raison de leur absence réelle de la séquence. En revanche, on retrouve les occurrences de Chromo dans Prosite ou SMART, par exemple dans les protéines CHD3_HUMAN (Q12873) ou CHD4_HUMAN (Q14839).

Les règles associées à la cible *single* ont un membre gauche dont l'architecture est très variable. Le domaine Chromo est trouvé en position N-terminale dans la règle R28, alors qu'il est au contraire trouvé dans d'autres architectures en position C-terminale, comme par exemple dans les règles R1, R2 ou R12. En position N-terminale, le domaine Chromo est fortement associé avec les domaines Pre-SET et SET (respectivement identifiés par PF05033 et PF00856 dans Pfam), avec une faible valeur de support, dans la règle R8 par exemple. Dans le cas où le domaine Chromo est en position C-terminale, on le trouve associé au motif protéique Retrotransposon gag (Retrotrans_gag, PF03732), au motif zinc knuckle (zf-CCHC, PF00098), au domaine reverse transcriptase (RVT_1, PF00078), au domaine integrase core (rve, PF00665) ou au domaine retroviral aspartyl protease (RVP_2, PF08284).

Nous avons envisagé la possibilité d'une correspondance entre la position du domaine Chromo dans l'architecture de la protéine et une fonction biologique exprimée sous la forme d'un ou de plusieurs termes de la sous-ontologie "Molecular Function" de GO (Gene Ontology). Nous avons distingué deux groupes dans la sous-famille *single* : le premier regroupe toutes les protéines avec un domaine Chromo en position C-terminale (groupe 1) alors que le second regroupe les protéines avec un domaine Chromo en position N-terminale (groupe 2). Les termes GO associés aux protéines des groupes 1 et 2 ont été collectés dans Uniprot (Release 14.1, Sep 2, 2008). Les termes GO spécifiques à chaque groupe ont été identifiés de la façon suivante : nous notons GO_fonction1 (respectivement GO_fonction2) l'ensemble des termes GO associés aux protéines du groupe 1 (respectivement groupe 2) qui apparaissent fréquemment dans le groupe 1 (respectivement groupe 2). Nous disons qu'une protéine vérifie un ensemble de termes GO si on peut lui associer au moins un terme GO de l'ensemble.

Les ensembles sont définis comme suit :

```
GO_function1 = {GO:0003723 (RNA-binding),
GO:0003964 (RNA-directed DNA polymerase activity),
GO:0004190 (aspartic-type endopeptidase activity)}

GO_function2 = {GO:0005524 (ATP-binding),
GO:0004386 (helicase activity),
GO:0005515 (protein binding),
GO:0018024 (histone-lysine N-methyltransferase activity),
```

```
GO:0003824 (catalytic activity)}
```

SNK a été utilisé pour découvrir des associations entre les architectures des domaines protéiques et les deux cibles GO_fonction1 et GO_fonction2. Nous nous sommes intéressés aux c-SNoKs de longueur supérieure ou égale à 2. La liste de tous ces c-SNoKs est donnée en annexe 5.6.2.

Nous présentons ci-dessous quelques règles qui illustrent une forte association entre la combinaison séquentielle de domaines et la fonction biologique exprimée sous la forme d'un ensemble de termes GO.

Ces règles confirment la pertinence fonctionnelle de distinguer deux groupes dans la sous-famille *single*, à l'aide d'une caractéristique partagée par l'architecture (la position du domaine Chromo). De plus, les règles montrent bien les "signatures" de ces groupes. Par exemple, Rb2, Rb7 et Rb8 présentent des architectures caractéristiques du groupe 2, alors que les autres règles listées ci-dessous sont caractéristiques du groupe 1. Elles montrent que des propriétés fonctionnelles peuvent être liée à l'architecture des protéines .

```
Rb2: Chromo,SNF2_N, ->  GO_function2, (0.04,1.00)
Rb3: zf-H2C2,Chromo, ->  GO_function1, (0.04,1.00)
Rb7: Chromo,Helicase_C, ->  GO_function2, (0.03,1.00)
Rb8: SNF2_N,Helicase_C, ->  GO_function2, (0.03,1.00)
Rb9: rve,Chromo, ->  GO_function1, (0.88,1.00)
Rb13: zf-CCHC,Chromo, ->  GO_function1, (0.60,1.00)
Rb17: RVT_1,Chromo, ->  GO_function1, (0.94,1.00)
Rb18: Retrotrans_gag,Chromo, ->  GO_function1, (0.70,1.00)
Rb20: RVP_2,Chromo, ->  GO_function1, (0.67,1.00)
```

The Reverse transcriptase (RNA-dependent DNA polymerase) : Après l'étude du domaine Chromo, nous avons exploré la combinaison de domaines la plus fréquente dans les règles impliquant le motif protéique Retrotransposon gag (Retrotrans_gag, PF03732), le domaine reverse transcriptase (RVT_1, PF00078) et le domaine integrase core (rve, PF00665) comme illustré dans les deux règles comprenant le domaine Retrotransposons gag :

```
R2: Retrotrans_gag,RVT_1,Chromo, -> single, (0.31,1.00)
R3: Retrotrans_gag,rve,Chromo, -> single, (0.30,1.00)
```

qui ont le plus grand support.

Guidé par ces deux règles, nous avons découpé la famille Reverse transcriptase de Pfam en deux sous-familles : RVT_1_fam, comprenant 499 architectures, et RVT_2_fam, comprenant 290 architectures.

Afin d'identifier des spécificités de chaque sous-famille, nous avons utilisé SNK en prenant les sous-familles "RVT_1_fam" et "RVT_2_fam' comme cibles (499+290 architectures). Les règles de longueur 3 suivantes ont été obtenues.

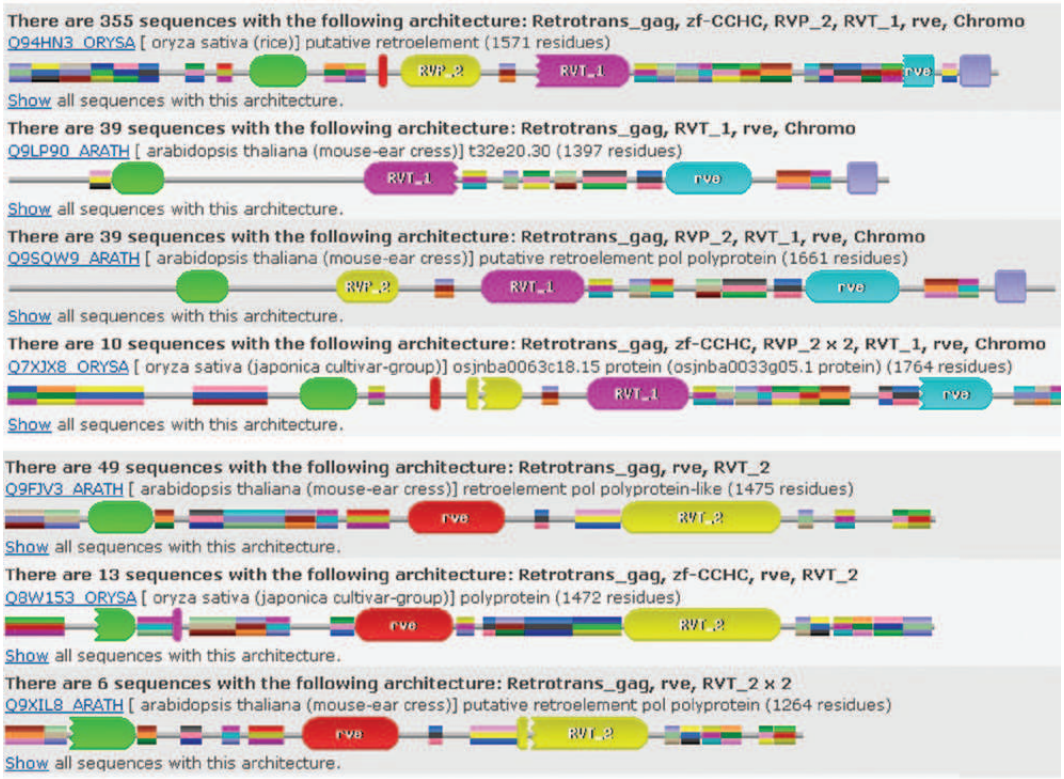


FIG. 3.4 – *Capture d'écran de Pfam pour la famille Chromo*

```
Rd1: Retrotrans_gag,rve,Chromo, ->  RVT_1_fam, (0.04,0.91)
Rd2: Retrotrans_gag,rve,RVT_2, ->  RVT_2_fam, (0.03,0.86)
```

Nous avons envoyé une requête à Pfam avec les domaines "Retrotrans_gag" ET "rve" ET "Chromo" d'une part et avec "Retrotrans_gag" ET "rve" ET "RVT_2" d'autre part. Nous avons obtenu deux ensembles distincts, mais homogènes d'architectures comme présentés dans la figure 3.4. Le rôle mutuellement exclusif joué par les domaines Chromo et RVT_2 dans la partie C-terminale des protéines choisies est très visible. De plus, la précédence en ordre inverse du domaine "rve" avec RVT_1 et RVT_2 (visible sur la figure, mais non décrit dans les commentaires de Pfam), est une autre caractéristique remarquable.

Pour mettre en lumière ce dernier point, nous avons considéré les domaines RVT_1 et RVT_2 comme un seul et même domaine RVT. Nous avons utilisé SNK pour détecter des corrélations entre l'architecture en domaines (et spécialement une inversion des domaines RVT et rve) et l'appartenance à RVT_1_fam et RVT_2_fam.
Avec un seuil de confiance égal à 0.8, un seuil de support à 0.02, et une longueur minimum de 1, 12 règles ont été obtenues. Deux d'entre elles sont particulièrement intéressantes :

```
Rc10: RVT,rve, -> RVT_1_fam, (0.70,0.99)
Rc12: rve,RVT, -> RVT_2_fam, (0.30,0.98)
```

Nous avons mis en évidence deux règles simples, mais caractéristiques où l'ordre des domaines est manifestement contraint et pourrait jouer un rôle fonctionnel.

Deuxième partie

Partie 2 : Mapping d'ontologies pour l'annotation fonctionnelle

Chapitre 4

Annotation et ontologies biologiques

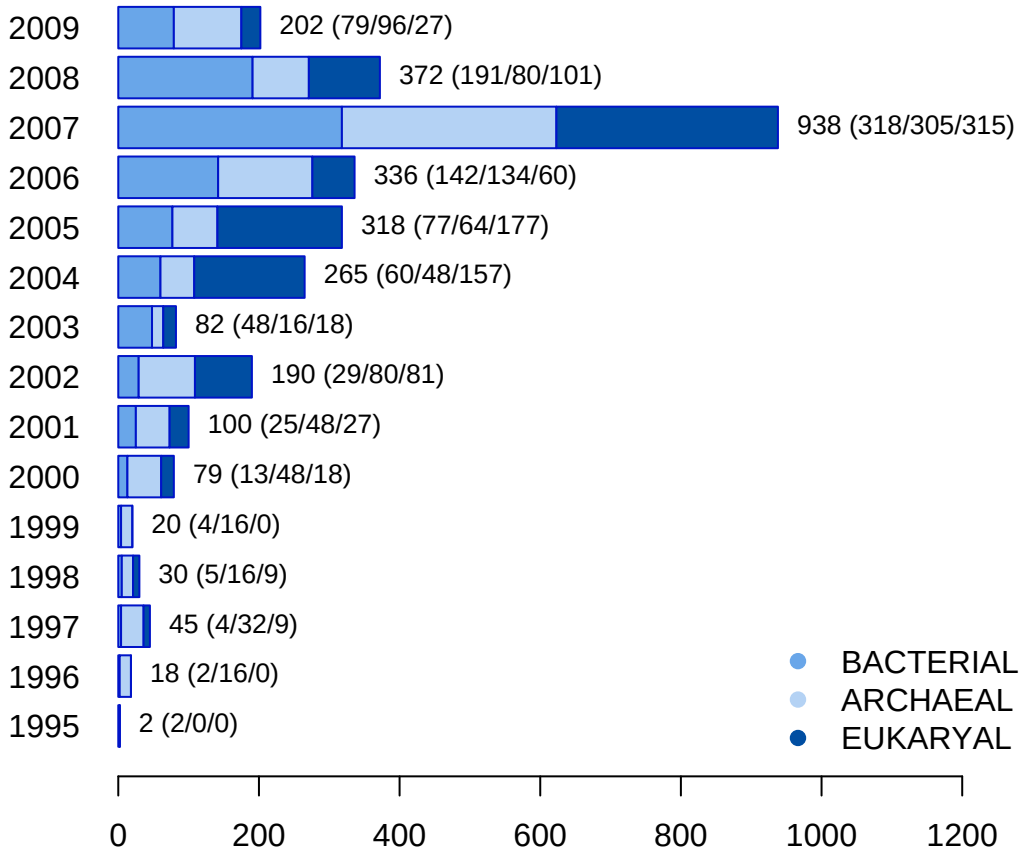
Sommaire

Une étape importante du post-séquençage de génome est l'annotation fonctionnelle (voir section 1.3). De plus en plus de données biologiques sont apportées par les scientifiques lors de cette étape et sont stockées dans des sources de données publiques ou privées (voir figure 4.1). Le problème d'un traitement et d'une analyse efficaces de ces données constitue l'un des goulots d'étranglement majeurs de la bioinformatique aujourd'hui [106].

4.1 Ontologies biologiques pour l'annotation de protéines

Lors de l'annotation, la fonction est souvent renseignée dans un champ textuel libre du système d'annotation. L'annotateur est alors complètement maître des termes qu'il utilise. La nécessité de recourir à des ontologies biologiques et des hiérarchies fonctionnelles résulte directement de cette

FIG. 4.1 – *Évolution du nombre de génomes séquencés*



possibilité pour l'annotateur de choisir les termes : un même terme peut avoir un sens légèrement différent pour des scientifiques et surtout une même idée peut être exprimée de nombreuses façons. Cela ne pose pas de problème important pour un expert humain, capable de comprendre ces subtilités, mais pour toute utilisation et analyse automatique, cela devient un handicap important. Par exemple, on pourra trouver *synthase* ou *synthetase* indifféremment, *metabolism of carbohydrates* pourra aussi s'exprimer par *synthesis and utilisation of carbohydrates*.

Le besoin d'utiliser un vocabulaire commun clairement défini est donc naturellement apparu. Un autre problème est la variabilité de la précision d'une annotation : un annotateur peut décrire une protéine comme étant impliquée dans le *Metabolism of tryptophan* et un autre décrira la protéine comme étant impliquée dans le *Metabolism of amino-acids*. Ces deux descriptions de la fonction sont justes, mais pas avec le même niveau de précision. *Metabolism of tryptophan* est plus précis que *Metabolism of amino-acids*. Il est nécessaire d'organiser hiérarchiquement les termes du vocabulaire contrôlé, en liant les termes, de sorte que les termes parents soient plus généraux que leurs descendants.

Dans la suite de ce chapitre, nous présenterons les ontologies qui sont des structures permettant de rendre compte de cette notion de généralisation et nous donnerons des exemples d'ontologies biologiques célèbres dans le domaine de la bioinformatique.

4.1.1 Ontologies biologiques et hiérarchies fonctionnelles

Ontologies Le concept d'ontologie est employé dans des domaines très différents, en philosophie, en linguistique ou en intelligence artificielle. En philosophie, Aristote définit l'ontologie comme la "Partie de la métaphysique qui étudie l'être en tant qu'être, étude des propriétés générales de ce qui existe". En informatique, une ontologie est plus un modèle conceptuel que l'étude de ce qui est. Ainsi Gruber [45] définit une ontologie comme "la spécification d'une conceptualisation" ("*Specification of a conceptualization*").

Une ontologie représente un point de vue spécifique d'une communauté sur des données. Cela peut expliquer l'existence de plusieurs ontologies dans des domaines proches, en particulier en biologie.

Les principaux composants d'une ontologie sont :
- des concepts (par exemple *cell wall biogenesis*)
- des relations entre ces concepts :
  - les relations assurant la structure hiérarchique de l'ontologie : *is-a* ou *part-of*, ...
  - les autres relations, appelées attributs ou rôles (par exemple *regulates*, *a-pour-fonction*)
- des axiomes (par exemple des restrictions de cardinalité : chaque protéine est associée à au moins une source)

Auxquels s'ajoutent :
- des contraintes (par exemple des restrictions de disjonction : une hélice ne peut pas être un feuillet et vice-versa)
- des instances (par exemple la protéine Q10287)
- des valeurs (la protéine identifiée par le code Q10287 est la *1,3-beta-glucan synthase component bgs1*)

Riichiro Mizoguchi [81] donne les idées fondamentales de ce qui constitue une classe (que nous appelons aussi **concept**), et une relation *is-a* :

TAB. 4.1 – *Données chiffrées sur quelques ontologies biologiques*

Ontologie	# concepts	prof. max.	# concepts au niveau 1	# feuilles	# desc. max. d'un concept
Subtilist	63	3	6	54	10
FunCat 2.1	1362	6	28	1022	25
MultiFun	653	5	10	591	185
Gene Ontology	27454	???	???	???	???

– **Une propriété intrinsèque**. La propriété intrinsèque d'une *chose* est une propriété qui est essentielle à la *chose*. La *chose* perd son identité si cette propriété change.
– **La définition ontologique d'une classe**. X est une classe si et seulement si chaque élément x de X satisfait la propriété intrinsèque de X. On dit alors que x est une **instance** de X.
– **Une relation *is-a***. La relation *classe A is-a classe B* n'est satisfaite que si et seulement si chaque instance de la classe A est aussi une instance de la classe B.

Les ontologies peuvent être classées en fonction des composants et des informations concernant ces composants [63]. Un type simple d'ontologie est un *vocabulaire structuré*, essentiellement composé d'une liste de concepts. Lorsque ces concepts sont organisés par des liens *is-a*, il s'agit d'une *taxonomie*. Dans un *thésaurus*, les concepts sont organisés sous la forme d'un graphe, les arcs du graphe représentent des relations comme la synonymie, la généralisation, ou la spécialisation d'un terme, l'équivalence de deux termes. Un exemple de thésaurus très célèbre est Wordnet [77]. Enfin une *base de connaissance* peut contenir tous les types de composants, et permet de plus de raisonner sur l'ontologie.

Hiérarchies fonctionnelles Parmi les ontologies, les **hiérarchies fonctionnelles** forment un groupe partageant des points communs. Il s'agit d'ontologies simples, le plus souvent structurées sous forme d'arbre, dont les concepts sont liés uniquement par des relations *is-a*, et sans contrainte d'aucune sorte. Les hiérarchies fonctionnelles sont dédiées à un domaine d'application bien délimité : l'annotation fonctionnelle de protéines.

Ces ontologies ont souvent été d'abord de simples vocabulaires contrôlés [63] avant d'être complétées et éventuellement organisées en hiérarchies.

Dans les vocabulaires contrôlés hiérarchisés que nous présenterons dans ce chapitre, lorsqu'une protéine est annotée par une fonction, elle est nécessairement annotée par les fonctions plus générales que cette dernière. C'est la règle dite du *true path*.

4.1.2 Exemples d'ontologies et de hiérarchies fonctionnelles

Comme le font remarquer Soldatova et King dans [106], la recherche sur la biologie et les ontologies trouve son origine dans l'Antiquité avec les travaux d'Artistote (en biologie notamment sur la classification des espèces). Il a fallu attendre 2400 ans pour que la bioinformatique les réunisse à nouveau.

Dans cette section, nous présenterons des exemples d'ontologies biologiques et de hiérarchies fonctionnelles communément utilisées en biologie et plus spécifiquement pour l'annotation fonctionnelle de génomes. Nous présenterons les hiérarchies par ordre chronologique de leur développement.

MultiFun

MultiFun [95] a été développée en 1998 par Monica Riley pour annoter le génome de *Escherichia coli K12*. Il s'agit d'une hiérarchie fonctionnelle conçue à l'origine pour décrire les produits des gènes par une seule fonction. Cette restriction s'est avérée trop limitante et MultiFun a été étendue pour permettre l'association d'un produit de gène à de multiples catégories. Enfin MultiFun intègre la classification EC (Enzyme Classification) et une version modifiée de la nomenclature TC (classification des protéines de transports).

Les bases de données EcoCyc [57] et MetaCyc [58] ont adopté MultiFun pour une partie de l'annotation des génomes qu'elles contiennent.

Les relations entre les concepts ne sont pas clairement définies dans la hiérarchie, mais sont de types *subclass-of* (ou *is-a*).

Cette hiérarchie fonctionnelle est l'une des premières à avoir été utilisée. De par sa construction, elle est dédiée à l'annotation de génomes bactériens. Elle est peu utilisée aujourd'hui, probablement parce qu'elle n'est pas liée à des systèmes d'aide à l'annotation (type plateforme d'annotation).

Subtilist et Subtilist modifiée

La première version de Subtilist [83] est utilisée dans la base éponyme pour décrire les fonctions des protéines de *Bacillus subtilis*. Une protéine peut être annotée par plusieurs concepts de Subtilist.

Dans le cadre de cette thèse, nous avons utilisé une version complétée de Subtilist employée par les annotateurs de l'INRA de Jouy-en-Josas. Cette version enrichie a été utilisée pour l'annotation manuelle de 3 génomes bactériens dans le cadre du projet AGMIAL (présenté dans la section 1.5) : *Lactobacillus sakei*, *Lactobacillus bulgaricus* et *Flavobacterium psycrophilum*.

Subtilist est de taille plus restreinte que les autres hiérarchies fonctionnelles puisqu'elle contient une soixantaine de concepts seulement alors que FunCat et MultiFun en contiennent de l'ordre d'un millier.

FunCat

La hiérarchie fonctionnelle FunCat [99] (Functional Catalogue), a été développée en 2004 par le *Munich Information Center for Protein Sequences*. Les caractéristiques succinctes de la hiérarchie sont présentées dans la table 4.1. La hiérarchie a évolué au fil des années, avec l'ajout et la disparition de nombreux concepts. La version actuelle 2.1 est publiée depuis janvier 2007.

FunCat a été utilisée pour annoter 4 génomes bactériens (*Listeria monocytogenes EGD*, *Listeria innocua Clip11262*, *Helicobacter pylori KE26695 (ATCC 700392)*, *Bacillus subtilis*, *Desulfotalea psychrophila*), 2 génomes de champignons (*Saccharomyces cerevisiae*, *Neurospora crassa*) et 1 génome eucaryote (*Mus musculus*), 1 génome d'archébactérie (*Thermoplasma acidophilum*). FunCat est aujourd'hui utilisée automatiquement pour attribuer les fonctions aux protéines de la base PEDANT [115] (dans le système du même nom).

La hiérarchie fonctionnelle FunCat est organisée sous la forme d'un arbre. Les relations d'héritage

entre les concepts ne sont pas clairement décrites, mais sont de type *subclass-of* (ou *is-a*). On peut néanmoins relever la présence d'une relation qui devrait être de type Part-of au sein de la hiérarchie, par exemple au niveau des concepts `70.10 nucleus` ou `70.16 mitochondrion` :

```
70.10 nucleus
  70.10.03 chromosome
  70.10.04 centromere / kinetochore
  70.10.05 nuclear membrane
  70.10.06 nuclear matrix
  70.10.07 nucleolus
  70.10.09 nuclear speckles
70.16 mitochondrion
  70.16.01 mitochondrial outer membrane
  70.16.03 mitochondrial intermembrane space
  70.16.05 mitochondrial inner membrane
  70.16.07 mitochondrial matrix
```

Les descendants de 70.10 et de 70.16 sont liés à leurs ancêtres par une relation *part-of* et non *is-a*, car la membrane externe de la mitochondrie n'est pas une mitochondrie, mais appartient bien à la mitonchodrie. De la même façon, un chromosome n'est pas un noyau, mais fait partie du noyau.

FunCat se différencie des classifications déjà existantes (par exemple la classification EC, voir section 4.1.2) dans le sens où elle est basée sur la voie métabolique dans laquelle l'enzyme est impliquée, alors que la classification EC classe les enzymes en fonction des mécanismes chimiques sous-jacents. Les auteurs de FunCat comparent leur hiérarchie à la Gene Ontology. Selon eux la force de FunCat réside d'une part dans sa simplicité et d'autre part dans le caractère intuitif et la compréhension aisée de sa structure d'arbre par rapport à une structure de DAG plus complexe conduisant plus facilement à une utilisation erronée des termes de la Gene Ontology par les annotateurs.

Observations sur les hiérarchies fonctionnelles

Avant de poursuivre avec un exemple d'ontologie biologique, nous soulignons quelques problèmes qu'il est possible de rencontrer dans les hiérarchies fonctionnelles, et qui resteront valables pour plusieurs autres classifications biologiques telles que les EC Numbers ou les TC Numbers.

Sur le nommage L'utilisateur de l'ontologie attend un nom clair, concis dans la mesure du possible et compréhensible indépendamment des autres noeuds de l'ontologie. Ce n'est souvent pas le cas, le nom de certains concepts n'étant alors qu'une précision du nom du concept dont ils descendent. D'autre part, le nom de concept des hiérarchies fonctionnelles mélange deux voire trois propriétés du concept.

- Le nom à proprement parler que nous appellerons **label**.
- Une description du concept, qui est souvent une phrase de quelques mots ou un court paragraphe. Par exemple dans MultiFun : `1.7.33.1` est accompagné de la phrase `Salvage pathways of adenine, hypoxanthine, and their nucleosides`.

– Des exemples ou des énumérations figurent dans les noms des concepts. Dans FunCat `14.07.01 modification with fatty acids` est illustré par des exemples `(e.g. myristylation, palmitylation,farnesylation)`.

Ces problèmes rendent plus complexes les étapes de comparaison des noms des concepts.

Sur la structure Les hiérarchies fonctionnelles, comme un certain nombre de classifications biologiques, sont organisées sous forme d'arbres pour des raisons de simplicité d'utilisation et de compréhension : l'héritage multiple de propriétés venant de plusieurs parents est évitée. Cette structure d'arbre n'est pas en cause, mais les relations qui lient les concepts sont à établir clairement. Or dans les hiérarchies fonctionnelles, ce n'est pas le cas. L'utilisateur peut deviner qu'il s'agit de relation de type *Subclass-of* ou *is-a*, mais la situation est un peu plus complexe et il est par exemple possible d'identifier des relations de types *part-of* (voir le paragraphe 4.1.2 décrivant FunCat).

Sur le format Nous n'avons pas encore évoqué l'accessibilité en ligne sur internet et les formats de stockage de ces hiérarchies, mais il s'agit là aussi d'un problème. Hormis FunCat qui est disponible en XML, les hiérarchies et les classifications EC et TC (organisées aussi sous forme d'arbre voir section 4.1.2) sont présentées dans des formats variables, le plus souvent sous forme de texte libre organisé par des sauts de ligne et des tabulations. De plus, l'ontologie elle-même est parfois difficile à trouver en ligne indépendamment des données qu'elle annote.

Gene Ontology

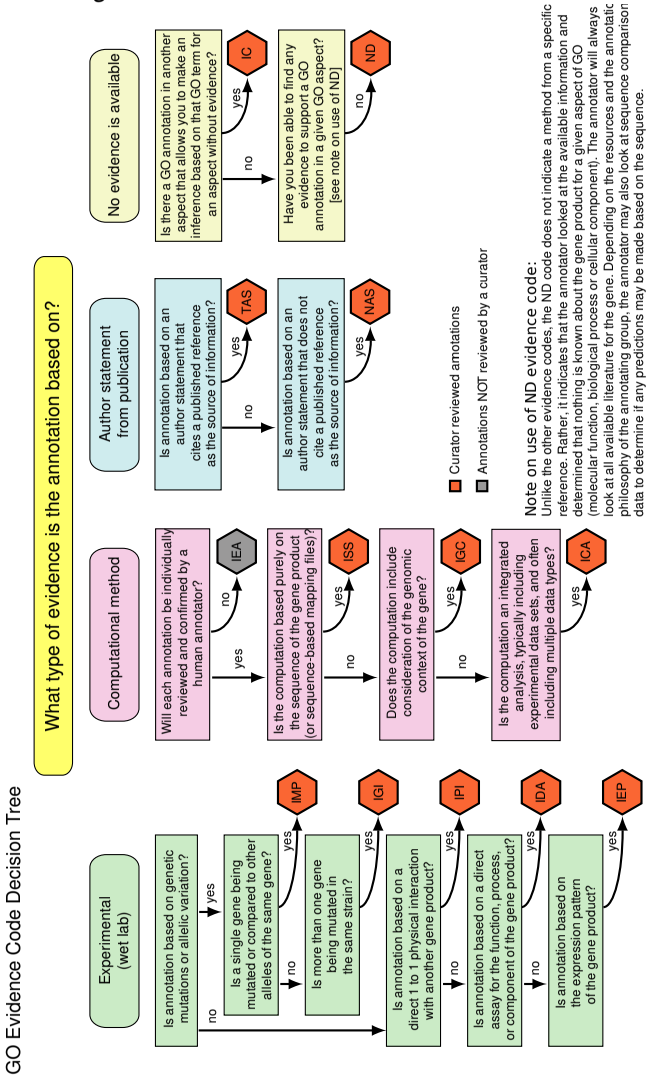
En même temps que les hiérarchies fonctionnelles, d'autres structures ont été proposées pour décrire les fonctions des protéines. C'est le cas en particulier de Gene Ontology [111] (GO). Originellement, GO a été conçue pour permettre les comparaisons des annotations d'organismes multicellulaires (*Caenorhabditis elegans* – un ver, *Drosophila melanogaster* – la mouche du vinaigre, *Mus musculus* – la souris, et d'autres).

GO est composée de 3 sous-ontologies : la première traite de la fonction moléculaire du produit du gène (*Molecular Function*), la seconde des processus biologiques dans lesquels le produit du gène est impliqué (*Biological Process*), et enfin la troisième et dernière sous-ontologie permet de décrire la localisation cellulaire du produit du gène (*Cellular Component*). Les ontologies sont structurées à l'aide de deux relations *is-a* et *part-of*, sous la forme d'un DAG (*Directed Acyclic Graph*). GO n'est pas statique et évolue en fonction des demandes des utilisateurs, si bien que des termes sont ajoutés ou supprimés en permanence.

A chaque terme GO (chaque concept de l'ontologie) sont associés un certain nombre d'attributs :
– Un nom qui peut être un mot ou une courte phrase.
– Un identifiant alphanumérique unique.
– Une définition composée d'un texte qui peut citer des références.
– Un espace de nommage (*namespace*) définissant la sous-ontologie à laquelle le terme appartient.

Les termes peuvent aussi avoir des "synonymes". En réalité le terme synonyme est mal choisi, puisqu'il peut s'agir aussi bien de termes proches ayant un sens un peu plus général, ou un peu plus précis, ou simplement proches sémantiquement, que de synonymes au sens classique du terme.

FIG. 4.2 – *Organisation des evidence code dans la Gene Ontology*



Les termes sont censés être neutres vis-à-vis des espèces, mais certains termes peuvent être espèces-spécifiques (*sensu*). Ils auront alors un sens différent selon l'espèce à laquelle le terme sera appliqué. Il est à noter que l'utilisation des termes *sensu* a tendance à s'effacer au fur et à mesure des mises à jour de GO.

Lors de l'annotation d'une séquence avec un terme GO, un tag va renseigner sur l'origine de l'annotation : inféré électroniquement par alignement de séquence, révélé par expérience de biologie humide... Cette méta-annotation est appelée *evidence code* (voir figure 4.2).

La Gene Ontology est très fréquemment utilisée aujourd'hui. AmiGO[1] recence 249595 produits de gènes annotés avec des termes GO de façon manuelle ou automatique au 2 juillet 2009.

Bien que les 3 sous-ontologies ne soient pas liées a priori, des recherches ont essayé d'établir des relations entre celles-ci [16] et ont permis d'enrichir les 3 sous-ontologies de relations inter-

[1] http://amigo.geneontology.org/

ontologies.

Mots clés UniProt Uniprot [13] est l'une des banques de données biologiques les plus célèbres. Elle est composée de deux sources de données complémentaires : SwissProt et TrEMBL qui contiennent des informations sur des protéines. Ces informations proviennent d'annotateurs humains ou d'annotations automatiques validées par des annotateurs pour SwissProt et d'un système d'annotation automatique pour TrEMBL [69]. Les deux banques utilisent entre autres un vocabulaire commun pour l'annotation des protéines : les mots clés UniProtKB/Swiss-Prot, qui sont classés en 10 grandes catégories :
– *Biological process*
– *Cellular component*
– *Coding sequence diversity*
– *Developmental stage*
– *Disease*
– *Domain*
– *Ligand*
– *Molecular function*
– *PTM*
– *Technical term*
Ils sont décrits dans un formalisme interne à UniProt, et organisés sous la forme d'un DAG.

Autres vocabulaires contrôlés

Les ontologies que nous présentons maintenant sont issues d'ontologies tournées vers le domaine biomédical (MeSH, UMLS), ou elles ne peuvent pas être utilisées seules pour l'annotation de protéines (EC numbers, TC numbers) parce qu'elles sont dédiées à la description d'un domaine plus spécialisé que les hiérarchies fonctionnelles précédentes.

Enzyme Commission number Plus couramment trouvée sous l'acronyme **EC number** [5], cette hiérarchie organise les enzymes en fonction des réactions chimiques qu'elles mettent en oeuvre. Les débuts de cette classification datent de 1956, et sont le fruit du travail d'une commission internationale. Cette classification fait autorité aujourd'hui, et est quotidiennement utilisée pour la description des enzymes.

Les EC numbers ne peuvent toutefois pas remplacer les hiérarchies fonctionnelles : parmi les protéines, la classification ne permet de décrire que les enzymes, et le point de vue sur les enzymes est strictement chimique et non fonctionnel. La classification indique le type de réaction, mais pas du tout le lieu, le substrat, etc.

Par ailleurs, certains éléments de la classification ne sont pas à ce jour associés à des enzymes [67], et certains concepts ne semblent avoir que très peu de protéines associées.

Enfin, comme dans beaucoup de classifications biologiques, les noms des concepts sont peu appropriés à un traitement automatique. Par exemple, les noms des trois premiers concepts sont :

```
1. Oxidoreductases;
```

```
1.1  Acting on the CH-OH group of donors;
1.1.1  With NAD+ or NADP+ as acceptor;
```

Si bien que le nom du concept 1.1.1 n'a de sens que si ses deux ancêtres sont connus.

Transporter Classification database TCDB [101], est parfois trouvé sous le nom de TC number. Il s'agit d'une classification des protéines de transport, incluant les canaux ioniques. Sa structure est très similaire à celle de la classification EC. Les mêmes réserves peuvent être formulées sur les choix de nommage des concepts. En pratique, la TCDB est moins répandue que les EC numbers.

Ces deux classifications (EC Numbers et TC Numbers) sont des recommandations de l'*International Union of Biochemistry and Molecular Biology* [2].

KEGG Ontology KEGG [56] présente plusieurs systèmes de classification, parmi lesquels on trouve KEGG BRITE qui est une collection de classifications hiérarchiques représentant les connaissances sur des aspects variés des systèmes biologiques. KEGG BRITE contient différents types de relations, tels que des mapping de données génomiques ou moléculaires vers KEGG BRITE via l'utilisation des identifiants KEGG Orthology, etc.

KEGG BRITE n'est pas disponible au téléchargement sur le site de KEGG [1], et semble mélanger concepts, relations et propriétés.

UMLS Le Unified Medical Language System® (UMLS) [70, 15] est un compendium qui regroupe des informations biomédicales issues de plusieurs sources. L'UMLS est composé de 3 sources de connaissances :

- Un métathésaurus [3] (*metathesaurus*®) qui est une base de données de très grande taille, multi-usages et multi-langues qui contient des informations sur les concepts biomédicaux ou liés à la santé (établissement de factures hospitalières par exemple), leurs noms et les relations qui les lient. Le métathésaurus est construit à partir des versions électroniques de différents thesauri, classifications, d'ensembles de codes ou de listes de vocabulaires contrôlés utilisés aussi bien pour les soins au patient que les services de facturation. Le métathesaurus permet d'établir de nouvelles relations entre les termes issus de ces différents vocabulaires.
- Un réseau sémantique [4] qui consiste en (1) un ensemble de catégories, ou Types Sémantiques (*Semantic Types*), qui fournissent une catégorisation à tous les concepts du métathésaurus, et (2) un ensemble de relations, ou Relations Sémantiques (*Semantic Relations*), qui existent entre les types sémantiques.
- Le lexique SPECIALIST [2] qui a été développé pour fournir les informations nécessaires au *SPECIALIST Natural Language Processing (NLP) System*. C'est un lexique contenant des termes d'anglais courant aussi bien que des termes biomédicaux. Pour chaque terme, il spécifie les informations syntaxiques, morphologiques et orthographiques.

MeSH MeSH® (Medical Subject Headings) est un vocabulaire contrôlé conçu par l'*American National Library of Medicine* et utilisé pour indexer, cataloguer et rechercher dans des documents traitant de sujets biomédicaux ou liés à la santé. Les termes sont organisés sous la forme d'une

[2] http://www.chem.qmul.ac.uk/iubmb/nomenclature/

hiérarchie, et peuvent être regroupés sous plusieurs catégories telles que : anatomie (*anatomy*), organismes (*organisms*), maladies (*diseases*), dont la plupart peuvent se retrouver dans l'OBO Foundry (voir paragraphe 4.2.2). MeSH regroupe plus de 15000 termes.

MeSH décrit les relations entre les termes par des liens *is-a* qui représentent à la fois les relations effectivement *is-a* et *part-of*

Ontologies OBO, ontologies générales L'OBO Foundry [105] contient plus de 60 ontologies dans des domaines variés (voir la section 4.2.2). Parmi celles-ci, on trouve : la *sequence ontology* [33], le *Mouse adult gross anatomy ontology*, la *Foundational Model of Anatomy* [98]...

Par ailleurs, les domaines de la bioinformatique, et du biomédical peuvent parfois aussi utiliser des ontologies moins spécialisées dans le domaine bio-médical. Le thesaurus d'anglais général WordNet [77] est ainsi souvent utilisé.

4.2 Formats de représentation des ontologies biologiques

4.2.1 Des fichiers plats à OWL

De même que la complexité des ontologies biologiques varie, leurs formats de représentation sont multiples. Les hiérarchies fonctionnelles sont souvent disponibles sous la forme de fichiers plats. Ces fichiers comportent un concept par ligne. Les relations entre les concepts ne sont pas du tout identifiées, mais les relations *is-a* sont détectables dans l'identifiant du concept. Il est facile de déduire que le concept 1.1 est le descendant du concept 1 et qu'une relation *is-a* les lie.

FunCat [99] est disponible au format XML ; les concepts sont modélisés sous la forme de catégories qui contiennent éventuellement d'autres catégories. Les relations de parenté entre les concepts sont figurées par la structure de l'arbre XML.

Les ontologies plus complexes telles que la Gene Ontology, sont, elles, représentées dans un formalisme spécialement conçu pour exploiter la richesse d'une ontologie : le langage OWL (*Ontology Web Language*). Ce langage, recommandation du W3C, permet aussi bien d'exprimer des concepts, que de définir des relations entre concepts ou des concepts complexes à partir de concepts plus simples. Il permet également d'exprimer toutes les spécificités des ontologies.

Quels que soient leurs formats initiaux de représentation, les hiérarchies fonctionnelles et les ontologies biologiques peuvent s'exprimer dans le format OWL. Dans le cadre de cette thèse, nous avons converti les hiérarchies SubtiList, MultiFun et FunCat au format OWL, ce qui permettra l'utilisation de raisonneur pour la détection d'incohérences dans les ontologies par exemple.

L'utilisation de fichiers au format OWL, ainsi que leur compréhension ne sont malheureusement pas intuitives. Pour permettre une popularisation des ontologies biomédicales et biologiques, le format OBO *Open Biomedical Ontologies* a été proposé dans le cadre de l'*OBO Foundry* [105].

4.2.2 OBO

L'*OBO Foundry* [105] est un projet collaboratif impliquant des développeurs d'ontologies biologiques ou biomédicales. Le but de l'*OBO Foundry* est d'établir un ensemble de principes pour le

développement d'ontologies avec pour but de créer un ensemble d'ontologies de référence dans le domaine biomédical.

Les principes de l'*OBO Foundry*[3] sont décrits de la façon suivante :

1. **The ontology must be *open* and available to be used by all without any constraint other than (a) its origin must be acknowledged and (b) it is not to be altered and subsequently redistributed under the original name or with the same identifiers.**

   The OBO ontologies are for sharing and are resources for the entire community. For this reason, they must be available to all without any constraint or license on their use or redistribution. However, it is proper that their original source is always credited and that after any external alterations, they must never be redistributed under the same name or with the same identifiers.

2. **The ontology is in, or can be expressed in, a *common shared syntax*. This may be either the OBO syntax, extensions of this syntax, or OWL.**

   The reason for this is that the same tools can then be usefully applied. This facilitates shared software implementations. This criterion is not met in all of the ontologies currently listed, but we are working with the ontology developers to have them available in a common OBO syntax.

3. **The ontologies possesses a *unique identifier space* within the OBO Foundry.**

   The source of a term (i.e. class) from any ontology can be immediately identified by the prefix of the identifier of each term. It is, therefore, important that this prefix be unique.

4. **The ontology provider has procedures for identifying distinct successive *versions*.**

5. **The ontology has a clearly specified and clearly *delineated content*.**

   The ontology must be orthogonal to other ontologies already lodged within OBO.

   The major reason for this principle is to allow two different ontologies, for example anatomy and process, to be combined through additional relationships. These relationships could then be used to constrain when terms could be jointly applied to describe complementary (but distinguishable) perspectives on the same biological or medical entity.

   As a corollary to this, we would strive for community acceptance of a single ontology for one domain, rather than encouraging rivalry between ontologies.

6. **The ontologies include textual *definitions* for all terms.**

   Many biological and medical terms may be ambiguous, so terms should be defined so that their precise meaning within the context of a particular ontology is clear to a human reader.

7. **The ontology uses relations which are unambiguously defined following the pattern of definitions laid down in the *OBO Relation Ontology*.**

8. **The ontology is *well documented*.**

9. **The ontology has a plurality of independent *users*.**

10. **The ontology will be developed *collaboratively* with other OBO Foundry members.**

[3] http://www.obofoundry.org/crit.shtml

Ces principes permettent d'éviter un écueil dû à la popularité des ontologies en biologie. Paradoxalement, alors que l'un des rôles d'une ontologie est de faciliter une meilleure compréhension des données en proposant un vocabulaire partagé, et l'interopérabilité entre les différentes sources de données, le développement d'un grand nombre d'ontologies sur des domaines proches nuit à cet objectif. Pour assurer ce rôle, il faut être capable d'identifier des concepts équivalents, de trouver des correspondances entre les relations, etc. Les principes de l'*OBO Foundry* assurent que les relations utilisées seront compatibles entre les ontologies, que le nommage des concepts sera cohérent avec le reste des concepts des ontologies d'OBO, et que l'ensemble des ontologies sera exprimé selon un même formalisme en OWL ou au format OBO. D'autre part, le premier principe précise que les ontologies doivent être ouvertes, la communauté des utilisateurs potentiels pourra donc compléter une ontologie existante ou l'adapter à ses propre besoins, sans être obligée de créer une nouvelle ontologie qui pourrait s'avérer redondante ou en tout cas avec un large recouvrement.

L'*OBO Foundry* organise les ontologies selon les domaines d'applications qui sont : l'anatomie, la biochimie, la fonction biologique, les processus biologiques, les séquences biologiques, l'environnement, les expériences, la santé, le phénotype et les protéines.

Les ontologies y sont disponibles dans un format OBO. Ce format a pour but d'être lisible par un être humain, facile à traiter informatiquement, extensible, et redondant au minimum. Ci-dessous, nous présentons un exemple de terme de la Gene Ontology, au format OBO :

```
[Term]
id: GO:0000010
name: trans-hexaprenyltranstransferase activity
namespace: molecular_function
def: "Catalysis of the reaction: all-trans-hexaprenyl diphosphate +
    isopentenyl diphosphate = diphosphate + all-trans-heptaprenyl
    diphosphate." [EC:2.5.1.30]
subset: gosubset_prok
synonym: "all-trans-heptaprenyl-diphosphate synthase activity"
EXACT [EC:2.5.1.30]
synonym: "all-trans-hexaprenyl-diphosphate:isopentenyl-diphosphate
    hexaprenyltranstransferase activity" EXACT [EC:2.5.1.30]
synonym: "heptaprenyl diphosphate synthase activity" EXACT [EC:2.5.1.30]
synonym: "heptaprenyl pyrophosphate synthase activity" EXACT [EC:2.5.1.30]
synonym: "heptaprenyl pyrophosphate synthetase activity" EXACT [EC:2.5.1.30]
xref: EC:2.5.1.30
xref: MetaCyc:TRANS-HEXAPRENYLTRANSTRANSFERASE-RXN
is_a: GO:0016765 ! transferase activity, transferring alkyl or aryl
    (other than methyl) groups
```

Le format OBO souffre cependant de certains inconvénients. En particulier, il manque d'une définition formelle [44, 9, 82, 85]. Par exemple le tag *relationship* est défini de la façon suivante : *This tag describes a typed relationship between this term and another term. [...] The* `necesseraly` *modifier allows a relationship to be marked as "not necessarily true"*
La notion de relation "*non nécessairement vraie*" n'est pas clairement définie, pas plus que la notion de *relation*. Par exemple considérons le terme :

```
[term]
id: GO:0001555
name: oocyte growth
is_a: GO:0016049 ! cell growth
relationship: part_of GO:0048601 ! oocyte morphogenesis
```

Il peut être interprété de plusieurs façons :
- chaque instance du terme `GO:0001555` doit avoir au moins une relation *parf-of* avec une instance du terme `GO:0048601`. Formellement : `GO:0001555` $\sqsubseteq \exists$ `part_of GO:0048601`
- une instance de `GO:0001555` peut être connectée via une relation *parf-of* seulement à des instances de `GO:0048601`. Formellement `GO:0001555` $\sqsubseteq \forall$ `part_of GO:0048601`

Un ordinateur n'est pas capable de déterminer le sens d'un terme, ce qui constitue un problème pour les raisonneurs automatiques [85]. Ce manque de définition formelle fait reposer les tâches de maintien de l'intégrité de l'ontologie entièrement sur les épaules des développeurs de l'ontologie. Pour permettre l'utilisation de raisonneurs et utiliser la puissance du langage OWL, des traducteurs de OBO vers OWL ont été proposés [82] permettant d'utiliser OWL-DL, qui est une logique avec une sémantique formellement définie.

4.3 Alignements d'ontologies

Dans les faits, et malgré les initiatives telles que l'*OBO Foundry*, il existe des ontologies portant sur le même domaine d'application. En particulier dans le domaine de l'annotation fonctionnelle, différentes ontologies représentent un point de vue différent sur les données, mais expriment pour la plupart des idées communes.

Afin de pouvoir rapatrier, traiter, et utiliser de façon homogène l'ensemble des données annotées par les ontologies relatives à un même domaine, il est nécessaire de pouvoir établir des correspondances entre les termes de ces ontologies. L'identification des concepts équivalents, ou de relations communes est alors une tâche incontournable, c'est l'un des buts du mapping.

4.3.1 Définition du mapping

Nous nous replaçons dans le cadre défini par [34] et reprenons quelques définitions :

Matching Le matching est le processus qui consiste à identifier les correspondances ou les relations entre les entités de différentes ontologies.

Alignement Un alignement est l'ensemble des correspondances entre deux ontologies ou plus. Un alignement est la sortie du processus de matching.

Correspondance Une correspondance est une relation qui tient ou est supposée tenir, étant donné un algorithme de matching particulier ou un oracle, entre les entités d'une ontologie.

Mapping La mapping est la version orientée de l'alignement.

Dans notre cas, nous ne considérons pas tous les composants de l'ontologie, mais seulement l'ensemble des concepts des ontologies. Nous donnerons une définition adaptée à notre problème dans le chapitre suivant.

Formellement, nous pouvons définir le processus de matching comme suit :

Définition 4.3.1 (Processus de matching) [34] Le processus de matching peut être vu comme une fonction qui prend deux ontologies O_1 et O_2, un alignement d'entrée A, un ensemble de paramètres p et un ensemble de ressources et d'oracles r, retourne l'alignement A'

$$A' = f(O_1, O_2, A, p, r)$$

Nous cherchons un ensemble de correspondances qui lient les concepts de O_1 et O_2 par une relation ρ de $\mathcal{R}$ qui comprend deux relations $\equiv$ (équivalence), et $\leq$ (subsomption.)

Définition 4.3.2 (Correspondance) Une correspondance est un 4-uplet (c, d, ρ, s) où c et d sont des concepts de O_1 et O_2 respectivement, $\rho \in \mathcal{R}$ une relation et s un score dénotant la qualité de la correspondance.

Nous pourrons évoquer trois types d'alignements en fonction du nombre de concepts impliqués dans une correspondance. L'alignement peut être total si tous les concepts de l'ontologie source sont mis en correspondance avec un (des) concept(s) de l'ontologie cible, et partiel sinon.
- L'alignement 1–1 met en correspondance exactement un concept de l'ontologie O_2 avec un concept de l'ontologie O_1 si l'alignement est total et au moins un concept de l'ontologie O_2 sinon.
- L'alignement 1–n (*one-to-many*) met en correspondance à un concept de l'ontologie O_1 des concepts de l'ontologie O_2 si l'alignement est total et au moins 1 concept de O_2 si l'alignement est partiel.
- L'alignement n–m (*many-to-many*) met en correspondance un ensemble de concepts de O_1 avec un ensemble de concepts de O_2.

4.4 Méthodes d'alignement d'ontologies

Dans cette section nous allons présenter différentes méthodes qui permettent d'établir des relations d'équivalence entre des éléments de ces ontologies. Dans leur ouvrage "Ontology Matching", Euzenat et Shvaiko [34] présentent de façon approfondie l'ensemble des techniques classiques mises en oeuvre. Nous n'allons ici présenter que les grands groupes de techniques.

Dans les alignements d'ontologies l'équivalence entre les concepts est obtenue par des **matchers**, qui calculent des similarités entre des propriétés spécifiques des concepts des ontologies.

Nous allons donc maintenant présenter diverses techniques pour détecter des similarités entre concepts.

4.4.1 Similarité entre chaînes de caractères ou textes

Les matchers les plus utilisés et probablement les plus efficaces sont basés sur la comparaison de textes ou de chaînes de caractères.

Ces matchers nécessitent souvent une normalisation consistant en une uniformisation de la forme des éléments à comparer : normalisation de la casse, des accents, des éléments de ponctuation... La suppression des mots vides qui n'apportent pas de sens à l'expression (tels que "of", "the"...) et la lemmatisation (suppression du préfixe et du suffixe d'un mot), peuvent aussi être des étapes de la normalisation. Dans les cas des ontologies biologiques et biomédicales, certaines de ces étapes sont à considérer prudemment. Par exemple, un préfixe peut changer radicalement le sens d'un terme : `Hydrogenase` et `Dehydrogenase` sont deux notions vraiment différentes, mais qui ne diffèrent que par leur préfixe.

Les mesures de similarités qui peuvent être utilisées sont nombreuses. Nous détaillerons ici les distances élémentaires largement utilisées pour l'alignement d'ontologies.

Egalité de chaînes de caractères Cette méthode rudimentaire attribue une similarité de 1 en cas d'égalité stricte entre les chaînes de caractères, sinon 0. Elle est souvent utilisée pour établir des "ancres" (voir section 4.5).

Distance de Hamming Intuitivement, la distance de Hamming entre deux chaînes de caractères est le nombre de caractères qui diffèrent entre elles. Cette valeur est normalisée par la longueur de la plus grande des deux chaînes.

Définition 4.4.1 (Distance de Hamming) Soient s et t deux chaînes. $\#s$ (resp. $\#t$) dénote la longueur de la chaîne s (resp. t), $s[i]$ (resp. $t[i]$) dénote le i^e caractère de la chaîne s (resp. t).

$$\sigma_{Hamming}(s,t) = \frac{(\sum_{i=1}^{\min(|s|,|t|)} s[i] \neq t[i]) + |\#s - \#t|}{\max(\#s, \#t)}$$

n-gram Un n-gram est une sous-chaîne de longueur n. Pour mesurer la similarité entre deux chaînes, on identifie dans les deux chaînes tous les n-grams contenus. La similarité des deux chaînes est alors le nombre de n-grams partagés. En pratique cette similarité est normalisée.

Définition 4.4.2 (Similarité n-gram) Soit $\text{ngram}(s,n)$ l'ensemble des sous-chaînes de s de longueur n. La similarité n-gram entre deux chaînes est donnée par :

$$\sigma_{n-gram}(s,t) = \frac{|\text{ngram}(s,n) \cap \text{ngram}(t,n)|}{\min(\#s, \#t) - n + 1}$$

Distance de Levenshtein Aussi appelée distance d'édition, la distance de Levenshtein [68] entre deux chaînes de caractères est le nombre minimum d'opérations d'édition (ajout, suppression, remplacement) nécessaires pour transformer une chaîne en l'autre.

Needleman et Wunsch La distance établie à partir de l'algorithme de Needleman et Wunsch [86] est très similaire à la distance de Levenshtein. La différence réside dans le fait que l'ajout ou la suppression d'un caractère ajoute au coût de l'opération une variable d'ajustement. En bioinformatique, l'algorithme de Needleman et Wunsch a longtemps servi à réaliser des alignements globaux de séquence.

Autres mesures de similarité Nous ajouterons à cette liste non exhaustive les mesures de similarité entre séquences biologiques qui peuvent être calculées à partir d'algorithmes tels que celui de Smith et Waterman, ou BLAST.

Ces mesures de similarité peuvent aussi bien être utilisées entre les éléments du schéma de l'ontologie (nom de concepts), qu'entre les instances de l'ontologie.

Il existe des implémentations libres en Java de la plupart de ces mesures de similarités : Simetrics, SecondString, Alignement API, SimPack.

TF-IDF Il ne s'agit pas d'une mesure de similarité entre chaînes de caractères, mais d'une méthode de pondération fréquemment utilisée pour évaluer l'importance d'un mot par rapport à un document au sein d'un corpus. Certaines techniques nécessitent de connaître l'ensemble des éléments à comparer pour prendre en compte des éléments généraux. La mesure TF-IDF utilise entre autres la fréquence d'apparition d'un mot dans l'ensemble des documents, permettant ainsi de donner aux mots une importance plus ou moins forte.

La valeur du TF-IDF pour un terme t_i donné, dans un document d_j donné par $TF\_IDF_{i,j}$ est évaluée par :

$$TF\_IDF_{i,j} = tf_{i,j} * idf_i$$

Soit un document d_j et un terme t_i. La fréquence du terme t_i dans d_j est :

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

où $n_{i,j}$ est le nombre d'occurrences du terme t_i dans d_j et $\sum_k n_{k,j}$ représente le nombre d'occurrences de tous les termes t_k dans le document d_j.

La fréquence inverse du terme t_i dans l'ensemble D des documents est :

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

où $|D|$ est le nombre total de documents du corpus et $|\{d_j \in D \text{ tel que } t_i \in d_j\}|$ est le nombre de documents de D qui contiennent au moins une fois le terme t_i.

4.4.2 Similarité utilisant la structure des ontologies

Les mesures de similarité utilisant la structure des ontologies sont nombreuses. Elles sont malheureusement souvent difficiles à utiliser dans les cas des ontologies biologiques car celles-ci ont une structure simple (voir section 4.1.2). Ces mesures de similarités peuvent utiliser des informations données telles que les types de données, les cardinalités, les relations liant les concepts, les clés...

4.4.3 Similarité utilisant les instances

Aussi nommées techniques extensionnelles dans [34], ces techniques peuvent être utilisées pour les ontologies biologiques puisqu'on dispose des protéines annotées par des termes des ontologies. Les mesures de similarité basées sur les instances sont variées :

Instances partagées Dans les cas les plus simples, des instances peuvent être annotées par des concepts de deux ontologies. Il est alors possible de définir une mesure de similarité basée sur le nombre d'instances partagées entre les concepts C et D. La mesure de Jaccard évalue le rapport de la taille de l'intersection entre les ensembles d'instances annotés par C et D sur la taille de l'union de ces ensembles.

Définition 4.4.3 (Extension) L'extension d'un concept C, notée $\text{ext}(C)$ est l'ensemble des instances annotées par ce concept.

Définition 4.4.4 (Similarité de Jaccard) Soit C et D deux concepts, la mesure de Jaccard noté $\sigma_{Jaccard}(C, D)$ est égale à :

$$\sigma_{Jaccard}(C, D) = \frac{|\,\text{ext}(C) \cap \text{ext}(D)|}{|\,\text{ext}(C) \cup \text{ext}(D)|}$$

D'autres mesures permettent de quantifier la similarité entre concepts. En se basant sur les instances, Kirsten *et al.* [59] proposent d'utiliser la similarité *Kappa* permettant l'utilisation d'un test statistique.

Définition 4.4.5 (Similarité Kappa) Soit C (resp. D) un concept de O_1 (resp. O_2). Soit Ω l'ensemble de toutes les instances annotées par des concepts des ontologies O_1 et O_2. Définissons :
$N_{11} = |\,\text{ext}(C) \cap \text{ext}(D)|$
$N_{12} = |\,\text{ext}(C) \setminus \text{ext}(D)|$
$N_{21} = |\,\text{ext}(D) \setminus \text{ext}(C)|$
$N_{22} = |\Omega \setminus (\text{ext}(C) \cup \text{ext}(D))|$
et

$$P = \frac{N_{11} + N_{22}}{|\Omega|}$$

$$P' = \frac{|\,\text{ext}(C)| \times |\,\text{ext}(D)| + |\Omega/\,\text{ext}(C)| \times |\Omega/\,\text{ext}(D)|}{|\Omega|^2}$$

ainsi,

$$\sigma_{Kappa}(C, D) = \frac{P - P'}{1 - P'}$$

D'après [52] il est possible de calculer un test statistique Z à partir de cette similarité :

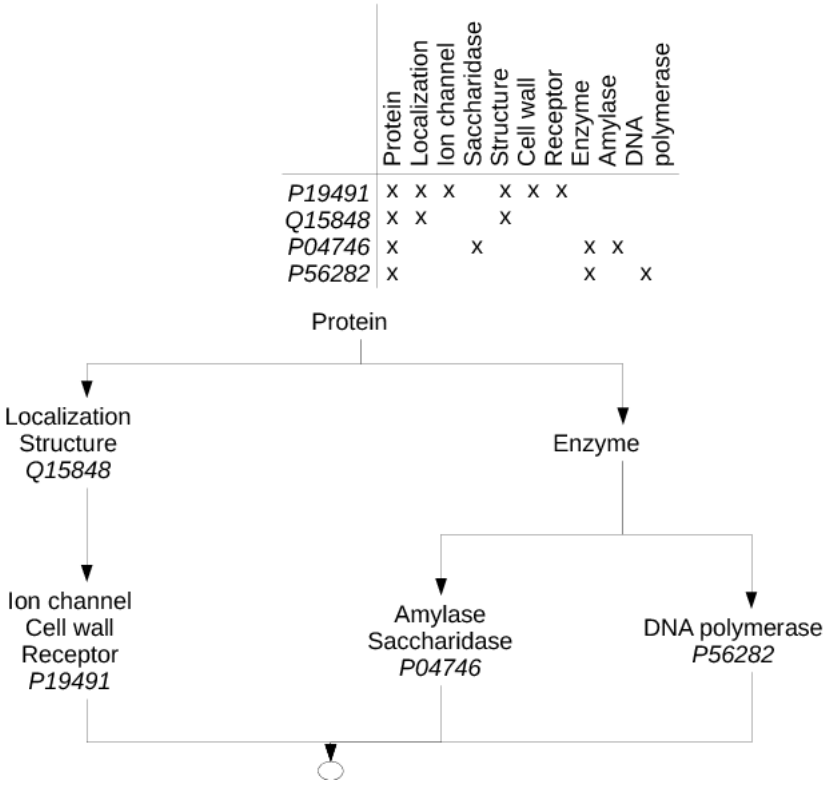
$$Z = \sigma_{Kappa}(C, D) \times \sqrt{\frac{(N_{11} + N_{12} + N_{21} + N_{22})(1 - P')}{P'}}$$

La valeur de Z suit une distribution normale. Nous pouvons considérer avec une erreur de 5% que nous rejetons l'hypothèse nulle "les deux concepts sont différents" contre l'hypothèse "les deux concepts sont équivalents" si

$$Z \geq 1.64486$$

Analyse de concepts formels Il est possible d'utiliser des outils de l'analyse de concepts formels (FCA) [40] pour déduire des relations entre les concepts. L'analyse de concepts formels est une méthode de classification symbolique qui a pour but de regrouper les éléments ayant des propriétés communes. La classification hiérarchique résultante qui regroupe des objets qui

FIG. 4.3 – *Illustration de l'utilisation d'outil d'analyse formel de concepts*



partagent des propriétés identiques est appelée treillis de concepts (ou treillis de Galois). En analyse de concepts formels, un concept peut être défini par son intension (l'ensemble de ses attributs ou propriétés) et son extension (l'ensemble des documents possédant ces attributs). La figure 4.3 montre comment il est possible d'organiser les concepts formels sur un exemple simple. En alignement d'ontologies, les concepts des ontologies deviennent les attributs des concepts formels et les instances annotées par ces concepts sont les documents.

Le tableau de la figure 4.3 décrit un ensemble d'instances et les classes auxquelles elles appartiennent (les classes proviennent des deux ontologies que nous souhaitons aligner). On construit le treillis de Galois correspondant, voir la figure 4.3. De l'analyse du treillis, on peut déduire les correspondances suivantes :

Localization	$\equiv$	Structure	Localization	$\geq$	Cell wall	Structure	$\geq$	Ion channel
Localization	$\geq$	Receptor	Ion channel	$\equiv$	Cell wall	Ion channel	$\equiv$	Receptor
Enzyme	$\geq$	Saccharidase	Amylase	$\equiv$	Saccharidase			

Où $\equiv$ désigne l'équivalence entre concepts et $\geq$ la subsomption. Il faut noter que les relations trouvées dans ce tableau ne sont pas forcément des réalités biologiques, mais simplement le résultats de l'analyse des données dont nous disposons dans cet exemple.

Techniques utilisant l'apprentissage Il est possible d'utiliser des techniques issues de l'apprentissage pour identifier des concepts équivalents entre eux. Pour cela, les instances et les concepts de l'une des ontologies serviront de jeu d'apprentissage, et celles de l'autre ontolo-

gie de jeu de prédiction. L'utilisation d'algorithmes d'apprentissage permet de dégager des caractéristiques des concepts au niveau des instances dans le jeu d'apprentissage : l'algorithme trouve les valeurs des propriétés des instances qui permettent de discriminer les concepts les uns des autres. Ces caractéristiques sont ensuite utilisées sur les instances du jeu de prédiction, et permettent d'assigner à un ensemble d'instances le concept lui correspondant.

Les techniques mises en oeuvre peuvent être nombreuses. Dans la section 4.5, nous présenterons des systèmes utilisant un classifieur Bayésien naïf.

4.5 Systèmes pour l'alignement

La recherche de correspondances entre les ontologies a fait l'objet de nombreux travaux dans plusieurs communautés scientifiques. Les problématiques d'alignement d'ontologies et d'alignement de schémas de bases de données relationnelles sont en effet très proches. Les concepts de l'ontologie correspondent aux attributs des schémas de bases de données relationnelles, de même pour les relations. Les techniques mises en oeuvres sont compatibles et se sont souvent complétées mutuellement. Dans cette section, nous nous intéressons aux systèmes les plus marquants issus de l'une ou l'autre de ces communautés. Nous les classons en fonction du type de correspondances recherchées et du type de données utilisées pour la détection de ces correspondances.

L'ensemble des méthodes d'alignement utilise une représentation interne des ontologies ou des schémas souvent différente de celle utilisée pour échanger et représenter l'ontologie (ou le schéma) en dehors de la méthode. La première étape de l'alignement ou de la fusion est donc le plus souvent une traduction de l'ontologie (ou du schéma) dans le formalisme de représentation interne. Les formalismes de représentation interne peuvent être très variables selon les techniques qui sont mises en place. L'intérêt de cette traduction étant minime, nous ne détaillerons pas ici cette première étape.

4.5.1 Méthodes basées sur les schémas

DELTA [23] est un système qui cherche à établir des correspondances entre les attributs de deux schémas de base de données. L'approche est semi-automatique, avec une part très importante laissée à l'expert pour valider chaque mapping.

Le système est basé exclusivement sur des comparaisons de chaînes de caractères. Toutes les informations disponibles sur un attribut sont transformées en chaînes de caractères. Les attributs sont ensuite comparés entre eux à l'aide de cette chaîne. DELTA cherche à établir des correspondances 1–1, en proposant à l'expert pour chaque attribut l'ensemble des attributs classés selon la proximité syntaxique à la chaîne correspondante à l'attribut d'intérêt. Charge à lui de choisir la correspondance la plus pertinente.

Les données alignées sont des schémas relationnels issus de l'*U.S. Air force*.

SKAT/ONION Le *Semantic Knowledge Articulation Tool*, SKAT [79], est un système qui cherche à établir des correspondances entre les ontologies. L'expert y joue un rôle central, puisqu'il donne au système des règles de *match* et de *mismatch* exprimées en logique du premier

ordre, et qu'il approuve ou rejette les matchs proposés automatiquement par le système, et produit donc l'alignement final.

ONION (ONtology compositION) [80] est le successeur de SKAT. Il a pour but de trouver les correspondances entre de multiples ontologies pour permettre de répondre de façon unifiée à des requêtes sur ces ontologies. L'alignement est donc vu comme un ensemble de règles d'articulation (règles qui fournissent des relations entre les concepts des ontologies) permettant le passage d'une ontologie à l'autre.

Anchor-PROMPT [89] est une extension de PROMPT [88]. Il s'agit d'un outil d'alignement et de fusion d'ontologies. Le système identifie des ancres qui sont des correspondances évidentes, à l'aide de comparaisons de chaînes de caractères. Il va ensuite analyser les chemins limités par les ancres entre les ontologies, en utilisant des propriétés structurales, en particulier sur les types de relations qui lient les concepts. Puis le système calcule les fréquences des concepts apparaissant souvent à la même place dans les chemins. En utilisant les fréquences ainsi déterminées et les retours de l'expert, l'algorithme définit une correspondance entre les concepts.

L'algorithme PROMPT utilise de nombreux matchers basés sur des mesures de similarité entre les chaînes de caractère, telles que la distance d'édition, ou des mesures personnalisées. Il peut également utiliser des informations liées à la structure telles que les domaines et les types de données pour déterminer les correspondances.

Cupid [71] implémente un algorithme utilisant des techniques basées sur les similarités syntaxiques et structurales. Le système calcule des coefficients de similarité en utilisant des thesauri spécifiques au domaine étudié. Les schémas donnés en entrée sont représentés sous forme de graphes ; les noeuds représentent les éléments du schéma et sont parcourus de façon *bottom-up* et *top-down*.

L'algorithme de matching procède en trois phases et ne fonctionne que pour les structures d'arbre.

COMA/COMA++ COMA [29] (COmbination of MAtching algorithms) est un outil de matching de schéma basé sur une composition parallèle de matchers (les matchers sont évalués indépendamment les uns des autres). Il met à disposition une bibliothèque étendue d'algorithmes de matching, un cadre pour combiner les résultats obtenus et une plateforme pour l'évaluation de l'efficacité de chaque matcher.

COMA contient 6 matchers élémentaires, 5 matchers hybrides et 1 matcher orienté sur la réutilisation de mappings déjà obtenus par ailleurs. La plupart de ces matchers sont basés sur la comparaison de chaînes de caractères. Un composant original, appelé *reuse-oriented matcher*, essaie de réutiliser les résultats précédemment obtenus pour de nouveaux schémas ou fragments de schéma.

Au sein de COMA, les schémas sont représentés sous la forme d'un DAG dont les noeuds sont les chemins. Cela a pour but de capturer le contexte dans lequel chaque élément apparaît. Les éléments qui distinguent COMA de Cupid sont une architecture plus flexible et la possibilité de réaliser des itérations dans le processus du matching. Cela présume des interactions avec l'utilisateur qui approuve les matchs ou mismatchs pour améliorer graduellement la précision du matcher.

COMA++ est construit au dessus de COMA. Il introduit plus de détail dans la réutilisation d'alignement et fournit aussi une meilleure implémentation de COMA ainsi qu'une interface graphique.

COMA recherche des relations 1–1 entre les concepts des ontologies qu'il matche.

Similarity flooding Similarity flooding [76] est basé sur l'idée de la propagation des similarités. Les schémas sont représentés sous forme de graphes orientés et étiquetés.

L'algorithme commence par une méthode basée sur la similarité entre les chaînes de caractères pour obtenir une version initiale de l'alignement, ensuite raffinée par des itérations successives du processus. Le concept basique sous-jacent à Similarity Flooding est la propagation de la similarité à partir de noeuds similaires vers les voisins grâce à des propagations de coefficients.

S-Match [43] prend en entrée deux structures exprimées sous forme de graphes, telles que des classifications, des schémas XML, des ontologies. Il renvoie comme résultats des relations logiques, par exemple des relations d'équivalence, de subsomption...

Les relations sont exprimées (i) en décrivant les entités des ontologies sous forme logique et (ii) en réduisant le problème de l'alignement à un problème de validation de propositions. En particulier, les entités sont traduites en formules propositionnelles. Cela permet une traduction du problème de correspondance en un problème de satisfiabilité en logique propositionnelle, qui peut être efficacement résolu par l'utilisation de SAT Solver.

S-Match a été conçu et développé comme une plateforme pour le matching sémantique (matching permettant d'établir non seulement des relations d'équivalence, mais aussi de mismatch, de recouvrement, ...), c'est un système modulaire de calcul de relation sémantique dont les composants peuvent être ajoutés, enlevés ou personnalisés à volonté. Le système utilise une composition parallèle au niveau des matchers. Les ontologies d'entrée (sous forme d'arbre) sont représentées intérieurement au format XML.

Un module utilise des oracles qui apportent des connaissances du domaine (WordNet, UMLS...). L'output est un arbre enrichi (PTree) stocké dans une base de données qui en permet l'édition et la navigation.

S-Match contient une bibliothèque de 20 matchers basiques au niveau des éléments répartis dans 3 catégories : comparaison de chaînes de caractères, basés sur la structure, et dépendant du domaine (utilisant l'UMLS par exemple).

4.5.2 Systèmes basés sur les instances

GLUE [31] est le successeur de LSD [32]. C'est un système qui emploie de multiples techniques de fouille de données pour découvrir de façon semi-automatique un mapping sémantique 1–1 entre deux taxonomies. L'idée de l'approche est de calculer la similarité comme une fonction de la distribution jointe des concepts.

GLUE utilise une approche d'apprentissage multistratégie en trois étapes : la première apprend les probabilités jointes des distributions des classes des deux taxonomies. GLUE utilise pour cela deux matchers, l'un basé sur du Bayésien naïf et l'autre sur la comparaison des noms des concepts. Un poids est assigné manuellement à chacun des matchers. La deuxième étape

réside dans l'estimation de la similarité entre les concepts en utilisant une fonction définie par l'utilisateur. Il en résulte une matrice de similarité entre les termes des taxonomies. Enfin, la troisième et dernière étape applique des contraintes supplémentaires sur la structure. Un exemple de contrainte est : "si tous les enfants d'un noeud x matchent le noeud y alors x matche aussi y.

iMAP [28] recherche des correspondances entre des schémas de bases de données relationnelles. Le problème de la mise en correspondance est reformulé en un problème de recherche dans un espace de match, qui peut être de très grande taille. Pour effectuer cette recherche efficacement, iMAP utilise de multiples matchers basiques basés sur des comparaisons de textes, d'unités, etc., qui s'occupent chacun d'une sous-partie de l'espace de recherche.

iMAP est un système semi-automatique capable de découvrir des correspondances 1–1 (par exemple, amount ≡ quantity), mais aussi des correspondances plus complexes de style agrégation (par exemple, address ≡ concat(city, street)).

Le système fonctionne en trois étapes. Tout d'abord, les correspondances candidates sont générées en utilisant les matchers basiques. La stratégie de recherche est pilotée par une technique de *beam search* [100] (algorithme de recherche heuristique qui permet de réduire l'espace de recherche en sélectionnant les résultats les plus intéressant selon un critère heuristique). La seconde étape consiste à évaluer toutes les correspondances candidates en utilisant des matchers plus complexes basés par exemple sur un prédicteur Bayésien naif, qui serait trop coûteux à utiliser sur l'ensemble des correspondances candidates initiales. Enfin, la dernière étape consiste en la sélection de la meilleure correspondance pour chaque concept. Le système est aussi capable d'expliquer le résultat grâce à un module d'explication.

CAIMAN [60] (Community support Application Interoperability by MApping oNtologies) est un système principalement dédié à l'alignement d'ontologies simples, principalement des hiérarchies. Il s'agit d'un système semi-automatique permettant le partage de documents.

Le système calcule une valeur de probabilité entre les concepts des deux ontologies en utilisant des techniques d'apprentissage principalement basées sur les textes (Rocchio classifier, TFIDF...). Finalement, les résultats au dessus d'un seuil fixé arbitrairement sont renvoyés.

FCA-Merge Comme son nom l'indique, FCA [109] utilise des techniques d'analyse de concepts formels. Il permet la fusion de deux ontologies du même domaine, annotant un ensemble de documents.

Le système fonctionne en trois étapes :
- extraction des instances. Le système analyse les documents associés aux ontologies et en extrait les concepts par des techniques de traitement du langage naturel.
- calcul de treillis des concepts
- génération interactive de l'ontologie fusionnée.

La première étape, d'extraction des instances, permet de travailler avec des ontologies qui ne partagent pas d'instances, puisque les instances sont découvertes par FCA-Merge. Les deux ontologies doivent néanmoins être associées à un même ensemble de documents.

4.5.3 Approches mixtes basées sur les schémas et les instances

Clio [78] permet l'alignement de schémas XML et relationnel. Le système est conçu pour permettre de gérer et faciliter l'intégration et la transformation de données au sein d'un environnement hétérogène.

Clio a pour philosophie de rendre l'alignement opérationnel. Le système utilise un ensemble de techniques de *data management* ayant prouvé leur valeur. La première étape d'identification des concepts se correspondant est réalisée automatiquement avec l'aide de composants de matching de schéma ou manuellement. L'algorithme de matching de schéma de Clio combine de manière séquentielle des matchers basés sur les instances et des matchers utilisant des techniques variées telles que classifieur Bayésien naïf, comparaison de chaînes de caractères...

Dans un second temps, en utilisant les correspondances candidates précédemment calculées ainsi que des contraintes issues de schémas données en entrée (contraintes liés aux clés primaires et secondaires), le système produit un alignement.

Grâce à son formalisme de représentation, Clio permet la composition de mappings.

IF-Map [55] est basé sur la théorie du flot d'information. Le système réalise l'alignement de deux ontologies en observant comment elles sont liées à une ontologie de référence. Les ontologies alignées doivent avoir des instances (sans nécessairement les partager), mais pas l'ontologie de référence.

S'il est possible de rapprocher les concepts des ontologies alignées à un concept de l'ontologie de référence et que leurs instances peuvent être assignées à un concept de l'ontologie de référence, alors IF-Map utilise les outils d'Analyse de Concepts Formels entre les 3 ontologies, et les instances pour trouver le treillis de Gallois dont sera extrait l'alignement.

Si un tel mapping n'est pas disponible, IF-Map va générer des paires de candidats et des instances artificiellement en utilisant des heuristiques basées sur les chaînes de caractères.... La méthode IF-Map est spécifiée en logique de Horn et exécutée par un interpréteur Prolog. Elle gère des ontologies au format RDF et KIF.

OLA [35] a été conçu dans l'idée d'équilibrer les contributions de chaque composant des ontologies (classes, contraintes, instances...). Il prend en entrée deux ontologies exprimées en OWL. La similarité entre concepts suit deux principes : (i) elle dépend de la catégorie de nœud concernée (classe, propriété...), et (ii) elle prend en compte toutes les caractéristiques de cette catégorie (superclasses...).

La similarité entre les concepts est calculée à partir d'un système d'équations utilisant des similarités basées sur les comparaisons de chaînes de caractères, la structure... L'algorithme commence par les similarités basées sur les labels et les types de données. Il relance ensuite itérativement un algorithme de point fixe jusqu'à obtenir des valeurs qui ne s'améliorent plus. À partir de ces valeurs, un alignement est retourné.

FALCON-AO [50] permet l'alignement de deux ontologies exprimé en OWL. Il est constitué de 3 composants principaux : V-Doc, un matcher linguistique (calculé sur la base de TF-IDF,

WordNet...), GMO un matcher structural itératif, et PBM un matcher dédié au partitionnement de grandes ontologies.

Il permet de chercher des alignements *many-to-many*, et peut travailler aussi bien sur des ontologies que sur des schémas relationnels.

TaxoMap [94] utilise une stratégie de partitionnement des ontologies basée sur leur structure, puis d'alignement des partitions formées. Les méthodes d'alignements se concentrent ensuite sur ces parties pour rechercher des correspondances.

TaxoMap est capable de détecter une large variété de relations entre les concepts des ontologies : des équivalences entre les labels, des inclusions, et des relations de proximité (est proche de).

4.5.4 OAIE Challenge

Depuis 4 ans, en marge de la conférence *International Semantic Web Conference* a lieu un workshop dédié à l'alignement d'ontologies *Ontology Alignment Evaluation Initiative*. L'un des grands intérêts de ce workshop est le challenge OAIE qui propose des *benchmarks* de test pour comparer les forces et les faiblesses des différentes méthodes d'alignement ainsi que leur capacité à résoudre les problèmes posés actuellement par les alignements (utilisation des instances, explication des alignements...).

De l'édition 2008, il ressortait que certains matchers simples permettent de trouver d'excellents résultats, mais que leurs performances sont dégradées par les autres matchers si on les combine entre eux. Ce constat est partagé par les auteurs de [42], qui vont même jusqu'à proposer un système d'alignement basé exclusivement sur la comparaison des noms de labels et dont les résultats sont meilleurs qu'un système plus complexe. Nous nous pencherons plus en avant sur cette observation dans le prochain chapitre.

4.6 Microbiogenomics

Ce travail s'inscrit dans le cadre du projet ANR Masse de données Microbiogenomics dans lequel l'équipe Bioinformatique est fortement impliquée et qui a débuté en 2005.

Les équipes participant au projet sont : l'équipe MIG (Mathématique Informatique et Génomes) de l'INRA de Jouy-en-Josas, l'équipe EMBG (Évolution Moléculaire et bioinformatique des Génomes) de l'Institut de Génétique et Microbiologie d'Orsay, l'équipe Bioinformatique du LRI, CNRS-Université Paris-Sud XI, et le projet inSitu du LRI, INRIA CNRS Université Paris-Sud XI.

Les objectifs de ce projet étaient doubles. Il visait d'une part à une meilleure compréhension du fonctionnement des génomes procaryotes par l'analyse des données génomiques de comparaison de ces génomes, et plus précisément l'étude de l'évolution des gènes, protéines et des génomes procaryotes, et d'autre part à une meilleure efficacité des tâches d'annotation et de réannotation fonctionnelle de génomes. Pour cela le projet a exploré différentes pistes : (i) la fouille de données, en particulier autour des arbres phylogénétiques. L'objectif est de mettre au point des méthodes de comparaison et de clustering d'arbre, et aussi de pouvoir les comparer, et en déduire ainsi des arbres consensus par exemple. (ii) La visualisation de grandes quantités de résultats, qui

fait appel à des méthodes d'interface homme machine avec le développement de visualisation multi-échelles. (iii) Enfin, il apparait clairement la nécessité d'intégrer les données. Pour cela l'entrepôt de données Microbiogenomics a été conçu. Il est capable de gérer de grandes quantités de données issues de sources hétérogènes. La provenance des données, information capitale aux utilisateurs est conservée et les sources restent indépendantes, facilitant ainsi l'intégration de nouvelles sources de données aussi bien publiques que privées. Les données de l'entrepôt sont conservées localement, pour en permettre un accès rapide, en accord avec la volonté d'utiliser des algorithmes de fouilles de données. Avec l'entrepôt Microbiogenomics a été conçu un système d'interrogation : GenoQuery [66] .

Notre travail s'inscrit dans l'intégration des données. Certaines des données génomiques de Microbiogenomics proviennent de l'annotation manuelle de génomes. Nous avons vu dans ce chapitre que les ontologies biologiques étaient nombreuses et aussi pertinentes les unes que les autres selon les informations que l'utilisateur souhaitait exprimer. Mais il est nécessaire de pouvoir accéder de façon transparente à l'ensemble de ces informations intégrées aussi bien pour les tâches d'annotation ou de réannotation (correction ou confirmation des fonctions des protéines d'un génome, au vu de connaissances nouvelles) que pour les tâches de fouilles de données. L'établissement de correspondances entre les concepts des différentes hiérarchies biologiques est apparu nécessaire. La nature des données, et des applications ont nécessité la conception d'une méthode nouvelle et fortement ancrée dans les données biologiques. Le prochain chapitre présentera notre contribution à la résolution de ce problème : O'Browser.

Chapitre 5

O'Browser : une nouvelle méthode pour l'alignement d'ontologies biologiques

Sommaire

Les ontologies biologiques sont de taille très variable : les plus petites font moins d'une centaine de concepts, tandis que les plus grandes peuvent atteindre plusieurs dizaines voire des centaines de milliers de concepts. Si nous adoptons une approche naïve du problème de l'alignement de deux ontologies, il nous faudra comparer l'ensemble des concepts de l'ontologie 1 à l'ensemble

des concepts de l'ontologie 2. Avec de très grandes ontologies, le nombre de paires de concepts candidats est trop important pour permettre l'application des méthodes actuelles de mapping. Il est donc nécessaire d'en diminuer le nombre. Nous allons dans un premier temps nous intéresser à deux stratégies mises en place dans notre système pour réduire le nombre de paires de concepts candidats au mapping : l'ancrage et le typage.

Ensemble de concepts Soit $\mathcal{O}_1$ (resp. $\mathcal{O}_2$) une ontologie, nous notons O_1 (resp. O_2) l'ensemble des concepts de l'ontologie.

5.1 Équivalence entre concepts

Pour définir formellement l'équivalence entre concepts, nous nous inspirons des définitions de Euzenat et Shvaiko [34] sur la sémantique de l'alignement, que nous adaptons au cas des ontologies biologiques pour l'annotation de protéines.

Intuitivement, nous disons que deux concepts C et D sont équivalents s'ils partagent le même ensemble de protéines ancestrales. Nous formalisons cette notion comme suit.

Soit U_t l'ensemble des protéines ancestrales à un instant t. Soit $\mathcal{O}_1$ et $\mathcal{O}_2$ deux ontologies. Soit Γ_1 (resp. Γ_2) l'ensemble de génomes annotés par des concepts de l'ontologie O_1 (resp. O_2).

L'ensemble des protéines de génomes de Γ_1 annotées par le concept C de O_1, $m_{O_1}(C)$ est défini par :
$m_{O_1}(C) = \{p / \exists g \in \Gamma_1$ et $p \in g$ et p est annotée par $C\}$
L'ensemble des protéines de génomes de Γ_2 annotées par le concept D de O_2, $m_{O_2}(D)$ est défini par :
$m_{O_2}(D) = \{p / \exists g' \in \Gamma_2$ et $p \in g'$ et p est annotée par $D\}$

Soit E un ensemble de protéines, $\gamma(E)$ est l'ensemble des protéines ancestrales des protéines de E :
$\gamma(E) = \{p \in U_t$ tel que $\exists q \in E$ et p est ancêtre de $q\}$

La correspondance $(c, d, \equiv, score) \in O_1 \times O_2 \times \mathcal{R} \times I\!R$ est satisfiable pour Γ_1, Γ_2 et U_t si et seulement si :

$$\frac{|\gamma(m_{O_1}(C)) \cap \gamma(m_{O_2}(D))|}{|\gamma(m_{O_1}(C)) \cup \gamma(m_{O_2}(D))|} = score$$

Plus les protéines annotées par les concepts C et D de O_1 et O_2 respectivement ont le même ensemble de protéines ancestrales, plus les fonctions biologiques décrites par les concepts sont proches, et plus le score est élevé.

En pratique, l'ensemble des protéines ancestrales n'est pas connu puisque nous n'avons accès qu'aux génomes actuels. Nous allons approximer cette relation par une mesure de la spécificité et de la réciprocité des relations d'homologie entre les protéines annotées par les concepts C et D. Un matcher sera particulièrement dédié à cette tâche, le matcher homologie (introduit dans la section 5.3.1).

5.2 Détection des ancres et des types par l'utilisation des connaissances du domaine

Dans O'Browser [93], les connaissances du domaine sont exploitées de deux façons : d'une part en utilisant les données biologiques proprement dites, et d'autre part en utilisant l'expertise du biologiste ou du bioinformaticien. Nous détaillons dans cette section deux étapes de O'Browser dans lesquelles l'expert est impliqué, et qui permettront une réduction de l'espace des candidats au mapping.

5.2.1 Ancres

La première étape de O'Browser est la recherche d'**ancres**. Une ancre est constituée de deux concepts sémantiquement très proches d'après une connaissance *a priori* du domaine, qui doivent donc être mis en correspondance dans l'alignement.

Deux concepts impliqués dans une ancre font souvent consensus auprès des experts. Ils sont utilisés dans les mêmes circonstances et avec le même vocabulaire. Ainsi, les ancres sont facilement identifiables à l'aide de méthodes simples et relativement peu coûteuses. Dans O'Browser, les ancres sont identifiées en utilisant (a) une quasi-identité entre les noms de concepts et/ou (b) les relations d'homologie entre les protéines annotées par l'un des deux concepts. Le maximum des scores calculés à partir de (a) et (b) est associé à chaque paire de concepts. Seuls les meilleurs résultats sont pris en compte : ils sont établis à partir d'un seuil décidé par un administrateur du système.

Les paires avec un score supérieur ou égal à ce seuil sont proposées à l'expert pour validation. Une paire candidate peut être validée ou refusée. Nous demandons ici une explication à l'expert sur les motivations du rejet d'une paire candidate. Trois raisons peuvent conduire à ce rejet : (i) les deux concepts associés ne sont pas comparables du tout parce qu'ils concernent deux champs différents de la biologie (e.g. *Germination* dans le cas des plantes et *Germination* dans le cas des Procaryotes), (ii) les deux concepts sont comparables mais la paire de concepts ne peut pas être considérée comme une ancre parce que les notions, bien que proches, restent différentes (e.g. *Metabolism of DNA* et *Metabolism of RNA*), et (iii) l'un des deux concepts est trop général par rapport à l'autre (e.g. *Metabolism of Amino-Acid* et *Metabolism of Glutamine*).

Cette approche n'est pas la seule envisageable : on pourrait réutiliser des ancres préalablement identifiées, ou demander à l'expert d'en fournir lui-même une liste.

D'autres systèmes utilisent les ancres [48, 89, 50]. Elles sont identifiées automatiquement par comparaison des labels de concepts, O'Browser demande à l'expert une validation de ces ancres et tire, pour l'instant, des informations de l'étape de validation plus que de l'ancre elle-même.

Après la phase de validation, toutes les paires validées par l'expert sont considérées comme des ancres.

L'analyse de l'étape de validation des ancres va nous permettre d'établir des contraintes pour les candidats. Le cas (iii) nous permet d'éliminer des candidats au mapping :
Si un concept C de O_1 est déclaré par l'expert plus général qu'un concept D de O_2, nous pouvons déduire que C, et aucun ancêtre de C, ne seront jamais mis en correspondance avec D ou un descendant de D.

Définition 5.2.1 (Candidats exclus) On définit $CANDIDAT\_EXCLU \subseteq O_1 \times O_2$ comme l'ensemble des couples de concepts qui ne pourront jamais être mis en correspondance.

Si C concept de O_1 est déclaré plus général que D concept de O_2, alors :
$(C, D) \in CANDIDAT\_EXCLU$ et
$\forall\, C' \in O_1, \forall\, D' \in O_2[(C \preceq C' \wedge D' \preceq D) \Rightarrow (C', D') \in CANDIDAT\_EXCLU]$.

Nous pouvons déduire une autre contrainte à partir du cas (i) : si l'ancre (C, D) est acceptée, c'est qu'aucun descendant du concept C de O_1 ne sera jamais mis en correspondance avec un concept ancêtre de D de O_2 et réciproquement, aucun ancêtre de C ne sera jamais mis en correspondance avec un descendant de D. Nous notons $\mathcal{A}$ l'ensemble des ancres.

Formellement :

Nous dirons que C est strictement subsumé par D, noté $C \prec D$ si et seulement si $C \preceq D$ et $C \neq D$.

$\forall\, C \in O_1, \forall\, C' \in O_1, \forall\, D \in O_2, \forall\, D' \in O_2\{[(C, D) \in \mathcal{A} \wedge C \prec C' \wedge D' \prec D] \Rightarrow (C', D') \in CANDIDAT\_EXCLU\}$

$\forall\, C \in O_1, \forall\, C' \in O_1, \forall\, D \in O_2, \forall\, D' \in O_2\{[(C, D) \in \mathcal{A} \wedge C' \prec C \wedge D \prec D'] \Rightarrow (C', D') \in CANDIDAT\_EXCLU\}$

5.2.2 Types

Nous proposons d'associer des **types** aux concepts des ontologies pour l'annotation fonctionnelle. Les concepts qui appartiennent à un même champ de génomique fonctionnelle sont associés à un même type. Comme mentionné précédemment, différents points de vue sur l'annotation sont regroupés dans une hiérarchie fonctionnelle (e.g. le métabolisme, la localisation cellulaire...). Or des concepts de types distincts ne seront jamais mis en correspondance. L'utilisation des types pour générer des paires de concepts candidats au mapping, permet ainsi de réduire l'espace de recherche puisque toutes les paires n'ont pas besoin d'être considérées, mais seulement celles regroupant deux concepts du même type.

Dans notre approche, l'identification des types fait intervenir l'expert biologiste. Au départ, il utilise les concepts les plus généraux des ontologies pour définir les types de base ($BASE\_TYPE$). Les types ainsi définis sont ensuite propagés automatiquement aux concepts descendants, construisant ainsi la relation $TYPE$.

Les approches des systèmes Falcon-AO [50] et TaxoMap [94] reposent aussi sur un partitionnement des ontologies. Il est réalisé à partir d'informations structurales dont nous ne disposons pas.

Plus formellement, nous avons : soit T l'ensemble des termes représentant les fonctions des gènes et soit $\preceq$ la relation de subsomption entre les concepts des ontologies.

- $BASE\_TYPE \subseteq (O_1 \cup O_2) \times T$
- $TYPE$ est la plus petite relation incluse dans $(O_1 \cup O_2) \times T$ qui contient $BASE\_TYPE$ et est fermée pour la relation de subsomption de O_1 et O_2 :
  (1) $BASE\_TYPE \subseteq TYPE$

(2) $\forall C, C' \in O_1 \cup O_2, \forall \tau \in T$
$[(C' \preceq C \wedge (C, \tau) \in TYPE) \Rightarrow (C', \tau) \in TYPE]$
$\nexists TYPE' \subset TYPE$ vérifiant les propriétés (1) et (2)

Nous supposons qu'une relation de subsomption $\trianglelefteq$ est définie dans T. La relation $TYPE$ est close pour cette relation de subsomption.

$\forall\, t \in T,\ \forall\, t' \in T,\ \forall\, C \in O_1 \cup O_2$
$\{[(C, t) \in TYPE \wedge t \trianglelefteq t'] \Rightarrow (C, t') \in TYPE\}$

5.2.3 Automatisation de la découverte des types

L'utilisation des types va permettre de réduire fortement le nombre de paires de concepts candidats au mapping. Nous présentons dans cette partie une méthode pour en automatiser l'identification, et assister l'utilisateur dans sa recherche de types. Dans ce but, nous proposons d'utiliser deux sortes d'informations : des informations issues de l'analyse de relations biologiques au sein des ontologies d'une part, et des informations issues de l'analyse des ancres d'autre part.

Utilisation des informations biologiques

Nous utilisons les relations d'homologie entre les protéines annotées par les ontologies biologiques. Dans la section 5.3.1, nous utiliserons ces mêmes informations pour établir des liens entre les concepts de deux ontologies distinctes, mais dans cette section ce sont les relations entre les concepts d'une seule et même ontologie qui nous intéressent.

Nous faisons l'hypothèse que s'il existe des relations d'homologie entre les protéines annotées par un concept C_1 et un concept C_2 d'une même ontologie, alors les deux concepts sont liés par une relation. Par exemple, les concepts *Cell Wall* fils du concept *Localization* et *Cell Wall* fils du concept *Biological Process* seront très probablement liés car les protéines qu'ils annotent sont homologues entre elles, et même dans ce cas précis ils annotent en partie les mêmes protéines. Dans cet exemple, les relations sont de type `is-localized-in` et `has-biological-process`.

Cependant deux concepts liés par homologie ne sont pas forcément proches sémantiquement (ils ne partagent pas le même type). L'exemple ci-dessus en est une illustration : les protéines peuvent être annotées par l'un et l'autre des concepts parce qu'ils reflètent des points de vue différents sur l'annotation et non en raison de leur proximité sémantique. Or nous souhaitons pouvoir identifier les concepts partageant les mêmes types. Pour identifier de tels cas, nous ajoutons une nouvelle condition : la proximité structurale. Nous considérons que si deux concepts sont proches dans l'ontologie, il est probable qu'ils reflètent un même point de vue sur les données. Deux concepts "proches" dans l'ontologie et partageant des relations d'homologie seront considérés comme étant du même type.

Formalisons ces quelques idées :
Nous notons $\overline{T}$, l'ensemble des types définis automatiquement, ou étendus grâce aux relations d'homologies établies entre les concepts d'une même ontologie. $\overline{T}$ est tel que $T \subseteq \overline{T}$.

Nous considérons que nous disposons d'une relation $PROCHE \subseteq (O_1 \times O_1) \cup (O_2 \times O_2)$ qui est l'ensemble des paires de concepts proches dans une ontologie (c'est-à-dire dont les instances par-

tagent des relations d'homologie et proches structuralement). Nous définissons $TYPE_1$ comme la plus petite relation incluse dans $(O_1 \cup O_2) \times \overline{T}$ qui contient $TYPE$ et qui vérifie les propriétés suivantes :

Nous distinguons plusieurs cas de propagation du type entre deux concepts proches :
Soit C et D deux concepts de O_1 (resp. O_2).

1er cas : le concept C a un type, le concept D n'en a pas, et C et D sont proches. Le type existant dans TYPE est alors propagé de C vers D dans $TYPE_1$.

Propriété 5.2.1 (Propagation du type au voisinage) .
$\forall\, C, D \in (O_1 \cup O_2)\ \forall \tau \in T$
$\{[(C,\tau) \in TYPE \wedge \nexists \tau' \in T((D,\tau') \in TYPE) \wedge (C,D) \in PROCHE]$
$\Rightarrow (D,\tau) \in TYPE_1\}$

2e cas : les concepts C et D n'ont pas de types mais sont proches. S'ils ont un type un jour alors ce sera le même.

Propriété 5.2.2 $\{\nexists \tau \in T((C,\tau) \in TYPE) \wedge \nexists \tau' \in T((D,\tau') \in TYPE) \wedge (C,D) \in PROCHE]$
$\Rightarrow \forall\, t \in \overline{T}[(C,t) \in TYPE_1 \Leftrightarrow (D,t) \in TYPE_1]\}$

3e cas : les deux concepts ont deux types distincts τ et τ' et sont proches. On ajoute un type à $\overline{T}$, obtenu par exemple par la concaténation des deux types τ et τ', ou par le nom d'un ancêtre commun à τ et τ', et on prolonge la relation de subsomption $\trianglelefteq$ de T à $\overline{T}$.

Propriété 5.2.3 $\forall\, C, D \in O_1 \cup O_2\ \forall \tau, \tau' \in T$
$\{[(C,\tau) \in TYPE \wedge (D,\tau') \in TYPE \wedge \tau \neq \tau' \wedge (C,D) \in PROCHE]$
$\Rightarrow [\exists \tau'' \in \overline{T} \backslash T((C,\tau'') \in TYPE_1 \wedge (D,\tau'') \in TYPE_1 \wedge \tau \trianglelefteq \tau'' \wedge \tau' \trianglelefteq \tau'')]\}$

Enfin, on propage les types nouvellement définis aux descendants :

Propriété 5.2.4 (Propagation aux descendants) $\forall\ C, C' \in O_1 \cup O_2, \forall \tau \in \overline{T}\ \{[C' \preceq C \wedge (C,\tau) \in TYPE_1] \Rightarrow (C',\tau) \in TYPE_1\}$

Utilisation de la validation des ancres

En commentant les résultats de l'étape de validation des ancres, l'expert fournit des informations très importantes. Lorsqu'il valide une ancre ou la refuse, il peut indiquer si les deux concepts sont de même type ou non.

L'analyse des résultats de la validation va nous fournir un ensemble $\mathcal{A}$ de couples de concepts de même type, avec $\mathcal{A} \subseteq O_1 \times O_2$.

Formellement :
Nous notons T^*, l'ensemble des types définis automatiquement, ou étendus grâce aux ancres à

partir de $\overline{T}$. T^* est tel que $T \subseteq \overline{T} \subseteq T^*$.

Nous définissons $TYPE_2$ comme la plus petite relation incluse dans $(O_1 \cup O_2) \times T^*$ qui contient $TYPE_1$ et qui vérifie les propriétés 5.2.1, 5.2.2 et 5.2.3 adaptées dans le cas de la validation des ancres. Plus précisément, on passe de $TYPE_1$ à $TYPE_2$ de la même façon que de $TYPE$ à $TYPE_1$, en remplaçant T par $\overline{T}$, $\overline{T}$ par T^*, $PROCHE$ par $\mathcal{A}$, $TYPE$ par $TYPE_1$ et $TYPE_1$ par $TYPE_2$.

Nous considérons maintenant la relation de subsomption de types $\trianglelefteq$ étendue à T^*. Si un concept a un type, il hérite aussi de tous les types plus généraux dans $TYPE_2$.

Propriété 5.2.5 (Subsomption de type) $\forall\, t \in T^*,\ t' \in T^*,\ \forall\, C \in O_1 \cup O_2$
$\{[(C,t) \in TYPE_2 \wedge t \trianglelefteq t'] \Rightarrow (C,t') \in TYPE_2\}$

Propriétés des types

Les données avec lesquelles nous travaillons ne sont pas nécessairement bien typées, et les propriétés que nous avons définies risquent parfois de ne pas pouvoir s'appliquer. Nous définissons deux propriétés sur les types :

Intuitivement, nous dirons que $TYPE_1$ est voisin-cohérent pour tout couple de concepts (C,D) "proches", si les types de C et D ne sont pas les mêmes dans $TYPE$, alors il existe un type dans $TYPE_1$ qui les subsume. Formellement, $TYPE_1$ est voisin-cohérent s'il vérifie la propriété suivante :

Propriété 5.2.6 (Voisin-cohérence) $\forall\, (C,D) \in O_1 \cup O_2, \forall\, \tau, \tau' \in T$
$[((C,D) \in PROCHE \wedge (C,\tau) \in TYPE \wedge (D,\tau') \in TYPE \wedge \tau \neq \tau')$
$\Rightarrow \exists \tau'' \in \overline{T}\{(C,\tau'') \in TYPE_1 \wedge (D,\tau'') \in TYPE_1 \wedge \tau \trianglelefteq \tau'' \wedge \tau' \trianglelefteq \tau''\}]$

Intuitivement, nous dirons que $TYPE_2$ est ancre-cohérent si pour tout couple de concepts (C,D) formant une ancre, et si les types de C et D ne sont pas les mêmes dans $TYPE_1$ alors il existe un type dans $TYPE_2$ qui les subsume. Formellement, $TYPE_2$ est ancre-cohérent s'il vérifie la propriété suivante :

Propriété 5.2.7 (Ancre-cohérence) $\forall\, (C,D) \in \mathcal{A}, \forall\, \tau, \tau' \in \overline{T}$
$[((C,\tau) \in TYPE_1 \wedge (D,\tau') \in TYPE_1 \wedge \tau \neq \tau')$
$\Rightarrow \exists \tau'' \in T^*\{(C,\tau'') \in TYPE_2 \wedge (D,\tau'') \in TYPE_2 \wedge \tau \trianglelefteq \tau'' \wedge \tau' \trianglelefteq \tau''\}]$

Nous dirons qu'un typage à partir de $TYPE\_BASE$ est bien formé si et seulement si $TYPE_1$ est voisin-cohént et $TYPE_2$ est ancre-cohérent.

Utilisation des types pour restreindre le nombre de candidats au mapping

Si deux concepts n'ont pas de types en commun, ils ne peuvent pas être candidats au mapping et seront donc rangés dans $CANDIDAT\_EXCLU$.

Propriété 5.2.8 (Construction de l'ensemble des candidats exclus) $\forall\ C \in O_1, \forall\ D \in O_2$
$\{\nexists \tau \in T^*[(C,\tau) \in TYPE_2 \wedge (D,\tau) \in TYPE_2]$
$\Rightarrow (C,D) \in CANDIDAT\_EXCLU]\}$

Les ontologies biologiques peuvent contenir des concepts que nous exclurons du processus de mapping pour deux raisons. La première peut être qu'ils ne décrivent pas une fonction biologique précise. C'est le cas par exemple du concept "unknown function", qui n'est pas définissable formellement et pour lequel nous ne souhaitons pas trouver de correspondance. Nous pouvons aussi, deuxième raison, souhaiter exclure des concepts parce qu'ils ne couvrent pas le même domaine que le reste de l'ontologie. Par exemple, la hiérarchie Subtilist est dédiée à l'annotation de génomes d'organismes procaryotes, alors que la hiérarchie FunCat ne connaît pas une telle restriction. Une partie des concepts de FunCat ne pourront donc jamais être mis en correspondance avec un concept de Subtilist : les concepts concernant les mécanismes cellulaires exclusivement eucaryotes par exemple.

Nous définissons $CONCEPT\_EXCLU \subseteq O_1 \cup O_2$ l'ensemble de ces concepts.

Si un concept C appartient à cet ensemble aucun couple contenant C ne sera jamais candidat.

Propriété 5.2.9 $\forall C \in CONCEPT\_EXCLU$
$\forall\ D \in O_1 \cup O_2[((C \in O_1 \wedge D \in O_2) \vee (C \in O_2 \wedge D \in O_1)) \Rightarrow (C,D) \in CANDIDAT\_EXCLU]$

L'implémentation actuelle de O'Browser prend en compte le typage manuel (les types de $TYPE$), et les contraintes déduites de l'étape de validation des ancres pour l'établissement des candidats. L'utilisation des types étendus de $TYPE_1$ et $TYPE_2$ est l'un des aspects que nous souhaitons développer dans la prochaine version de O'Browser.

5.3 Matchers et combinaison de matchers

O'Browser utilise un ensemble de matchers pour évaluer la similarité entre deux concepts des deux hiérarchies. Chacun de ces matchers est dédié à la recherche d'un certain type de similarité entre les propriétés des concepts. Par exemple, le nom associé aux concepts peut être utilisé, ou les protéines annotées par ce concept, ou encore les domaines Pfam trouvés sur ces protéines.

Nous reprenons la définition d'un matcher de Euzenat et Shvaiko [34]

Définition 5.3.1 (Matcher) Formellement, un matcher M_i est défini comme une fonction qui, a deux ontologies O_1, O_2, un alignement d'entrée A, un ensemble de paramètres p et un ensemble de ressources et d'oracles r, associe un alignement (c'est-à-dire, un ensemble de correspondances) $A_i^m \subseteq O_1 \times O_2 \times \mathcal{R} \times I\!R$. Nous notons $(c, d, \rho_i, s^i_{c,d})$ une correspondance élément de A_i^m (voir définition 4.3.2).

$$M_i : (O_1, O_2, A, p, r) \mapsto A_i^m \subseteq O_1 \times O_2 \times \mathcal{R} \times I\!R$$

Pour simplifier la lecture, nous noterons parfois $M_i(C,D)$ la quantité $s^i_{c,d} \in I\!R$.

Nous utilisons dans O'Browser quatre types de matchers différents (les deux premiers sont basés sur les instances) : (1) matcher basé sur les relations d'homologie, (2) matcher basé sur les ressources externes, (3) matcher basé sur la syntaxe, et (4) matcher basé sur la structure. Ces

matchers utilisent une grande variété de techniques et exploitent les propriétés des concepts des ontologies de façon complémentaire. Nous détaillerons pour ces différents types de matchers les valeurs de A, p et r lorsqu'elles auront lieu d'être précisées.

5.3.1 Matcher homologie

Nous introduisons ici un matcher utilisant fortement l'information biologique à notre disposition.

L'évolution est *le* concept central de la biologie. Une conséquence de l'évolution est que tous les organismes présentent à divers degrés des liens de parenté dans l'arbre du vivant. Nous nous sommes intéressés au concept d'homologie. Deux protéines (ou gènes) présentes de nos jours dans un organisme sont dites homologues si elles descendent d'une protéine (ou d'un gène) commune ancestrale. Quand deux protéines partagent cette relation évolutive, elles partagent des propriétés communes. La propriété qui nous intéresse ici est le fait qu'elles puissent conserver la fonction de leur ancêtre ou avoir évolué vers des fonctions *proches*.

Le matcher *homologie* que nous avons développé utilise cette propriété. À la différence de la plupart des matchers basés sur les instances [59], les instances (ici les protéines) de nos concepts ne sont pas partagées. Nous n'utilisons pas non plus de phase d'apprentissage ou de test [112, 52, 30, 113]. Nous avons un ensemble $\text{ext}(C)$ de protéines appartenant à un organisme donné annotées avec le concept C de l'ontologie O_1, et un ensemble $\text{ext}(D)$ de protéines appartenant à un autre organisme, annotées avec le concept D de l'ontologie O_2. Nous avons déterminé l'homologie entre les protéines des ensembles $\text{ext}(C)$ et $\text{ext}(D)$ en utilisant BLAST [10] un programme classiquement utilisé pour ce genre de tâches en biologie. Nous notons une relation d'homologie entre deux protéines $e \in \text{ext}(C)$ et $f \in \text{ext}(D)$: $\text{homologous}(e, f)$.

Par rapport à la définition formelle du matcher, A l'alignement d'entrée est vide, de même que les paramètres p. Les ressources externes r sont les séquences des protéines, la mesure de similarité entre les instances (BLAST), le seuil déterminant l'homologie entre protéines, les types et l'ensemble $CANDIDAT\_EXCLU$.

Notre hypothèse est la suivante. Si deux concepts C et D sont totalement équivalents, nous nous attendons à ce que les protéines annotées par C dans le premier organisme soient *exclusivement* homologues aux protéines annotées par D dans le second, et réciproquement. En pratique, nous définissons le degré de *similarité* entre les concepts C et D comme :

$$\text{homologie}(C, D) = \frac{Cn_{total}}{Cn_{total} + Dn_{total}} * \frac{CDn_{concept}}{Cn_{relation}} + \frac{Dn_{total}}{Cn_{total} + Dn_{total}} * \frac{DCn_{concept}}{Dn_{relation}}$$

où :

– Le nombre de protéines annotées par C est :

$$Cn_{total} = |\,\text{ext}(C)|$$

– Le nombre de protéines annotées par C impliquées dans des relations d'homologie avec au moins une protéine annotée par au moins un concept de O_2 est :

$$Cn_{relation} = |\{e \in \text{ext}(C), \exists F \in O_2, \exists f \in \text{ext}(F), \text{homologous}(e, f)\}|$$

– Le nombre de protéines de C impliquées dans une relation d'homologie avec au moins une protéine de D est :

$$CDn_{concept} = |\{e \in \text{ext}(C), \exists d \in \text{ext}(D), \text{homologous}(e, d)\}|$$

Les termes Dn_{total},$Dn_{relation}$ et $DCn_{concept}$ sont définis de façon analogue pour le concept D.

En résumé, il s'agit d'un nouveau matcher qui utilise la similarité entre les instances pour l'homologie, pour évaluer la similarité entre les concepts. Il n'a besoin ni de phase d'apprentissage, ni de phase d'entraînement. Enfin il repose fortement sur les connaissances du domaine d'application, en utilisant d'une part les instances biologiques, ici les protéines, et une mesure de similarité entre protéines, BLAST. Cela rend les résultats de ce matcher à la fois dignes de confiance et compréhensibles pour les utilisateurs.

5.3.2 Matchers basés sur les propriétés partagées

La catégorie de matcher suivante utilise les propriétés partagées par les deux hiérarchies. Dans la référence [59], les auteurs présentent une approche basée sur des propriétés communes partagées, qui utilise la similarité *Kappa*. Nous utilisons une mesure de similarité plus simple mais suffisamment efficace pour nos données. Les propriétés biologiques que nous avons choisies proviennent de ressources externes variées et concernent les domaines Pfam, les termes GO ou les mots clés SwissProt, par exemple.

Par rapport à la définition formelle du matcher, A l'alignement d'entrée est vide, de même que les paramètres p. Les ressources externes r sont : les propriétés associées aux instances telles que les domaines Pfam, les termes de la Gene Ontology et les mots clés SwissProt, les types et l'ensemble $CANDIDAT\_EXCLU$.

La similarité entre deux concepts est alors évaluée en utilisant une distance de *Hamming* entre les ensembles de propriétés associées à chacun des deux concepts.

Soit prop(i) l'ensemble des valeurs des propriétés pour i associées à une instance i d'un concept pour une propriété biologique donnée (par exemple les domaines Pfam).

Soit C et D deux concepts de O_1 et O_2 respectivement, nous définissons p_1 et p_2 de la façon suivante :

$$p_1 = \{\text{prop}(e), e \in \text{ext}(C)\}$$

$$p_2 = \{\text{prop}(d), d \in \text{ext}(D)\}$$

$$\text{Hamming}(C, D) = 1 - \frac{|p_1 \cup p_2 - p_1 \cap p_2|}{|p_1 \cup p_2|}$$

5.3.3 Matcher syntaxique

La comparaison syntaxique de chaînes de caractères est très utile. Elle fonctionne souvent assez bien, mais ne permet pas de découvrir toutes les correspondances. Nous utilisons deux matchers conçus pour détecter deux types de similarités syntaxiques. Dans les deux cas, nous commençons par une préparation des données qui enlève les mots vides (mots non informatifs tels que "of", "from", ...). Nous avons fait le choix de garder les préfixes et les suffixes des termes en raison de leur importance en biologie.

Par rapport à la définition formelle du matcher, A l'alignement d'entrée est vide, de même que les paramètres p. Les ressources externes r sont : une liste des "mots vides", le typage et l'ensemble $CANDIDAT\_EXCLU$.

Le premier matcher utilise la distance de *Levenshtein* (distance d'édition).

Étant donné deux chaînes de caractères non vides s_1 et s_2, la similarité basée sur la distance de Levenshtein est évaluée par :

$$1 - \frac{\text{lev}(s_1, s_2)}{\max\{\text{len}(s_1), \text{len}(s_2)\}}$$

où $\text{lev}(s_1, s_2)$ est la distance de Levenshtein entre s_1 et s_2, et $\text{len}(s_1)$ (resp. $\text{len}(s_2)$) la longueur de s_1 (resp. s_2), et $\text{len}(s_1) + \text{len}(s_2) > 0$.

Le second matcher prend en compte la fréquence de chaque mot dans les labels des concepts des deux ontologies. Par exemple, le mot "Metabolism" est largement utilisé. Il peut avoir un impact très important sur la comparaison de "Metabolism of Carbon" et "Metabolism of Sulfur", alors que les deux termes ne sont pas sémantiquement équivalents. Pour cette raison, les mots sont pondérés par leur fréquence d'apparition dans les deux ontologies pour exprimer leur spécificité.

5.3.4 Matcher structural

Ce matcher est différent des trois autres types de matchers que nous avons vus dans cette section : il s'agit d'un modificateur de score. Par rapport à la définition formelle du matcher, A l'alignement d'entrée doit être non vide, de même que les paramètres p qui sont trois poids π_0, π_1 et ω. Les ressources externes r sont : les types et l'ensemble $CANDIDAT\_EXCLU$ et les deux ontologies $\mathcal{O}_1$ et $\mathcal{O}_2$ contenant les relations `is-a` entre les concepts.

En dernier, nous utilisons le matcher structural. Il n'est pas utilisé en parallèle des autres, mais il vient lisser les résultats de leur combinaison en prenant en compte l'influence des ancêtres et des descendants. Une idée communément admise en alignement d'ontologies est que plus les ancêtres et les descendants de deux concepts de deux ontologies différentes sont liés respectivement, plus les concepts eux-mêmes sont proches.

Le matcher que nous introduisons maintenant est spécialement conçu pour exploiter la structure arborescente des hiérarchies fonctionnelles. Dans un premier temps, le matcher procède par une approche *top-down* : l'influence des ancêtres est propagée aux descendants. Dans un second temps, c'est par une approche *bottom-up* que le matcher propage l'influence des descendants.

Formellement :
Définissons $\pi_0, \pi_1 \in [0, 1]$ tels que $\pi_0 + \pi_1 = 1$.
Soit $\text{sim}_0(C, D)$ la somme pondérée, en utilisant la pondération adaptative qui est introduite au paragraphe suivant, des scores des matchers pour une paire de concepts C et D, ou 1 si la paire de concepts est une ancre ($\text{sim}_0(C, D) = \sum_{i=1}^{k} W_{M_i}(M_i(C, D)) * M_i(C, D)$ si (C, D) n'est pas une ancre et $\text{sim}_0(C, D) = 1$ sinon).

Nous introduisons $\text{anc}(C)$ (resp. $\text{anc}(D)$) l'ensemble des concepts ancêtres de C (resp. D), sauf C (resp. D) lui-même.

Le matcher procède en deux temps. La première passe est évaluée en utilisant la formule suivante :

Nous définissons une mesure de similarité sim_1 qui étend sim_0 aux ancêtres.

$$\text{sim}_1(C, D) = \pi_0 \, \text{sim}_0(C, D) + \pi_1 \, \text{sim}_1(\text{anc}(C), \text{anc}(D))$$

$$\text{sim}_1(\text{anc}(C), \text{anc}(D)) = \frac{\sum_{A \in \text{anc}(C)} \sum_{B \in \text{anc}(D)} \omega^{k_1+k_2} \, \text{sim}_1(A, B)}{|\,\text{anc}(C)|\,|\,\text{anc}(D)|}$$

avec $k_1 = |\,\text{depth}(C) - \text{depth}(A)|$, $k_2 = |\,\text{depth}(D) - \text{depth}(B)|$, ω un poids $\in [0, 1]$ (plus les deux ancêtres sont distants de C et D, et moins leur rôle est important) et $\text{depth}(C)$ donne la profondeur du concept C dans l'ontologie.

Nous définissons ensuite de la même manière sim_2 à partir de sim_1 en étendant sim_1 aux descendants. La seconde passe est évaluée de façon similaire, avec sim_1 et sim_2 qui remplacent respectivement sim_0 et sim_1, et l'ensemble des ancêtres qui remplace l'ensemble des descendants.

Ce matcher est spécialement dédié aux ontologies qui sont des arbres. Dans le cas de DAG, on risque de propager à un noeud plusieurs fois l'influence de ses ancêtres, et de ne pas pouvoir garantir la stabilité du résultat.

[76, 35] proposent des approches pour des ontologies sous forme de graphes et propagent l'influence de l'ensemble des composants de l'ontologies.

Les scores pour chaque correspondance candidate au mapping seront transmis à O'Browser pour l'étape de sélection des correspondances.

Pour des raisons de facilité de notation, M_{struct} désigne le matcher structural et A^{struct} l'alignement produit en sortie de ce matcher.

5.3.5 Pondération adaptative

La prochaine étape consiste à combiner les résultats des différents matchers. Les approches classiques utilisent un facteur défini a priori par l'utilisateur pour les pondérer. Mais il peut très bien arriver que les bons résultats d'un matcher soient dégradés par les résultats d'un autre. Par exemple, supposons que le matcher homologie donne un bon score pour une paire de concepts, mais que les noms de ces concepts ne soient pas syntaxiquement proches. La combinaison des deux matchers pourrait être plus mauvaise que l'utilisation du seul matcher homologie.

Dans le but de résoudre ce problème, nous introduisons la notion de **pondération adaptative**. Nous pensons que la confiance attribuée à un matcher pour une paire de concept (C, D) doit dépendre en partie de son score pour (C, D). Par exemple, savoir que deux concepts ont des noms très proches est très informatif. De ce fait, une confiance de 1 (pour un maximum de 1) sera attribuée aux bons résultats du matcher syntaxique. Mais d'un autre côté, le fait que les noms de deux concepts ne soient pas syntaxiquement proches n'implique pas que les deux concepts soient éloignés. Nous attribuerons donc une valeur de confiance plus faible au matcher syntaxique pour ses résultats moyens ou faibles. Cette confiance faible nous permettra de donner un poids relatif plus fort aux bons résultats éventuellement obtenus par un autre matcher. Le poids associé au score d'une paire de concepts, donné par un matcher, est adaptatif dans le sens où il dépend de la valeur de ce score. La définition de la fonction de pondération est basée sur notre connaissance du matcher et de son domaine d'application.

Exemple :
Illustrons la notion de pondération adaptative sur deux exemples de matchers, l'un utilisé dans O'Browser et l'autre non.
Matcher syntaxique : si $M_{Syntactic}(C, D) = 1$, alors les deux concepts C et D sont très proches. Le rôle joué par le matcher syntaxique est alors très pertinent et son poids pour cette paire de concepts est de 1, alors que son poids est fixé à 0.15 pour des paires de concepts ayant un score inférieur à 0.5.
Matcher type de données : si $M_{Datatype}(C, D) = 1$, alors les deux concepts C et D ont le même type de données. Cela nous donne peu d'indices pour établir une correspondance. Le poids est fixé à 0.25. Si $M_{Datatype}(C, D) = 0$, alors les types de données sont fortement incompatibles, ce qui peut être une très bonne indication de l'impossibilité de correspondance des deux concepts. Le poids est alors fixé à 1.

Nous associons une fonction de pondération W_{M_i} à chaque matcher M_i (pondération adaptative).

Formellement :
Soit $\mathcal{M}$ l'ensemble des matchers M_i et soit $\mathcal{W}$ l'ensemble des fonctions de pondération adaptative W_{M_i} associées aux matchers M_i.

Définition 5.3.2 (Matcher) Une fonction de pondération associe à un matcher M_i et un score réel α, un réel s_w

$$W_{M_i} : I\!R \mapsto I\!R$$

$$W_{M_i}(\alpha) = s_w \in I\!R$$

La pondération associe à un ensemble d'alignements fournis par les matchers, un alignement pondéré A^p.

$$\{A_i^m\} \mapsto A^p$$

avec $A^p = \{(c, d, \rho, \sigma^p_{c,d})/\ \forall M_i \in \mathcal{M}\ \exists\ (c, d, \rho, s^i_{c,d}) \in A_i^m$ et

$$\sigma^p_{c,d} = \frac{\sum_{M_i \in \mathcal{M}} W_{M_i}(s^i_{c,d}) * s^i_{c,d}}{\sum_{M_i \in \mathcal{M}} W_{M_i}(s^i_{c,d})}\}$$

où $W_{M_i}(s^i_{c,d})$ est le poids qui est la valeur de la fonction de pondération adaptative pour le matcher M_i étant donné le score $s^i_{c,d}$ obtenu pour la paire (C, D) avec M_i.

Des propriétés peuvent être manquantes pour certains concepts ; pour cette raison des scores peuvent être manquants pour des matchers. Pour un couple de concepts donné, nous ne considérons lors de la pondération adaptative que les scores des matchers présents. C'est le cas par exemple avec le matcher utilisant les domaines Pfam, qui ne sont pas toujours associés aux protéines. Dans ce cas, le matcher est exclu de la pondération.

L'approche de [117] permet de donner plus d'importance aux meilleurs matchers (ou au plus mauvais) en utilisant une pondération moyenne ordonnée. La différence entre nos approches, est que nous pondérons le matcher en fonction d'une connaissance *a priori* du comportement du matcher alors que l'approche proposée dans [117] sélectionne toujours le même type de matcher.

5.3.6 Extension de la pondération adaptative

Nous souhaitons pouvoir utiliser les informations données par l'expert lors de l'étape de validation des ancres pour améliorer les fonctions de pondération des matchers, et ainsi mieux discriminer les correspondances des autres paires de concepts. Nous disposons pour cela de deux catégories de données : des ancres validées par l'expert qui sont des exemples de correspondances et les couples candidats refusés qui sont des contre-exemples de correspondances.

Notation Nous noterons $\mathcal{A}$ l'ensemble des ancres, et $\overline{\mathcal{A}}$ l'ensemble des couples de concepts candidats à l'établissement d'une ancre qui ont été refusés par l'expert.

Nous utilisons q matchers. Pour chaque matcher $M_i (1 \leq i \leq q)$, nous observons la distribution des $M_i(C, D)$ pour tous les couples (C, D) de $\mathcal{A} \cup \overline{\mathcal{A}}$. Nous déterminons des intervalles dans les résultats du matcher en regroupant les valeurs par paquets de partition de $[0, 1]$ en intervalles. Cette discrétisation est réalisée avec un algorithme de discrétisation basée sur l'analyse de la variance des groupes (nous pourrions utiliser d'autres algorithmes k-means par exemple). Pour chaque matcher M_i $(1 \leq i \leq q)$, n_i intervalles I_{ij} disjoints et recouvrant l'intervalle $[0, 1]$ sont identifiés $(1 \leq j \leq n_i)$. Dans un premier temps, $n_i \leq 5$ (par défaut O'Browser fixe $n = 3$) ou est laissé à la discrétion de l'utilisateur.

Pour déterminer les poids $W_{m_{ij}}$ optimaux à appliquer sur chacun de ces intervalles, nous proposons d'utiliser un algorithme génétique [49]. Chaque couple de concepts (C,D), exemple dans $\mathcal{A}$ ou contre-exemple dans $\overline{\mathcal{A}}$ est décrit par un vecteur numérique de taille $(\sum_{i=1}^{q} n_i) + 1$ de la façon suivante :
(C, D) : [classe (+1/-1) ; m_{11} ; m_{12} ; ... ; m_{1n_1} ;m_{21} ; m_{22} ; ... ; m_{2n_2} ;... ;m_{q1} ; m_{q2} ; ... ; m_{qn_q}] ;

Dans ce vecteur, m_{ij} représente la valeur sur le j-ième intervalle du i-ième matcher M_i pour (C, D). Pour un matcher M_i donné, une seule valeur m_{ij} est non nulle : $m_{ij} = M_i(C, D)$ si $M_i(C, D) \in I_{ij}$ et $m_{ij} = 0$ sinon et ce pour tout $i, 1 \leq i \leq q$, et classe est égale à +1 s'il s'agit d'un exemple ($(C, D) \in \mathcal{A}$) et -1 s'il s'agit d'un contre-exemple ($(C, D) \in \overline{\mathcal{A}}$).

L'algorithme génétique détermine pour chaque m_{ij} un poids $W_{m_{ij}}$ à appliquer pour optimiser la découverte des correspondances. De cette façon il est possible de reconstituer la fonction de pondération W_{M_i} en utilisant l'ensemble des $W_{m_{ij}}$ pour $1 \leq j \leq n_i$.

L'algorithme génétique permet d'optimiser un critère appelé "fonction de *fitness*". Ici nous souhaitons minimiser la somme des rangs des exemples, après classement des éléments de $\mathcal{A}$ et $\overline{\mathcal{A}}$ par ordre croissant en fonction de la somme des $M_i(C, D) * W_{M_i}(M_i(C, D))$. La meilleure combinaison de poids permettra dans le meilleur des cas de placer les exemples en tête du classement avant les contre-exemples, minimisant ainsi la somme de leurs rangs.

Après une initialisation aléatoire des poids, l'algorithme génétique procède itérativement. A chaque génération, l'algorithme va générer des poids, puis va calculer la fonction de *fitness* qui lui est associée. Les meilleurs individus (ici les jeux de poids générés) vont être conservés et croisés pour créer les poids de la génération suivante. Afin d'éviter de tomber dans des optima locaux, une fonction de mutation est appliquée à ces poids les modifiant légèrement. L'algorithme est relancé un nombre de fois défini par l'utilisateur, 500 fois par exemple (c'est la valeur par défaut dans O'Browser). C'est aussi l'utilisateur qui va déterminer le nombre d'individus à générer à chaque itération.

Les matchers utilisés pour découvrir les ancres ne sont pas pris en compte dans cette phase automatique.

[39] propose une approche basée sur la *Fuzzy Aggregation* pour apprendre automatiquement les valeurs des poids.

5.4 L'algorithme O'Browser

Dans cette section, nous présentons le fonctionnement de notre système. La figure 5.1 représente l'organisation et la succession des principales étapes de O'Browser.

Acquisition des données L'algorithme prend en entrée deux ontologies au format OWL ou en texte plat et les représente dans son format interne sous forme d'un DAG (ou sous forme d'un arbre). Les concepts sont liés aux données qu'ils annotent et qui sont présentes dans l'entrepôt de données Microbiogenomics.

L'algorithme utilise par ailleurs un certain nombre de paramètres : paramètres des matchers, fonctions de pondération des matchers, seuils de sélection des ancres.

Recherche des ancres O'Browser recherche les ancres en calculant toutes les similarités basées sur les noms de concepts et sur les relations d'homologies. Il ne sélectionne que les plus fortes (celles pour lesquelles la similarité est au-dessus d'un seuil donné en paramètre au système).

Identification des ancres et des types L'expert est ensuite sollicité pour (i) valider les ancres candidates d'une part et (ii) identifier les types d'autre part. Des interfaces graphiques dédiées l'aident dans ces deux tâches. Elles fournissent une représentation graphique des deux ontologies, et l'ensemble des instances annotées par les concepts concernés, aussi bien pour les ancres que pour les types.

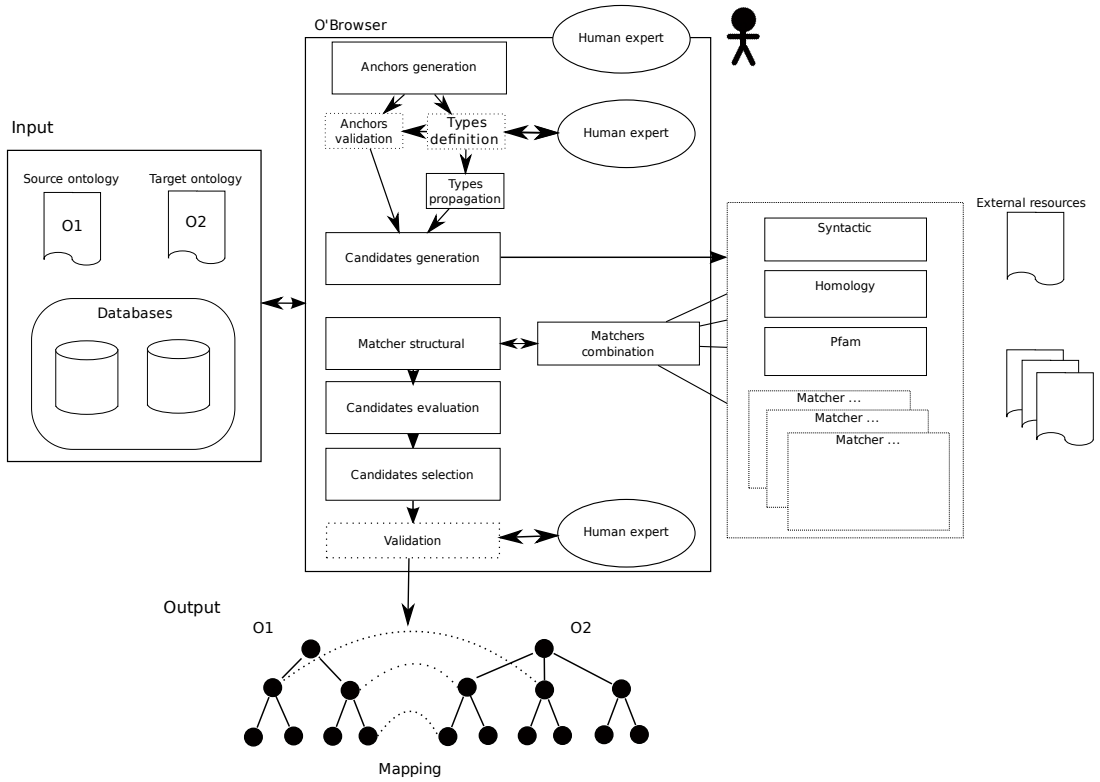
Extractions de contraintes O'Browser analyse les résultats des deux étapes semi-automatiques précédentes pour en extraire un maximum de contraintes d'exclusion.

A l'issue de cette étape, les couples de concepts candidats au mapping (appelés **couples candidats** par la suite) sont générés.

Évaluation des matchers Pour chaque couple candidat et pour chaque matcher, le système évalue la valeur du matcher en utilisant les propriétés des concepts. L'évaluation des matchers peut être effectuée parallèlement, puisque les matchers sont indépendants les uns des autres.

Combinaison des matchers Les matchers sont combinés en utilisant la stratégie de pondération adaptative. La valeur de chaque matcher pour chaque couple de concepts est modifiée par la fonction de pondération.

FIG. 5.1 – *Fonctionnement de O'Browser*



Matcher structural Les résultats de la combinaison des matchers sont pondérés par l'application du matcher structural qui les "lisse" en propageant l'influence des descendants et ancêtres.

Choix du mapping O'Browser sélectionne pour chaque concept de l'ontologie source O_1, le concept de l'ontologie cible qui lui est le plus proche, c'est-à-dire le concept de O_2 dans la correspondance qui a le score le plus élevé dans la liste des correspondances renvoyées par le matcher structural.

Évaluation des résultats Nous présenterons plus en détail cette étape dans la prochaine section.

5.4.1 Définition formelle

Nous définissons formellement l'ensemble des étapes présentées dans la section précédente. Le mapping est une fonction μ qui, à deux ontologies O_1 et O_2, un matcher structural M_{struct}, un ensemble de matchers non structuraux $\mathcal{M}$ et un ensemble de fonctions de pondération adaptative $\mathcal{W}$ associe un ensemble de correspondances A^f.

$$\mu : (O_1, O_2, M_{struct}, \mathcal{M}, \mathcal{W}) \mapsto A^f \subseteq O_1 \times O_2 \times \mathcal{R} \times I\!R$$

Détaillons l'étape de sélection des correspondances pour l'alignement final. A^f est l'ensemble des 4-uplets $(c, d, \rho, \sigma^f_{c,d})$ de $O_1 \times O_2 \times \mathcal{R} \times I\!R$ tels que pour c fixé dans O_1 on choisit dans O_2 le d associé à c dans A_{struct} qui a le meilleur score (si plusieurs d conviennent, on en choisit un au hasard) et on lui associe ce score $\sigma^f_{c,d}$.

Notons que nous sommes bien dans le cas d'un mapping (orienté) et non d'un alignement puisque :
si $(c, d, \rho, \sigma) \in \mu(O_1, O_2, M_{struct}, \mathcal{M}, \mathcal{W})$ alors nous n'avons pas nécessairement $(d, c, \rho, \sigma) \in \mu(O_2, O_1, M_{struct}, \mathcal{M}, \mathcal{W})$.

On peut résumer ces différentes étapes par le schéma ci-dessous :

$$\{A^m_i\} \overset{\text{pondération adaptative}}{\mapsto} A^p \overset{M_{struct}}{\mapsto} A^{struct} \overset{\text{sélection}}{\mapsto} A^f$$

5.4.2 Implémentation de O'Browser

O'Browser est entièrement implémenté et des copies d'écran de l'application sont disponibles en ligne http://www.lri.fr/~rance/obrowser. Les interfaces graphiques nécessaires aux étapes de validation des ancres et à l'identification des types sont implémentées en HTML/AJAX grâce à l'utilisation du Google Web Toolkit [1], et sont donc utilisables via n'importe quel navigateur web supportant le javascript.

Le coeur de O'Browser est implémenté en Java 1.6 et est lié à une base de données PostgreSQL, ainsi qu'à l'entrepôt de données Microbiogenomics. Des copies d'écrans sont visibles dans les annexes 5.6.2

[1] Google Web Toolkit http://code.google.com/intl/fr/webtoolkit/

O'Browser utilise un système de cache pour les étapes coûteuses, en stockant par exemple les résultats des matchers (dont la valeur pour un couple de concepts donné ne varie pas tant que les paramètres restent les mêmes d'une exécution à l'autre, et qu'aucune nouvelle donnée n'est ajoutée).

5.5 Validation des alignements

Le but de la validation de l'alignement trouvé est double : (a) d'une part évaluer la performance (précision, rappel, complexité, en temps...) de la méthode d'alignement et (b) d'autre part évaluer la qualité des résultats fournis.

Dans le premier cas, il faut connaître la solution de l'alignement pour pouvoir comparer les résultats obtenus avec les résultats théoriques, qui peuvent être donnés par un "Gold standard".

5.5.1 Validation avec "Gold Standard"

Dans le cas de l'alignement entre deux ontologies, si nous disposons d'un "gold standard" (un alignement validé), nous pouvons définir les quantités suivantes :

- MJ : le nombre de correspondances justes trouvées ;
- MR : le nombre de correspondances réellement présentes ;
- MT : le nombre de correspondances découvertes par la méthode d'alignement ;

La précision est définie par :

$$Pr = \frac{MJ}{MT}$$

Le rappel est défini par :

$$Ra = \frac{MJ}{MR}$$

A partir de ces deux valeurs, il est possible de déterminer la F-mesure :

$$F = \frac{2 \cdot (Pr \cdot Ra)}{(Pr + Ra)}$$

Ces mesures nous permettent d'évaluer la qualité globale de la méthode d'alignement. C'est le cas (a) décrit plus haut.

Des travaux récents [27] ont montré que d'autres mesures étaient plus adaptées à l'évaluation de ces valeurs (la précision et le rappel sémantique, et les Λ-*bounded precision measure* et Λ-*bounded recall measure* [27]) dans le cas de l'alignement d'ontologies.

Précision et rappel permettent d'évaluer une méthode pour laquelle nous possédons déjà les résultats. Souvent dans notre cas, nous n'aurons pas un tel sésame à notre disponition. Il nous faudra alors d'autres moyens pour mesurer la qualité des résultats.

5.5.2 Evaluation de la qualité des résultats sans "gold standard"

Sans *gold standard* nous ne pouvons évaluer la qualité d'un mapping qu'avec l'aide d'un expert. Nous proposons cependant une autre méthode qui permettra de mesurer la qualité des résultats du mapping. Avec ou sans "gold standard" les objectifs de la validation restent les mêmes : l'évaluation de la méthode, ou un indicateur de la qualité d'un résultat dans l'idée de pouvoir l'améliorer.

Expertise humaine

Dans certain cas, en particulier sur les ontologies de grande taille (par exemple issues du jeu de test *Anatomy* du challenge OAEI), les gold standard ne sont pas connus. L'évaluation des résultats de l'alignement est parfois faite manuellement [17], ou réalisée seulement sur une partie des résultats comme c'est le cas sur le track *anatomy* du challenge OAEI.

Nous considérons qu'il n'est alors pas possible de demander à un expert de valider l'ensemble des correspondances trouvées.

La difficulté de cette étape réside alors dans le choix des correspondances à soumettre à l'expert pour validation. Certaines techniques consistent à faire valider un ensemble correspondances choisies aléatoirement (100 par exemple) parmi l'ensemble des correspondances, pour constituer un *gold standard* partiel et se ramener à la situation connue. D'autres approches [22, 36] sont utilisables pour des ontologies formellement décrites, et utilisent des approches probabilistes ou logique de description floue.

Dans le but d'évaluer la qualité globale de l'alignement, cette approche est justifiée. Nous éliminerons de la validation les ancres qui ont déjà été expertisées, bien que cela crée un biais défavorable dans les résultats du mapping, puisque nous éliminons de bons résultats.

Nous sommes ici dans le cas (a) énoncé plus haut, où nous réalisons une évaluation globale de la méthode.

Si nous souhaitons améliorer l'alignement en itérant le processus (cas (b)), il faut demander à l'expert de valider les résultats les plus fragiles. Dans ce cas, nous pensons que le choix aléatoire n'est pas la meilleure stratégie à suivre, en particulier si le pourcentage de concepts couverts par la validation est faible.

Nous introduisons une nouvelle approche. Nous proposons de mesurer l'**ancrage** de chaque concept comme le rapport du nombre d'ancres présentes parmi ses ancêtres et descendants au nombre de ses ancêtres et descendants. C'est l'ancrage naïf.

Définition 5.5.1 (Ancrage naïf)

$$\text{Ancrage}_{\text{naif}}(C) = \frac{\#\text{ancres parmi les descendants et ancetres de } C}{\#\text{concepts ancetres ou descendants de } C}$$

Nous proposons une autre notion d'ancrage, plus élaborée, prenant en compte la distance du concept aux ancres de son contexte (ancêtres et descendants).

Définition 5.5.2 (Ancrage)

$$\text{Ancrage}(C,D) = \frac{\sum_{(A,B)\in((\text{anc}(C)\cup\text{desc}(C))\times(\text{anc}(D)\cup\text{desc}(D)))\cap\mathcal{A}} \omega^{|\,\text{level}(C)-\text{level}(A)|+|\,\text{level}(D)-\text{level}(B)|}}{|\,\text{anc}(C)\cup\text{desc}(D)|}$$

Dans cette formule $\mathcal{A} \subseteq (O_1 \times O_2)$ est l'ensemble des ancres, anc(C) (resp. desc(C)) est l'ensemble des ancêtres (resp. descendants) de C, C n'étant pas compris et $\omega \in [0,1]$ est un poids.

Les concepts avec les valeurs d'ancrage les plus faibles seront plus intéressants à valider et seraient donc soumis à l'expert.

Cette stratégie nous permettra éventuellement d'identifier des correspondances incorrectes et nous indiquera la nécessité de relancer une itération de O'Browser en changeant par exemple les fonctions de pondération pour améliorer la qualité des résultats.

Stratégie leave-one-out sur les matchers

Si les données le permettent, à chaque couple de concepts mis en correspondance sont associés les scores de plusieurs matchers (avant la combinaison). Nous proposons d'"éteindre" successivement les différents matchers et de mesurer à la i^e itération la stabilité des résultats. A chaque couple de concepts (C,D) mis en correspondance, on associera alors α_i, la somme des rangs obtenus pour D, après classement avec la combinaison des matchers M_j avec $j \neq i$.

Dans le cas (a) cette valeur permet à l'utilisateur de juger de l'influence d'un matcher unique sur le résultat (stabilité faible). Elle ne remplace néanmoins pas la valeur globale de score. Une valeur de score faible, associée à une forte stabilité sera plus intéressante qu'une valeur de score faible associée à une faible stabilité. En revanche une valeur de score forte, associée à une faible stabilité peut représenter l'influence de la pondération adaptative et n'est donc pas à remettre en cause, la stabilité n'étant alors qu'un indicateur.

Dans le cas (b), les valeurs faibles de stabilité ne seront pas forcément jugées fausses, mais permettront éventuellement de mettre en évidence un couple moins facile à valider qu'un autre. Les couples de concepts à faible stabilité peuvent alors être de meilleurs candidats pour une validation humaine que ceux à forte stabilité.

5.6 Exemples d'utilisation sur des hiérarchies fonctionnelles

5.6.1 Alignement de SubtiList et FunCat

Paramètres expérimentaux

Nous avons réalisé un premier test de notre système pour aligner deux hiérarchies fonctionnelles qui présentaient un intérêt pour nos collaborateurs : Subtilist et FunCat, version 2.1 (voir la section 4.1.2). Quelques caractéristiques des structures de ces deux hiérarchies sont données dans la table 4.1. Nous avons utilisé les données biologiques disponibles publiquement ainsi que les deux hiérarchies. Plus précisément, nous avons utilisé 3 génomes annotés manuellement avec Subtilist et 4 avec FunCat. Les données associées aux protéines proviennent de SwissPfam (Pfam

TAB. 5.1 – *Description des types dans les hiérarchies fonctionnelles*

Types	# concepts de Subtilist	# concepts de FunCat
Cell process	17	267
Cell rescue and defense	3	45
Information pathways	21	97
Metabolism	14	310
Transport / binding	8	115
Transposable element	2	7
Excluded from mapping	4	3
Not present in the other hierarchy	0	560

version 22.0), les termes de la Gene Ontology (voir 4.1.2) et les mots clés SwissProt de Uniprot (version 14.5). Toutes ces données ont été stockées dans l'entrepôt de données Microbiogenomics [66]. Finalement, nous avons utilisé un "gold standard" partiel, construit manuellement par un expert biologiste (41 des 63 concepts de Subtilist ont été mis en correspondance avec des concepts de FunCat).

Toutes les expériences ont été réalisées sur un processeur Intel Pentium 1,73 GHz avec 1 GB de RAM.

Résultats

Ancres : O'Browser a détecté 28 paires de concepts susceptibles d'être des ancres (4 en utilisant les relations d'homologie, 20 en utilisant une comparaison syntaxique et 4 en utilisant les deux mesures de similarité). L'expert a écarté 5 paires : dans deux cas les concepts n'étaient pas comparables, dans deux cas l'un des deux concepts était trop précis par rapport à l'autre, et dans le dernier cas les concepts étaient comparables mais n'étaient pas proches (le concept "subtilist :1.1 Cell Wall" enfant de "subtilist :1 Cell envelope and cellular processes" et le concept "funcat :42.01 Cell Wall" enfant de "funcat :42 BIOGENESIS OF CELLULAR COMPONENTS" sont clairement comparables, mais ne peuvent pas être considérés comme une ancre). Après cette étape de validation par l'expert, O'Browser a obtenu 23 ancres.

Il est à noter que 21 des 23 ancres sont localisées dans le même sous-arbre de Subtilist raciné par "subtilist :3 Information pathways". Cette branche est donc pratiquement exclue du processus de mapping, tandis les autres branches ne bénéficieront pas de données déjà validées.

Types : Parallèlement à l'étape de validation des ancres, l'expert a identifié 6 types de base (voir la table 5.1 pour plus d'information). Quelques concepts de O_1 et O_2 ont été exclus du processus parce que (i) ils ne faisaient pas référence à une fonction biologique précise (par exemple le concept "funcat :99 UNCLASSIFIED PROTEINS") ou (ii) ils n'étaient présents que dans l'une des deux ontologies.

En utilisant les types, les contraintes et les concepts exclus du mapping ($CONCEPT\_EXCLU$), nous avons déterminé $CANDIDAT\_EXCLU$ et réduit le nombre de paires candidates de 82 215

TAB. 5.2 – *Comparaison des résultats en utilisant différentes stratégies (résultats comparés aux gold standard de 41 correspondances)*

Strategie	# correspondances trouvées par O'Browser	# correspondances trouvées parmi les 5 meilleurs résultats par O'Browser
Ancres	23	23
Matcher syntaxique uniquement	**28**	33
Combinaison de matchers sans pondération	27	35
Combinaison de matchers pondération classique	29	35
Combinaison de matchers pondération adaptative	**33**	**38**

(le nombre de concepts de Subtilist multiplié par le nombre de concepts de FunCat) à moins de 12 000, ce qui fait un rapport intéressant de 7.

Paramètres des matchers : Nous avons utilisé les 6 matchers précédemment décrits. Le matcher homologie utilise le programme BLAST avec les paramètres suivants : les protéines avec au moins 60 % d'identité de résidus après alignement et une e-value d'au plus 10^{-4} ont été considérées comme étant homologues. Il faut noter que nous n'avons pas eu à évaluer à la volée le score de BLAST pour toutes les paires de protéines, puisque les résultats étaient déjà disponibles dans l'entrepôt de données Microbiogenomics.

Mapping obtenus : En utilisant O'Browser avec ces paramètres, nous avons obtenu les résultats suivants :
- Pour 80 % des concepts de Subtilist couverts par le **gold standard**, O'Browser a proposé comme résultat la solution.
- Pour 90 % des concepts de Subtilist couverts par le **gold standard**, O'Browser a proposé la solution parmi les 5 meilleures.

Une comparaison des résultats issus de différentes stratégies de combinaison des matchers est présentée dans la table 5.2.

5.6.2 Prédiction de type dans GO

Après cette première application, nous avons souhaité étendre l'utilisation de O'Browser à l'alignement de deux ontologies biologiques de taille plus importante. Nous avons choisi d'une part l'ontologie FunCat utilisée précédemment, et d'autre part Gene Ontology que nous avons présentée dans le chapitre 4.

Gene Ontology contient un nombre très élevé de concepts (plus de 29 000), il est donc impensable de tenter des approches dites "force brute" pour réaliser l'alignement. Le besoin d'utiliser le

TAB. 5.3 – *Prediction de type sur les termes de Gene Ontology*

Type	# concepts
cell process	1350
cell rescue defense	578
information pathways	3106
localization	1404
metabolism	1413
transport binding	1120
transposable elements	60

typage est donc encore plus frappant ici que pour l'alignement de Subtilist sur FunCat. Mais la taille de l'ontologie rend plus difficile l'étape de typage : les concepts les plus généraux de l'ontologie sont très éloignés des concepts les plus précis. Il n'est donc pas facile pour l'expert d'avoir une vue sur l'ensemble de l'ontologie et une assistance automatique est très utile.

A ce stade, les seules données utilisables dans un temps raisonnable sont les labels des concepts. Nous disposons grâce à l'alignement de Subtilist sur FunCat d'ensembles de concepts associés à des types. Nous avons utilisé une technique d'apprentissage pour assigner automatiquement un type au concept de Gene Ontology.

Type La première étape de la prédiction consiste en la création d'un vecteur de mots associé à chaque type. Les termes de Gene Ontology seront ensuite comparés en utilisant leur label aux vecteurs de chaque type. Le type le plus proche sera alors assigné au terme de type inconnu.

Nous avons testé différentes distances pour calculer la proximité entre les labels des termes et les vecteurs de mots associés à chaque type, et avons choisi d'utiliser une distance cosinus en donnant à chaque mot du vecteur un poids calculé en fonction de la fréquence du terme dans l'ensemble des concepts et sa spécificité ($TFIDF$ décrit ci-dessous).

Nous calculons la valeur du $TFIDF$ pour un mot w dans un type t. Nous utilisons deux quantités :
- $TF_{w,t}$ la fréquence du mot au sein d'un type. C'est-à-dire le nombre d'occurrences de w sur le nombre d'occurrences de l'ensemble des mots du type.
- IDF_w est le logarithme du rapport du nombre de types sur le nombre de types où le mot apparaît.

Pour chaque mot w dans un type t défini, on peut alors calculer la valeur du $TFIDF_{w,t} = TF_{w,t} * IDF_w$. La valeur du $TFIDF$ sera d'autant plus élevée que le mot est caractéristique du type.

Nous avons pu prédire des types pour 9031 concepts de Gene Ontology. Le détail est dans le tableau 5.3. Puis nous avons complété ce premier jeu de données en utilisant la règle dite du "true path" : si un concept possède un type alors ses descendants partagent eux aussi ce type. Les détails se trouvent dans le tableau 5.4.

Nous avons ensuite filtré les termes pour ne conserver que les termes provenant de sous-ontologies de GO compatibles avec le type. Par exemple, la sous-ontologie "Cellular Component" est dédiée à la localisation des produits des gènes et n'est donc pas compatible avec d'autres types que

TAB. 5.4 – *Extention des types sur les termes de Gene Ontology*

Type	# concepts	# concepts après "filtrage"
cell process	5997	5816
cell rescue defense	2779	2745
information pathways	17795	16071
localization	8006	2321
metabolism	6143	6120
transport binding	10813	10516
transposable elements	458	437

localization. Pour les autres types, nous nous sommes contentés d'éliminer les termes provenant de cette même sous-ontologie. Les résultats sont présentés dans le tableau 5.4.

Certains types apparaissent clairement comme contenant trop de concepts, c'est le cas de information pathways ou transport binding. Nous travaillons actuellement sur l'utilisation d'une mesure de similarité sémantique entre les termes de Gene Ontology pour affiner encore ces résultats avant de les faire valider par un expert.

Gold Standard A la différence de l'exemple précédent, nous ne disposons pas ici d'un "gold standard" réalisé pour nous par un expert, mais d'un alignement proposé sans garantie sur le site de Gene Ontology[2]. Il s'agit d'un gold standard partiel avec 840 correspondances proposées et 467 concepts de FunCat laissés sans correspondances dans Gene Ontology.

Ancres L'utilisation des données d'homologies n'est pas possible ici sur l'ensemble des candidats initiaux, du fait de leur trop grand nombre et du temps de calcul prohibitif que cela entrainerait. Nous n'avons donc recherché les ancres qu'en utilisant le critère de proximité syntaxique. O'Browser a détecté 316 ancres avec un seuil plus élevé que dans l'exemple précédent (avec le même seuil, on trouve 390 ancres). Sur 196 ancres comparables (présentes à la fois dans le gold standard et dans les ancres détectées par O'Browser), 148 aboutissent aux mêmes termes GO et 48 diffèrent. Les erreurs sont principalement dues à des homonymies dans GO entre la fonction moléculaire ou le processus biologique et la localisation cellulaire.

[2]http ://www.geneontology.org/external2go/mips2go

Conclusion et perspectives

Conclusion générale

Dans les chapitres précédents, nous avons traité différents problèmes liés à l'annotation fonctionnelle de protéines. Dans une première partie, nous avons cherché à établir des correspondances entre l'architecture en domaines de protéines et soit une propriété fonctionnelle, soit l'appartenance à une famille protéique. Ces recherches ont montré la nécessité de recourir à une standardisation pour l'annotation fonctionnelle, en utilisant des ontologies biologiques telles que la Gene Ontology ou les hiérarchies fonctionnelles. Mais les organismes que nous souhaitions étudier n'étaient pas tous annotés avec les mêmes classifications, ce qui rendait la tâche très difficile. Nous nous sommes donc intéressé dans un second temps au développement d'une méthode pour mettre en correspondance les concepts de ces différentes classifications.

Contributions

Dans le chapitre 3, nous avons introduit la notion de pépite séquentielle de connaissance et proposé un algorithme pour rechercher de telles pépites, SNK. Les pépites séquentielles de connaissance sont des règles d'association séquentielles, potentiellement peu souvent satisfaites par les données, et dont la qualité est garantie par l'utilisation d'une mesure de qualité. Nous avons introduit une formalisation de cette notion et prouvé la correction et la complétude de notre algorithme.

Dans un second temps, pour répondre aux demandes de nos collaborateurs, nous avons élargi la notion de pépites séquentielles de connaissance (SNoKs) à deux nouvelles notions : les pépites séquentielles contraintes (c-SNoKs) et les pépites séquentielles relâchées (s-SNoKs). Nous avons adapté l'algorithme SNK pour permettre la recherche de telles règles, et implémenté la méthode en Java. Afin de faciliter l'utilisation de l'outil SNK dans le cadre de l'étude des architectures en domaines des protéines, nous avons créé une application particulièrement adaptée à l'utilisation de données en provenance de la source de données de référence Pfam. Nous avons réalisé l'étude de deux familles protéiques en utilisant cette application. Enfin, pour faciliter le traitement des règles obtenues par notre outil SNK, nous avons introduit un visualisateur de domaines protéiques : DeeVee. Ces outils sont disponibles et utilisables en ligne à l'adresse http://www.lri.fr/~rance/SNK.

Au cours de ces recherches, il est apparu nécessaire d'utiliser les informations sur les fonctions des protéines. Mais les données dont nous disposions n'étaient pas annotées avec les mêmes hiérarchies fonctionnelles. Nous avons voulu permettre un usage simultané de ces données, annotées manuellement avec diverses ontologies biologiques, pour y rechercher des régularités. Cela nous a conduit à chercher des correspondances entre les composants de ces ontologies biologiques.

O'Browser (chapitre 5) est notre contribution au mapping d'ontologies biologiques. Nous avons développé une méthode basée sur une architecture classique de combinaison de matchers. Parmi les différents matchers que nous avons proposés, nous en avons introduit un qui utilise fortement les connaissances du domaine, puisqu'il est basé sur l'utilisation des relations d'homologie entre les protéines annotées par les ontologies biologiques à aligner, et qui est particulièrement bien adapté au problème d'équivalence entre concepts que nous avons défini.

Le nombre théorique de couples de concepts candidats au mapping peut être élevé. Il est apparu nécessaire de pouvoir en réduire le nombre et ainsi réduire le temps d'exécution de O'Browser. Nous avons pour cela introduit et formalisé la notion de typage des concepts des ontologies. Les

ontologies biologiques couvrent plusieurs champs fonctionnels, si bien que différents aspects de la fonction protéique sont souvent représentés dans une ontologie. Or deux concepts appartenant à deux champs fonctionnels distincts ne pourront jamais être mis en correspondance.

Enfin, nous avons observé que les bons résultats de certains matchers pouvaient être altérés par ceux d'autres matchers. Nous avons fait l'hypothèse que la confiance que peut accorder un utilisateur dans un matcher dépendait en partie de ses résultats. Nous avons donc proposé une nouvelle stratégie pour combiner les matchers, appelée pondération adaptative. Nous avons associé à chaque matcher une fonction de pondération, qui prend en compte la connaissance **a priori** de son comportement. Nous avons proposé d'automatiser la définition des fonctions de pondération en utilisant un algorithme génétique. Enfin l'utilisateur est au centre du système O'Browser, puisqu'il joue un rôle à deux niveaux de la méthode : au début avec une étape de validation d'ancres qui sont des correspondances facilement identifiables, puis à la fin lors de la validation des résultats.

Des copies d'écran de O'Browser sont consultables en ligne à l'adresse http://www.lri.fr/~rance/obrowser.

Perspectives

Les perspectives de ces travaux sont multiples.

Concernant la recherche de pépites séquentielles de connaissance (chapitre 3), le nombre de règles trouvées par SNK est potentiellement important et peut faire problème. L'analyse de ces règles est une tâche longue malgré l'aide du visualisateur DeeVee que nous avons conçu. Nous travaillons actuellement à l'utilisation d'une p-value, mesurant la surprise de trouver une règle pour une cible donnée, connaissant la famille dans laquelle cette règle a été trouvée. Cela nous permettra de classer les pépites séquentielles de connaissance trouvées, et de ne proposer à l'utilisateur que les règles inattendues, probablement plus difficiles à détecter que les autres et plus intéressantes pour le biologiste.

D'autre part, les pépites séquentielles de connaissance se sont avérées pertinentes dans les cas que nous avons étudiés, mais il pourrait être très intéressant de chercher des règles mixtes mêlant séquentialité et non-séquentialité. Nous pourrions alors prendre en compte non seulement des séquences protéiques, mais aussi des informations supplémentaires, comme la localisation cellulaire, la présence de signaux spécifiques... Cela nous permettrait de trouver des règles plus précises décrivant des fonctions ou des familles protéiques.

Les perspectives autour de O'Browser sont nombreuses. Nous souhaitons poursuivre la recherche d'autres matchers utilisant fortement les connaissances du domaine, qui sont facilement interprétables et compréhensibles par l'utilisateur. Pour cela les travaux sur les distances sémantiques entre les termes de la Gene Ontology [24] pourraient permettre d'établir une distance entre concepts. Nous utiliserions à cette fin les termes de la Gene Ontology associés aux protéines annotées par les ontologies que nous souhaiterions aligner. D'autres approches utilisant des ressources externes dédiées au domaine biomédical ou à celui de la biologie, telles que l'UMLS, ont été développées et il pourrait être pertinent de les intégrer à O'Browser. Nous avons formalisé la découverte de types à partir des relations d'homologie entre les protéines annotées par les concepts d'une même ontologie d'une part, et en utilisant les ancres d'autre part. Nous souhaitons implémenter la découverte de ces types dans la prochaine version de O'Browser.

Pour l'instant O'Browser est conçu pour détecter l'équivalence entre deux concepts d'ontologies différentes. Deux pistes s'ouvrent à nous pour étendre O'Browser. La première concerne la détection de relations de subsomption, détectables au moins à deux niveaux, (1) au niveau des noms des concepts, où l'on peut imaginer que le nom du concept le plus général sera une sous-chaîne du nom du concept le plus spécialisé, et (2) au niveau des instances, où les relations d'homologies devraient permettre d'établir que l'extension du concept le plus spécifique est un sous-ensemble de l'extension du concept le plus général. La seconde piste concerne le nombre de concepts impliqués dans une correspondance. Nous avons noté que les diverses ontologies biologiques n'avaient pas la même granularité, sans pour l'instant le prendre en compte. Il serait judicieux de pouvoir rechercher des relations non plus entre deux concepts (relation 1-1 localement) mais des relations d'équivalence (ou de subsomption) entre un concept et un ensemble de concepts (relation 1-n). Nous pourrions réaliser cela de différentes manières. Pour chaque concept de la première ontologie nous disposons de la liste des concepts de la deuxième ontologie classés par similarité décroissante donnée par O'Browser. Nous pourrions sélectionner les n meilleurs, ou les concepts dont la similarité serait située au-dessus d'un certain seuil. Une autre possibilité serait de modifier les matchers afin de mesurer pour une propriété donnée la similarité d'un concept pour tout groupe de concepts. Certaines approches naïves pourraient être envisagées, comme par exemple comparer les instances du premier concept avec l'union des instances des concepts du groupe. Il est cependant à prévoir que de telles approches risquent d'élever considérablement le nombre de candidats au mapping. Il faudrait alors trouver des stratégies supplémentaires, ou des heuristiques pour diminuer le nombre de candidats au mapping.

La différence de granularité entre les ontologies est parfois telle que l'alignement conduit à une importante perte d'informations lors du passage de l'ontologie la plus grande vers la plus petite. Nous pourrions, non plus aligner la plus grande ontologie et la plus petite, mais essayer d'enrichir la plus petite par comparaison avec la plus grande. Pour cela nous suggérons de recourir à la méthode d'Analyse de Concepts Formels. A chaque concept des ontologies peuvent être associées un grand nombre de propriétés : domaines Pfam, termes de la Gene Ontology, mots clés SwissProt... Nous avons utilisé ces données dans les matchers, mais nous pourrions, pour un concept donné et l'ensemble des instances qu'il annote, essayer de trouver des ensembles de protéines décrites par le même ensemble de propriétés. Ce couple (ensemble de protéines, propriétés communes aux protéines) nous permettrait de décrire un nouveau concept, qui pourrait éventuellement être ajouté à l'ontologie la plus petite.

Table des figures

Annexes

Pépites de connaissance

```
Règles de longueur min 1 :

RnaseH, ->  RVT_1_fam, (0.39,1.00)
Integrase_Zn, ->  RVT_1_fam, (0.29,1.00)
RVTthumb, ->  RVT_1_fam, (0.29,1.00)
Integrase, ->  RVT_1_fam, (0.28,1.00)
RVTconnect, ->  RVT_1_fam, (0.27,1.00)
Gag_p24, ->  RVT_1_fam, (0.03,1.00)
Gag_p17, ->  RVT_1_fam, (0.03,1.00)
RVP, ->  RVT_1_fam, (0.29,1.00)
RVP_2, ->  RVT_1_fam, (0.10,0.99)
RVT,rve, ->  RVT_1_fam, (0.70,0.99)
Chromo, ->  RVT_1_fam, (0.11,0.99)
rve,RVT, ->  RVT_2_fam, (0.30,0.98)
zf-CCHC,zf-CCHC, ->  RVT_1_fam, (0.03,0.97)
Retrotrans_gag, ->  RVT_1_fam, (0.21,0.91)

Règles de longueur min 2 :

RVT,RnaseH, ->  RVT_1_fam, (0.39,1.00)
RnaseH,rve, ->  RVT_1_fam, (0.39,1.00)
Integrase_Zn,rve, ->  RVT_1_fam, (0.29,1.00)
RVT,Integrase_Zn, ->  RVT_1_fam, (0.29,1.00)
RnaseH,Integrase_Zn, ->  RVT_1_fam, (0.29,1.00)
RVP,RnaseH, ->  RVT_1_fam, (0.29,1.00)
RVT,RVTthumb, ->  RVT_1_fam, (0.29,1.00)
RVTthumb,rve, ->  RVT_1_fam, (0.29,1.00)
RVTthumb,RnaseH, ->  RVT_1_fam, (0.29,1.00)
RVTthumb,Integrase_Zn, ->  RVT_1_fam, (0.28,1.00)
RVT,Integrase, ->  RVT_1_fam, (0.28,1.00)
RnaseH,Integrase, ->  RVT_1_fam, (0.28,1.00)
rve,Integrase, ->  RVT_1_fam, (0.28,1.00)
Integrase_Zn,Integrase, ->  RVT_1_fam, (0.28,1.00)
RVTthumb,Integrase, ->  RVT_1_fam, (0.28,1.00)
RVP,Integrase_Zn, ->  RVT_1_fam, (0.27,1.00)
RVP,RVTthumb, ->  RVT_1_fam, (0.27,1.00)
RVT,RVTconnect, ->  RVT_1_fam, (0.27,1.00)
RVTconnect,Integrase_Zn, ->  RVT_1_fam, (0.27,1.00)
RVTconnect,RnaseH, ->  RVT_1_fam, (0.27,1.00)
RVTconnect,rve, ->  RVT_1_fam, (0.27,1.00)
RVTthumb,RVTconnect, ->  RVT_1_fam, (0.27,1.00)
RVP,Integrase, ->  RVT_1_fam, (0.27,1.00)
RVP,RVTconnect, ->  RVT_1_fam, (0.27,1.00)
RVTconnect,Integrase, ->  RVT_1_fam, (0.27,1.00)
```

```
Retrotrans_gag,RnaseH, ->  RVT_1_fam, (0.05,1.00)
zf-CCHC,RnaseH, ->  RVT_1_fam, (0.04,1.00)
zf-CCHC,Integrase_Zn, ->  RVT_1_fam, (0.03,1.00)
zf-CCHC,Integrase, ->  RVT_1_fam, (0.03,1.00)
zf-CCHC,RVTthumb, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,RnaseH, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,rve, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,RVP, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,RVT, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,zf-CCHC, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,Integrase_Zn, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,Integrase, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,RVTthumb, ->  RVT_1_fam, (0.03,1.00)
zf-CCHC,RVTconnect, ->  RVT_1_fam, (0.03,1.00)
Gag_p24,RVTconnect, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,Gag_p24, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,Integrase_Zn, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,RVP, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,RVT, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,RVTconnect, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,RVTthumb, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,RnaseH, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,rve, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,zf-CCHC, ->  RVT_1_fam, (0.03,1.00)
Gag_p17,Integrase, ->  RVT_1_fam, (0.03,1.00)
RVP,RVT, ->  RVT_1_fam, (0.29,1.00)
RVP,rve, ->  RVT_1_fam, (0.29,1.00)
RVP_2,Chromo, ->  RVT_1_fam, (0.08,1.00)
zf-CCHC,Chromo, ->  RVT_1_fam, (0.07,1.00)
Retrotrans_gag,RVP_2, ->  RVT_1_fam, (0.09,1.00)
rve,Chromo, ->  RVT_1_fam, (0.11,0.99)
zf-CCHC,RVP_2, ->  RVT_1_fam, (0.08,0.99)
RVP_2,RVT, ->  RVT_1_fam, (0.10,0.99)
RVP_2,rve, ->  RVT_1_fam, (0.10,0.99)
RVT,rve, ->  RVT_1_fam, (0.70,0.99)
RVT,Chromo, ->  RVT_1_fam, (0.11,0.99)
Retrotrans_gag,Chromo, ->  RVT_1_fam, (0.08,0.99)
zf-CCHC,RVP, ->  RVT_1_fam, (0.04,0.99)
rve,RVT, ->  RVT_2_fam, (0.30,0.98)
zf-CCHC,zf-CCHC, ->  RVT_1_fam, (0.03,0.97)
Retrotrans_gag,zf-CCHC, ->  RVT_1_fam, (0.08,0.95)
Retrotrans_gag,rve, ->  RVT_1_fam, (0.21,0.91)
Retrotrans_gag,RVT, ->  RVT_1_fam, (0.21,0.91)
```

Copies d'écran de O'Browser

FIG. 5.2 – *Validation des ancres*

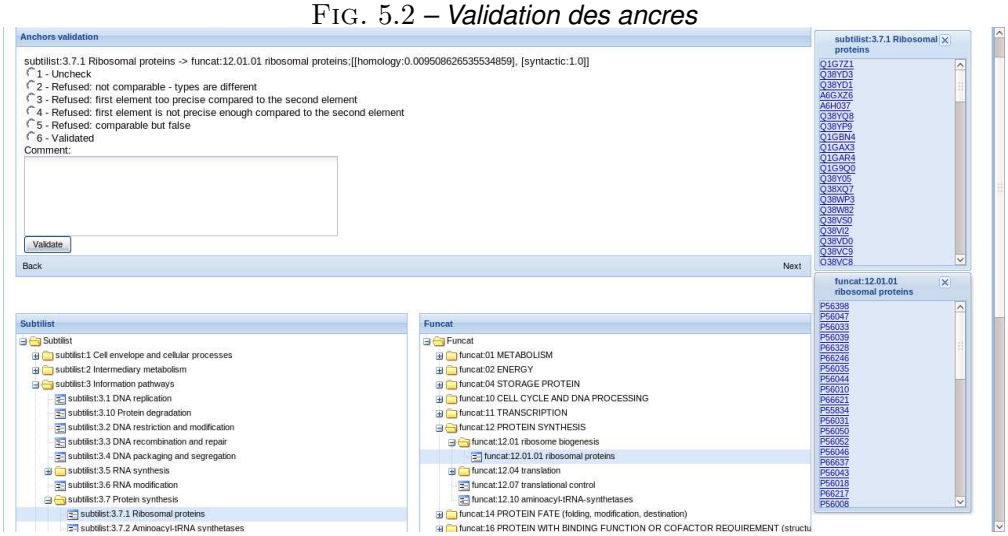


FIG. 5.3 – *Définitions des types*

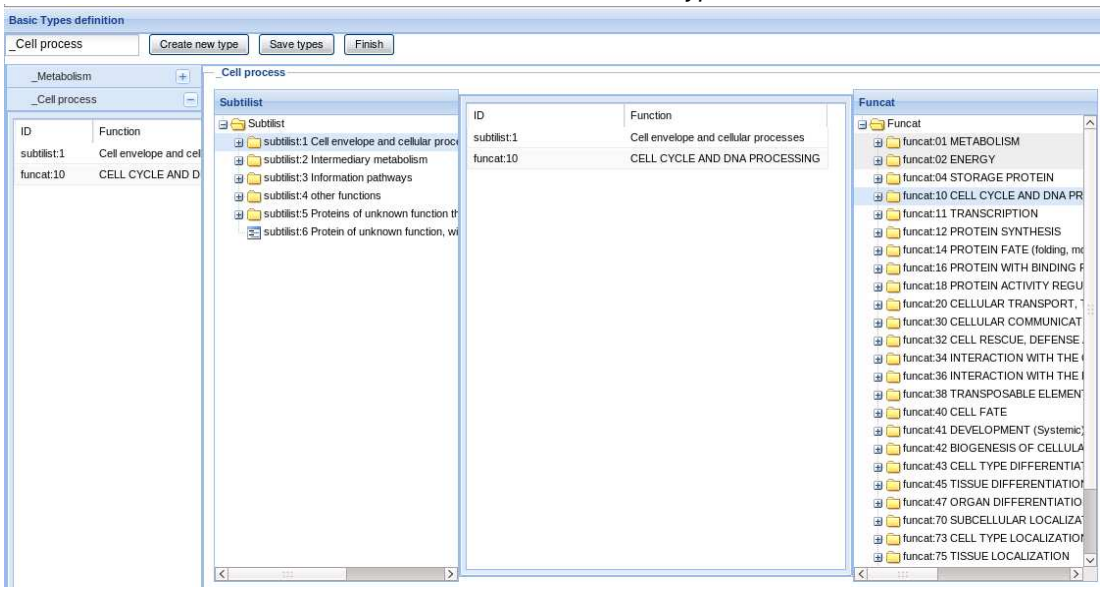


FIG. 5.4 – *Définition d'une fonction de pondération*

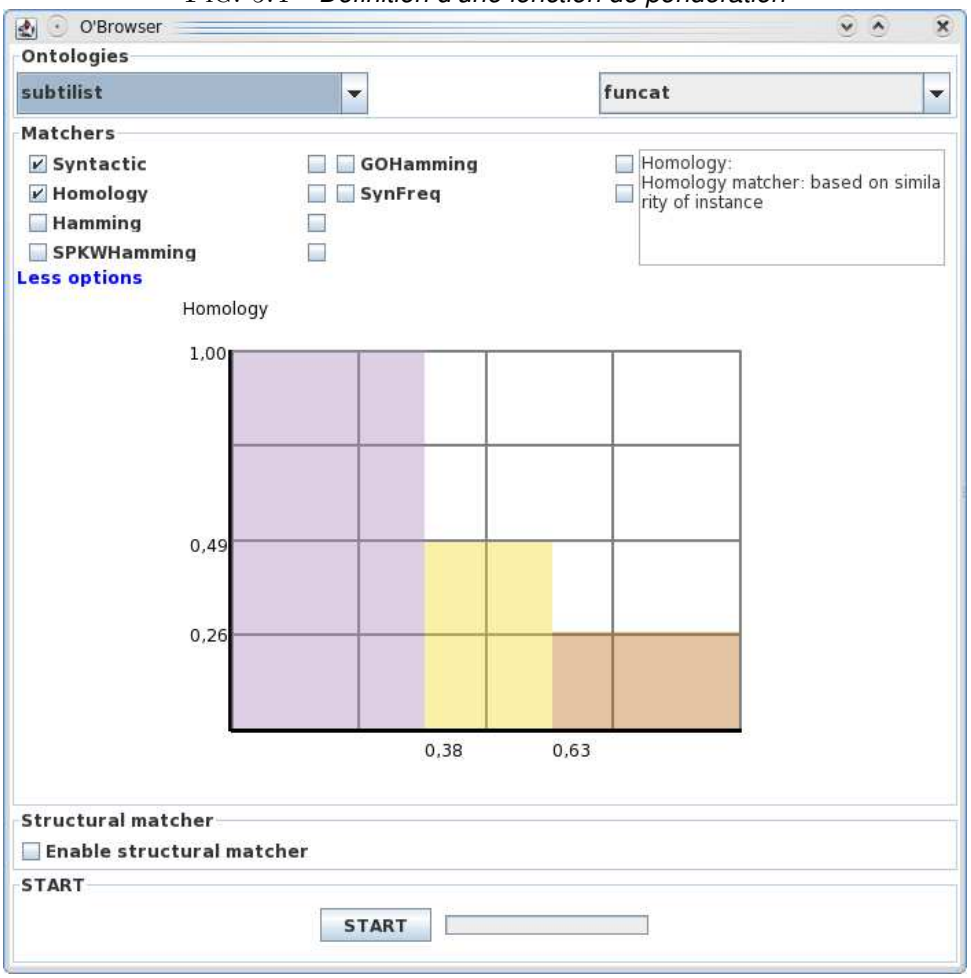
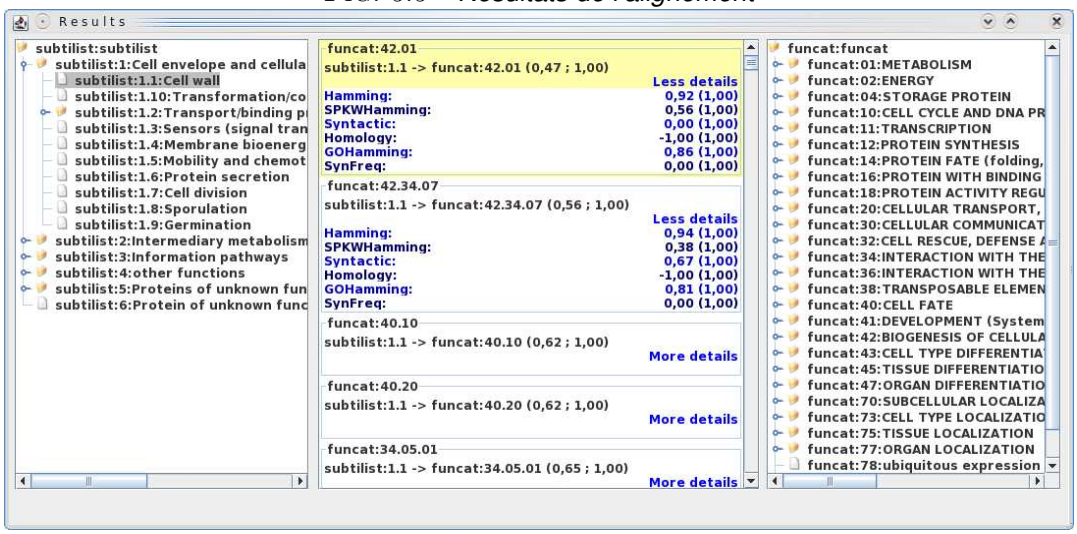


FIG. 5.5 – *Résultats de l'alignement*



Bibliographie

[1] http ://www.genome.jp/kegg/. 78

[2] http ://www.nlm.nih.gov/pubs/factsheets/umlslex.html. 78

[3] http ://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html. 78

[4] http ://www.nlm.nih.gov/pubs/factsheets/umlssemn.html. 78

[5] Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB). 77

[6] R. Agrawal and R. Srikant. Fast algorithm for mining association rules. In *VLDB Conf. Very Large Data Bases*, pages 487–499, 1994. 29

[7] Rakesh Agrawal, T. Imielienski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993. 31, 33

[8] Rakesh Agrawal and Ramakrishnan Skikant. Mining sequential patterns. *ICDE*, March 1995. 31, 34, 35

[9] Stuart Aitken, Yin Chen, and Jonathan Bard. Obo explorer : an editor for open biomedical ontologies in owl. *Bioinformatics*, 24(3) :443–444, Feb 2008. 81

[10] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215 :403–410, 1990. 21, 103

[11] J. Azé, L. Gentils, C. Toffano-Nioche, V. Loux, J-F Gibrat, P. Bessières, C. Rouveirol, A. Poupon, and C. Froidevaux. Towards a semi-automatic functional annotation tool based on decision tree techniques. *BMC Proceedings, International Workshop on Machine Learning in Systems Biology, MLSB'07*, 2(4), 2007. 23

[12] Jérôme Azé. Une nouvelle mesure de qualité pour l'extraction de pépites de connaissances. In *Revue RIA-ECA numéro spécial EGC03*, volume 17, pages 171–182, 2003. 40

[13] A. Bairoch, R. Apweiler, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O'Donovan, N. Redaschi, and L.S. Yeh. The universal protein resource (uniprot). *Nucleic Acids Res.*, 33 :D154–159, 2005. 59, 77

[14] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L. Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The pfam protein families database. *Nucleic Acids Research*, 32 :D138–D141, 2004. 9, 24, 60

[15] Olivier Bodenreider. The Unified Medical Language System (UMLS) : integrating biomedical terminology. *Nucleic Acids Res*, 32(Database issue) :D267–D270, Jan 2004. 78

[16] Olivier Bodenreider, M. Aubry, and A. Burgun. Non-lexical approaches to identifying associative relations in the gene ontology. In *Pacific Symposium on Biocomputing*, pages 104–115, 2005. 76

[17] Olivier Bodenreider, Terry F Hayamizu, Martin Ringwald, Sherri De Coronado, and Songmao Zhang. Of mice and men : aligning mouse and human anatomies. *AMIA Annu Symp Proc*, pages 61–65, 2005. 113

[18] Christian Borgelt. Efficient implementations of apriori and eclat. In *Workshop of Frequent Item Set Mining Implementations*, 2003. 33

[19] Christian Borgelt. An implementation of the fp-growth algorithm. In *Workshop Open Source Data Mining Software*, 2005. 33

[20] Sergei Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market data. In *ACM SIGMOD International Conference on Management of Data*, 1997. 33

[21] K. Bryson, V. Loux, R. Bossy, P. Nicolas, S. Chaillou, M. van de Guchte, S. Penaud, E. Maguin, M. Hoebeke, P. Bessières, and J-F. Gibrat. Agmial : implementing an annotation strategy for prokaryote genomes as a distributed system. *Nucleic Acids Res*, 34(12) :3533–3545, 2006. 24

[22] Silvana Castano, Alfio Ferrara, Davide Lorusso, Tobias Henrik Näth, and Ralf Möller. Mapping validation by probabilistic reasoning. In *5th European Semantic Web Conference (ESWC2008)*, pages 170–184, June 2008. 113

[23] Chris Clifton, Ed Hausman, and Arnon Rosenthal. Experience with a combined approach to attribute matching across heterogeneous databases. In *Proc. 7th IFIP Conference on Database Semantics*, pages 428–453, Leysin (CH), 1997. 88

[24] Francisco M. Couto, Mário J. Silva, and Pedro M. Coutinho. Measuring semantic similarity between gene ontology terms. *Data Knowl. Eng.*, 61(1) :137–152, 2007. 122

[25] F. H. Crick, L. Barnett, S. Brenner, and R. J. Watts-Tobin. General nature of the genetic code for proteins. *Nature*, 192 :1227–1232, Dec 1961. 15

[26] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 1859. 19

[27] Jérôme David and Jérôme Euzenat. On fixing semantic alignment evaluation measures. In *Third International Workshop On Ontology Matching (OM2008)*, October 2008. 112

[28] Robin Dhamankar, Yoonkyong Lee, An-Hai Doan, Alon Halevy, and Pedro Domingos. iMAP : Discovering complex semantic matches between database schemas. In *Proc. 23rd International Conference on Management of Data (SIGMOD)*, pages 383–394, Paris (FR), 2004. 91

[29] Hong-Hai Do and Erhard Rahm. COMA - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002. 89

[30] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. In *The VLDB Journal*, volume 12, pages 303–319, 2003. 103

[31] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontologies matching : a machine learning approach. *Handbook on ontologies*, pages 385–404, 2004. 90

[32] Anhai Doan, Pedro Domingos, and Alon Levy. Learning source descriptions for data integration. pages 81–86, 2000. 90

[33] Karen Eilbeck, Suzanna E Lewis, Christopher J Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The sequence ontology : a tool for the unification of genome annotations. *Genome Biol*, 6(5) :R44, 2005. 79

[34] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007. 11, 82, 83, 85, 96, 102

[35] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proc. 16th European Conference on Artificial Intelligence (ECAI)*, pages 333–337, Valencia (ES), 2004. 92, 106

[36] Alfio Ferrara, Davide Lorusso, Giorgos B. Stamou, Giorgos Stoilos, Vassilis Tzouvaras, and Tassos Venetis. Resolution of conflicts among ontology mappings : a fuzzy approach. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. 113

[37] K. Forslund and E.L.L. Sonnhammer. Predicting protein function from domain content. *Bioinformatics*, 24 :1681–1687, 2008. 23

[38] Christine Froidevaux, Frédérique Lisacek, and Bastien Rance. Extracting sequential nuggets of knowledge. In *International Conference on Database and Expert Systems Applications, DEXA'07*, volume LNCS 4653, pages 740–750, September 2007. 42

[39] Avigdor Gal, Giovanni Modica, Hassan Jamil, and Ami Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1) :21–32, 2005. 109

[40] B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. 86

[41] L. Geng and H.J. Hamilton. Interstingness measures for data mining : a survey. *ACM Computing Surveys*, 38(3) :9, 2006. 38, 39

[42] A. Ghazvinian, N. F. Noy, and M. A. Musen. Creating mappings for ontologies in biomedicine : Simple methods work. Technical report, Stanford Center for Biomedical Informatics Research, 2009. 93

[43] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. In *Proc. IJCAI Workshop on Ontologies and Distributed Systems*, pages 139–146, Acapulco (MX), 2003. 90

[44] Christine Golbreich, Matthew Horridge, Ian Horrocks, Boris Motik, and Rob Shearer. Obo and owl : Leveraging semantic web technologies for the life sciences. In *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, pages 169–182, November 2007. 81

[45] T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human Computer Studies*, 43 :907–928, 1995. 71

[46] J. Han, J. Pei, B. Mortazavi-asl, Q. Chen, U. Dayal, and M.C. Hsu. Freespan : frequent pattern-projected sequential pattern mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 355–359, 2000. 34

[47] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD International Conference on management of Data*, 2000. 33

[48] Md. Seddiqui Hanif and Masaki Aono. Alignment results of anchor-flood algorithm for oaei-2008. In *OM*, 2008. 97

[49] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992. 108

[50] W. Hu, Qu Yuzhong, and Cheng Gong. Matching large ontologies : A divide-and-conquer approach. *Data Knowl. Eng.*, 67(1) :140–160, 2008. 92, 97, 98

[51] N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, B. Cuche, E. De Castro, C. Lachaize, P.S. Langendijk-Genevaux, and C.J.A. Sigrist. The 20 years of prosite. *Nucleic Acids Res.*, 36 :D245–9, 2007. 24

[52] Ryutaro Ichise, Hiedeaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance-based learning. In *In : Proceedings of the eighteenth International Joint Conference on Artificial Intelligence. (2003*, pages 22–30, 2003. 86, 103

[53] ISWC'08 Third International Workshop on Ontology Matching, October 26 2008. Karlsruhe. 11

[54] Y. Kalfoglou and M. Schorlemmer. Ontology mapping : The state of the art. In *Dagstuhl Seminar on Semantic Interoperability and Integration*, volume 04391, 2005. 11

[55] Yannis Kalfoglou and Marco Schorlemmer. IF-Map : an ontology mapping method based on information flow theory. *Journal on Data Semantics*, I :98–127, 2003. 92

[56] M. Kanehisa and S. Goto. Kegg : Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 28(1) :27–30, Jan 2000. 78

[57] P. D. Karp, M. Riley, S. M. Paley, and A. Pelligrini-Toole. Ecocyc : an encyclopedia of escherichia coli genes and metabolism. *Nucleic Acids Res*, 24(1) :32–39, Jan 1996. 73

[58] Peter D Karp, Monica Riley, Suzanne M Paley, and Alida Pellegrini-Toole. The metacyc database. *Nucleic Acids Res*, 30(1) :59–61, Jan 2002. 73

[59] T. Kirsten, Andreas Thor, and Erhard Rahm. Instance-based matching of large life science ontologies. In *Data Integration in the Life Sciences DILS*, pages 172–187, 2007. 86, 103, 104

[60] Martin Lacher and Georg Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proc. 14th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 305–309, Key West (FL US), 2001. 91

[61] Stéphane Lallich, Olivier Teytaud, and Elie Prudhomme. Association rule interestingness : Measure and statistical validation. In *Quality Measures in Data Mining*, pages 251–275. 2007. 39

[62] Stéphane Lallich and Olivier Teytaud. évaluation et validation de l'intérêt des règles d'association. *Revue des Nouvelles Technologies de l'Information*, 2, 2003. 39

[63] P Lambrix, H Tan, V Jakoniene, and L Strömbäck. *Semantic Web : Revolutionizing Knowledge Discovery in the Life Sciences*, chapter Biological ontologies, pages 85–99. Springer, 2007. 72

[64] J. Lamoril, N. Ameziane, J.-C. Deybach, P. Bouizegarène, and M. Bogard. Les tehcniques de séquençage de l'adn : une révolution en marche. première partie. *Immuno-analyse et biologie spécialisée*, 23 :260–279, 2008. 18

[65] N. Lavrac, P. Flach, and B. Zupan. Rule evaluation measures : A unifying view. In S. Dzeroski and P. Flach, editors, *Ninth International Workshop on Inductive Logic Programming (ILP'99)*, pages 174–185. Springer-Verlag, June 1999. 39

[66] F. Lemoine, B. Labedan, and Ch. Froidevaux. GenoQuery : a new querying module for functional annotation in a genomic warehouse. *Bioinformatics*, 24 :322–329, 2008. 25, 94, 115

[67] Olivier Lespinet and Bernard Labedan. Orphan enzymes ? *Science*, 307(5706) :42, Jan 2005. 77

[68] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady akademii nauk SSSR*, 163(4) :845–848, 1965. In Russian. English Translation in Soviet Physics Doklady, 10(8) p. 707–710, 1966. 84

[69] Tania Lima, Andrea H Auchincloss, Elisabeth Coudert, Guillaume Keller, Karine Michoud, Catherine Rivoire, Virginie Bulliard, Edouard de Castro, Corinne Lachaize, Delphine Baratin, Isabelle Phan, Lydie Bougueleret, and Amos Bairoch. Hamap : a database of completely sequenced microbial proteome sets and manually curated microbial protein families in uniprotkb/swiss-prot. *Nucleic Acids Res*, 37(Database issue) :D471–D478, Jan 2009. 77

[70] C. Lindberg. The unified medical language system (umls) of the national library of medicine. *J Am Med Rec Assoc*, 61(5) :40–42, May 1990. 78

[71] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *Proc. 27th International Conference on Very Large Data Bases (VLDB)*, pages 48–58, Roma (IT), 2001. 89

[72] F. Masseglia, M. Teisseire, and P. Poncelet. Extraction de motifs séquentiels. problèmes et méthodes. *Revue EGC Ingénieur des Systèmes d'Informations, numéro spécial Extraction et usage multiple de motifs dans les BD*, 9(34) :183–210, 2004. 34, 38

[73] Catherine Mathé, Marie-France Sagot, Thomas Schiex, and Rouzé Pierre. Survey and summary : Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res.*, 30(19) :4103–4117, 2002. 22

[74] A. M. Maxam and W. Gilbert. A new method for sequencing dna. *Proc Natl Acad Sci U S A*, 74(2) :560–564, Feb 1977. 17

[75] Mark McDermott, Michael J O Wakelam, and Andrew J Morris. Phospholipase d. *Biochem Cell Biol*, 82(1) :225–253, Feb 2004. 59

[76] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding : a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE)*, pages 117–128, San Jose (CA US), 2002. 90, 106

[77] George Miller. Special issue, wordnet : An on-line lexical database. *International Journal of Lexicography*, 4(3), 1990. 72, 79

[78] Renée Miller, Laura Haas, and Mauricio Hernández. Schema mapping as query discovery. In *Proc. 26th International Conference on Very Large Data Bases (VLDB)*, pages 77–88, Cairo (EG), 2000. 92

[79] Prasenjit Mitra, Gio Wiederhold, and Jan Jannink. Semi-automatic integration of knowledge sources. In *Proc. 2nd International Conference on Information Fusion*, pages 572–581, Sunnyvale (CA US), 1999. 88

[80] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proc. 8th Conference on Extending Database Technology (EDBT)*, volume 1777 of *Lecture notes in computer science*, pages 86–100, Praha (CZ), 2000. 89

[81] Riichiro Mizoguchi. Tutorial on ontological engineering : part 3 : Advanced course of ontological engineering. *New Gen. Comput.*, 22(2) :198–220, 2004. 71

[82] Dilvan A Moreira and Mark A Musen. Obo to owl : a protege owl tab to read/save obo ontologies. *Bioinformatics*, 23(14) :1868–1870, Jul 2007. 81, 82

[83] I. Moszer, L.M. Jones, S. Moreira, C. Fabry, and A. Danchin. Subtilist : the reference database for the Bacillus subtilis genome. *Nucleic Acids Res*, 30 :62–5, 2002. 10, 73

[84] Gerardus Johannes Mulder. On the composition of some animal substances. *Journal für praktische Chemie*, 16 :129, 1839. 15

[85] Christopher J. Mungall. Obol : integrating language and meaning in bio-ontologies : Conference papers. *Comp. Funct. Genomics*, 5 :509–520, 2004. 81, 82

[86] Saul Needleman and Christian Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3) :443–453, 1970. 84

[87] F. Nikitin, B. Rance, M. Itoh, M. Kanehisa, and F. Lisacek. Using protein motif combinations to update kegg pathway maps and orthologue tables. In *Genome Inform Ser Workshop Genome Inform.*, volume 15, pages 266–275, 2004. 59, 60

[88] N. Noy and M. Musen. PROMPT : Algorithm and tool for automated ontology merging and alignment. In *Proc. of AAAI-2000*, pages 450–455. MIT Press, 2000. 89

[89] Natalya F. Noy and Mark A. Musen. Anchor-PROMPT : Using non-local context for semantic matching. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 63–70, 2001. 89, 97

[90] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD Rec.*, volume 24, pages 175–186, 1995. 33

[91] J. Pei, J. Pinto, Q. Chen, U. Dayal, and M.C. Hsu. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *Int. Conf. on Data Engineering*, 2001. 34, 36, 45, 46

[92] G. Piatetsky-Shapiro. *Discovery, Analysis, and Presentation of Strong Rules*. AAAI/MIT Press, Cambridge, MA, 1991. 39

[93] Bastien Rance, Jean-François Gibrat, and Christine Froidevaux. An adaptive combination of matchers : application to the mapping of biological ontologies for genome annotation. In *Proc. of the 5th Data Integration in the Life Sciences workshop DILS'09*, LNBI 5647, pages 113–126, 2009. 97

[94] Chantal Reynaud and Brigitte Safar. Utilisation de connaissances supplementaires pour la decouverte de mappings dans le systeme TaxoMap . In *Atelier DECOR, EGC 2007*, 2007. 93, 98

[95] M. Riley. Systems for categorizing functions of gene products. *Curr Opin Struct Biol*, pages 388–392, 1998. 10, 73

[96] S.C.G. Rison, T.C. Hodgman, and J.M. Thornton. Comparison of functional annotation schemes for genomes. *Functional & Integrative Genomics*, 1 :56–69, 2000. 10

[97] M. Ronaghi, M. Uhlén, and P. Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281(5375) :363, 365, Jul 1998. 18

[98] Cornelius Rosse and José L V Mejino. A reference ontology for biomedical informatics : the foundational model of anatomy. *J Biomed Inform*, 36(6) :478–500, Dec 2003. 79

[99] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter, and H.W. Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res.*, 14((32)18) :5539–5545, 2004. 10, 73, 79

[100] Stuart Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Prentice Hall, Englewood Cliffs (NJ US), 1995. 91

[101] Jr Saier, Milton H., Can V. Tran, and Ravi D. Barabote. TCDB : the Transporter Classification Database for membrane transport protein analyses and information. *Nucl. Acids Res.*, 34(suppl_1) :D181–186, 2006. 78

[102] F. Sanger and A. R. Coulson. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *J Mol Biol*, 94(3) :441–448, May 1975. 17

[103] F. Sanger, S. Nicklen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12) :5463–5467, Dec 1977. 17

[104] Ashoka Savasere, Edward Omiencinsky, and Shamkant B. Navathe. An efficient algorithm form mining association rules in large databases. In *VLDB International Conference on Very Large Data Bases*, pages 432–444, 1995. 33

[105] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, O. B. I. Consortium, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H Scheuermann, Nigam Shah, Patricia L Whetzel, and Suzanna Lewis. The obo foundry : coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol*, 25(11) :1251–1255, Nov 2007. 79

[106] Larisa N Soldatova and Ross D King. Are the current ontologies in biology good ontologies ? *Nat Biotechnol*, 23(9) :1095–1098, Sep 2005. 69, 72

[107] R. Srikant and R. Agrawal. Mining sequential patterns : Generalizations and performance improvements. In *Int. Conf. Extending Database Technology*, pages 3–17, 1996. 34

[108] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. *KDD 97*, pages 67–67, 1997. 35

[109] Gerd Stumme and Alexander Mädche. FCA-Merge : Bottom-up merging of ontologies. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 225–234, Seattle (WA US), 2001. 91

[110] P.N. Tan, V. Kumar, and J. Srivastava. Selecting the right interstingness measure for association patterns. In *SIGKDD'02*, 2002. 38

[111] The Gene Ontology Consortium. Creating the gene ontology resource : design and implementation. *Genome Res.*, 11 :1425–1433, 2001. http ://www.geneontology.org. 10, 75

[112] Andreas Thor, Toralf Kirsten, and Erhard Rahm. Instance-based matching of hierarchical ontologies. In *Proc. 12 th German Database Conf. (BTW*, 2007. 103

[113] Konstantin Todorov and Peter Geibel. Ontology mapping via structural and instance-based similarity measures. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. 103

[114] Hannu Toivonen. Sampling large databases for association rules. In *VLDB International Conference on Very Large Data Bases*, pages 134–145, 1996. 33

[115] Mathias C Walter, Thomas Rattei, Roland Arnold, Ulrich Güldener, Martin Münsterkötter, Karamfilka Nenova, Gabi Kastenmüller, Patrick Tischler, Andreas Wölling, Andreas Volz, Norbert Pongratz, Ralf Jost, Hans-Werner Mewes, and Dmitrij Frishman. Pedant covers all complete refseq genomes. *Nucleic Acids Res*, 37(Database issue) :D408–D411, Jan 2009. 73

[116] J.C. Wootton and S. Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.*, 17 :149–163, 1993. 24, 59

[117] Ronald Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on System, Man and Cybernetics*, 18(1) :183–190, 1988. 107

[118] M. Zaki. Spade :an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2) :31–60, 2001. 34, 38

[119] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithm for fast discovery of association rules. In *Intl. Conf. on Knowledge Discovery and Data Mining*, pages 12–15, 1997. 33

[120] Tao Zhang. Association rules. In *PAKDD*, pages 245–256, 2000. 39