THÈSE

présentée à

L'Université Joseph FOURIER
GRENOBLE I

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

(Arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Spécialité

INFORMATIQUE

par

Olivier SCHMELTZER

MODÉLISATION DE CARTES GÉNOMIQUES
UNE FORMALISATION ET UN ALGORITHME DE CONSTRUCTION FONDÉ SUR LE RAISONNEMENT TEMPOREL

Soutenue le 23 janvier 1995 devant le jury composé de :

M.	Yves	CHIARAMELLA	Président
Mme	Marie-Catherine	VILAREM	Rapporteurs
MM.	Christian	GAUTIER	
	Laurent	TRILLING	Examinateur
	François	RECHENMANN	Directeur

Thèse préparée au sein du laboratoire LIFIA/IMAG

« Say, what you doing?
— Making slides. When I started I put starfish sperm and ova in each of these glasses. Then every halh-hour, I kill one glass of developing embryos, and when I have the whole series, I mount them on slides like this, and one slide shows the whole development. »
Suzy bent over the dishes.
« I don't see nothing.
— They are too small. I can show you in the glass. »
Suzy backed up.
« Why do you do it for?
— So students can see how starfish get to be.
— Why do they want to know?
— Well, I guess because that's the way people get to be.
— Then why don't they study people? »
Doc laughed.
« It's a little difficult to kill unborn babies every half-hour. Here, take a look. »
John Steinbeck, *Sweet Thursday*

« Qu'est-ce que vous faites ? demanda-t-elle.
— Je fais des plaques. Pour commencer, je mets du sperme d'astérie et des ovules dans chacun de ces verres. Puis, de demi-heure en demi-heure, j'arrête le développement en cours en tuant les embryons et quand j'ai toute la série je la dispose sur une plaque comme celle-ci où on distingue tout le processus de transformation. »
Suzy se pencha et dit :
« Je ne vois rien.
— Ils sont trop petits. Je vais vous montrer dans le microscope. »
Suzy se releva.
« Pourquoi faites-vous ça ?
— Pour que les étudiants puissent voir comment se forme une astérie.
— Et pourquoi veulent-ils le savoir ?
— Parce que le processus est le même que pour les hommes.
— Pourquoi est-ce qu'ils n'étudient pas les hommes ? »
Doc rit.
« Il est assez difficile de tuer toutes les demi-heures des bébés en formation. Tenez, regardez. »
John Steinbeck, *Tendre jeudi*

Remerciements

Avant tout, je tiens à remercier les membres du jury, Yves Chiaramella pour avoir accepté la lourde tâche de présider, Marie-Catherine Vilarem et Christian Gautier pour leur lecture attentive, François Rechenmann pour m'avoir dirigé pendant ces trois années de thèse, et Laurent Trilling pour son intérêt dans ce travail.

Je remercie aussi bien sûr toutes les personnes qui m'ont aidé à quelque titre que ce soit tout au long de la thèse. Pour ne pas les nommer, et en espérant n'avoir oublié personne,

> Jérôme Gensel et Pierre Girard pour de nombreuses discussions sur Tropes et son implémentation première ;
>
> Bruno Orsier pour tout ce qu'il a pu m'apprendre dans de nombreux domaines de l'informatique et tout particulièrement dans l'utilisation de LaTeXet Emacs ;
>
> les divers lecteurs de l'équipe pour leurs commentaires avisés : Florence Lemaire, Patrice Uvietta, Jérôme Euzenat, François Rechenmann ;
>
> Alain Garreau pour tout le temps passé sur l'interface cartographique ;
>
> Franck Dorkeld et Christian Gautier pour leur aide sur la biologie moléculaire et plus particulièrement les cartes génomiques ;
>
> les petits derniers de l'équipe : Isabelle Crampé, Sueli Ferreira et Jean-Marc Gabriel ;
>
> tous les thésards et stagiaires de l'équipe pour les moments passés au RU, à prendre un café et à lire l'horoscope : Nathalie, Cécile, Isabelle, Jean-Marc, Alain, Jérôme, Pierre, Florence, Jean-Yves, Bruno, Nina, Jutta ;
>
> le reste des membres de l'équipe : Danielle, Gilles, Pierre ;
>
> et enfin, une dernière personne, non la moindre, qu'il n'est nul besoin de nommer ici, mais que je remercie du fond du cœur pour tous les moments passés ensemble.

Je souhaite aussi beaucoup de courage à ceux qui sont près de la fin et auxquels il tarde d'en avoir terminé : Jean-Maurice, Philippe, Loan, Pierre, Jérôme, Bruno et Nina. N'oubliez pas de me prévenir quand ce sera votre tour...

Table des matières

Chapitre 1

Introduction

1.1 Les projets «Génome»

Depuis environ une dizaine d'années ont été lancés des projets de biologie moléculaire appelés «Programmes Génome» dont le but a été initialement de séquencer le génome d'un organisme, en particulier l'Homme, mais s'est transformé pour devenir la détermination des cartographies physique et génétique [Robbins92]. Ce sont les États-Unis qui ont pris l'initiative d'un tel projet en 1986, sous la responsabilité du Department of Energy (DOE), avec comme organisme cible l'Homme. Ensuite, les National Institutes of Health (NIH) se sont également investis dans ce projet, de même que le Howard Hugues Medical Institute (HHMI). En France, le Centre d'Étude du Polymorphisme Humain (CEPH) a eu pour but la constitution d'une collection de familles, de manière à étudier la ségrégation de caractères génétiques chez l'Homme pour en déterminer la carte génétique. L'Association Française contre les Myopathies (AFM) a mis en place un laboratoire (le Généthon) rassemblant des machines et des techniciens afin d'aider les laboratoires à travailler plus efficacement et d'élaborer une carte globale du génome humain [Vaysseix92]. Récemment a été créé le Groupement de Recherches et d'Études sur les Génomes (GREG), dont un des projets, dont nous aurons à reparler, est de développer un environnement coopératif d'aide à l'analyse de séquences ; il implique le projet Sherpa de l'Inria Rhône-Alpes, la société Ilog, le laboratoire de Biométrie, Génétique et Biologie des Populations de Lyon et l'institut Pasteur pour une durée de deux ans (1994 et 1995). Enfin, au niveau international, a été créée une institution appelée HUGO pour *Human Genome Organisation*, dans le but de coordonner la recherche sur le génome humain ; cette organisation est pour le moment fortement représentée par les États-Unis [Jordan92].

Indépendamment des projets sur le génome humain, il est très utile de s'attaquer à des génomes d'autres organismes, qui peuvent apprendre aux biologistes moléculaires énormément d'informations sur leur fonctionnement [Danchin93]. D'une part les génomes d'animaux peuvent être plus simples que celui de l'Homme (tout en possédant éventuellement des mécanismes similaires), d'autre part des expériences peuvent leur être appliquées qui ne peuvent pas l'être à l'Homme. La souris est un organisme de choix car son génome est très voisin de celui de l'Homme (ceci signifie par exemple que des groupes de gènes de la souris se retrouvent chez l'Homme avec la même fonction). D'autres organismes classiques d'étude sont le colibacille *Escherichia Coli* auquel de nombreux laboratoires à travers le monde s'intéressent en raison de la petite taille de son génome et de la perspective pas trop lointaine de son séquençage complet ; la levure du boulanger, *Saccharomyces Cerevisiae* fait l'objet d'un projet européen de séquençage et présente l'intérêt d'avoir, comme l'Homme, des cellules à noyau ; la mouche drosophile, la plante *Arabidopsis Thaliana*, le ver des sols *Cœnorhabditis Elegans* (qui se prête à l'étude de la différenciation cellulaire) sont d'autres exemples d'organismes étudiés intensément. La pléthore d'organismes et l'importance des recherches en biologie moléculaire font que les données recueillies submergent littéralement la communauté, ce qui nécessite le développement d'outils informatiques adéquats permettant de gérer cette masse de données. C'est pourquoi, en plus de toute la recherche proprement biologique, de nombreux travaux sont menés en parallèle pour traiter efficacement les données produites. De plus en plus d'informaticiens se plongent ainsi dans les méandres de la biologie moléculaire, et ce dans différents domaines allant des bases de données à la programmation d'algorithmes performants de recherche de gènes sur les séquences, de l'apprentissage à la robotique en passant par la statistique [Frenkel91, Lander et al.91]. L'Intelligence Artificielle est un domaine privilégié car nombre des problèmes biologiques rencontrés impliquent l'utilisation de techniques d'IA ; ceci est fondamentalement dû au caractère incomplet, voire contradictoire, des données biologiques issues des expériences, qui nécessitent des systèmes à même de gérer l'incohérence, les descriptions multiples, le manque d'information, etc. [Hunter91]. Avant de détailler quelques uns des problèmes biologiques et leur traitement à travers des techniques informatiques, il est indispensable de préciser les concepts manipulés en biologie moléculaire.

1.2 Quelques notions de biologie moléculaire

La présentation qui suit s'inspire largement de différents ouvrages de biologie moléculaires, en l'occurrence [Suzuki et al.89, Kaplan et al.90, Kourilsky90].

1.2.1 Génome, chromosomes et ADN

Tout être vivant est constitué d'une ou plusieurs cellules (à l'exception des virus) séparant l'extérieur (où se trouvent les sources d'énergie et les matières premières) de l'intérieur (où est réalisé le projet cellulaire, la reproduction par exemple). Chaque cellule possède un exemplaire du *génome* de l'organisme auquel elle appartient (i.e. l'ensemble de son patrimoine génétique), qui est contenu dans ses chromosomes. Chaque chromosome est composé de deux brins d'ADN formés d'une séquence de quatre nucléotides (ou bases) différents : l'adénine, la cytosine, la guanine et la thymine. Ces deux brins sont dits complémentaires en ce sens que chacune des bases s'apparie de manière unique avec une autre. Ainsi, l'adénine s'apparie avec la thymine (et réciproquement) et la cytosine s'apparie avec la guanine (et réciproquement). L'information génétique est donc disponible en double exemplaire, sur chacun des brins d'ADN (figure 1.1). Cette structure permet la duplication de la molécule d'ADN (appelée *réplication*), grâce à la séparation des deux brins, qui donne lieu à une réplique conforme ; ce même mécanisme sert à synthétiser l'ARN messager lors de la transcription (Cf. plus loin).

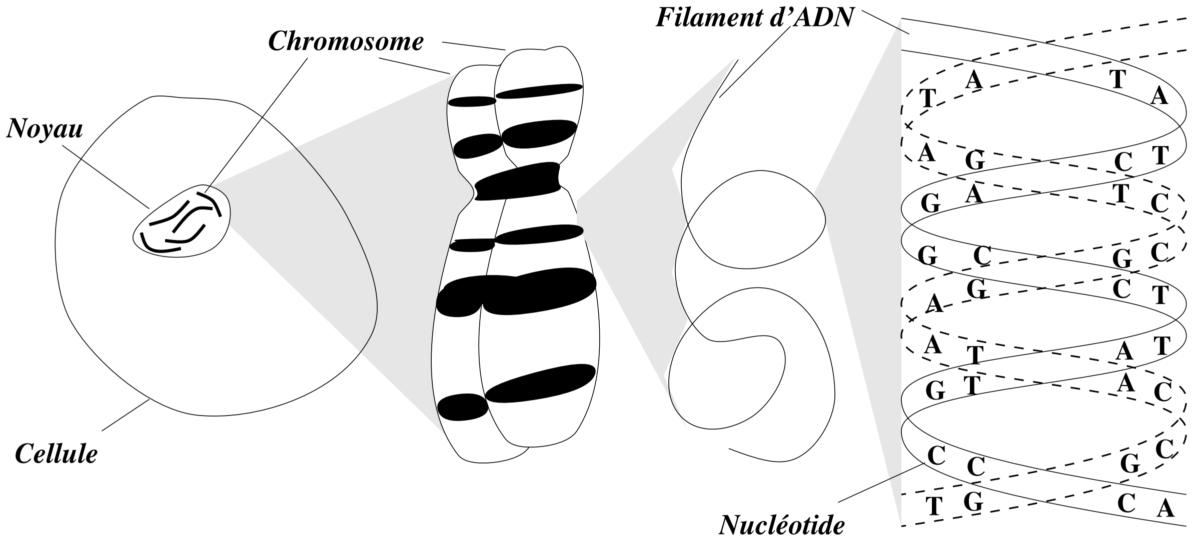


Figure 1.1 - : Cellule eucaryote. Dans le noyau de la cellule sont conservés les chromosomes ; l'ADN qu'ils contiennent est une macro-molécule formée de quatre bases appariées deux à deux : A avec T et C avec G.

Le nombre des chromosomes diffère d'une espèce à l'autre ; une bactérie telle que *E. Coli* en a un seul, l'Homme vingt-trois paires. Ce premier organisme est un exemple de *procaryote*, caractérisé par l'absence de noyau dans ses cellules, par opposition à l'Homme qui est un *eucaryote* ; c'est de plus un organisme asexué ; c'est pourquoi il ne possède qu'un jeu unique de chromosome : on parle alors d'organisme *haploïde*. Ce chromosome a la particularité que ses extrémités se rejoignent, lui donnant une forme circulaire. L'Homme, quant à lui, a un mode de reproduction sexué et dispose de vingt-trois paires de chromosomes linéaires. Certains êtres sexués sont des organismes *diploïdes*, à deux jeux de chromosomes ; seules leurs cellules sexuelles sont *haploïdes* ; la fusion d'un spermatozoïde avec un ovule redonne une cellule diploïde combinant les chromosomes parentaux. La production des cellules sexuelles est à la base d'un phénomène appelé recombinaison (Cf. §2.1.2), dont l'étude permet la construction de la carte génétique. L'apparition du mode de reproduction sexué a eu pour conséquence une augmentation de la complexité des organismes pour lesquels l'acquisition de modifications avantageuses a été accélérée. Ce mécanisme de reproduction est apparu de pair avec les êtres multi-cellulaires qui possèdent sur les organismes mono-cellulaires un avantage fondamental en la spécialisation cellulaire. Les cellules germinales ont pour fonction la reproduction, tandis que les cellules somatiques se spécialisent en différentes tâches, comme la production d'anticorps, le transport de l'oxygène, la digestion, etc.

1.2.2 Gènes, code génétique et protéines

Les quatre lettres de l'alphabet génétique, A, T, G, et C, s'agencent dans les chromosomes pour former des mots signifiants, les *gènes*. Un gène est transcrit en un simple brin, l'ARN messager, dans lequel l'uracile a remplacé la thymine ; cet ARN messager sera à son tour traduit en protéine. Cette description de la *transcription* vaut pour la plupart des organismes procaryotes. Elle se complexifie chez les eucaryotes (et chez certaines bactéries) car leurs gènes sont morcelés. Les portions excédentaires appelées *introns* doivent être éliminées de l'ARN messager pour ne conserver que les parties codantes ou *exons* [Danchin et al.84, Kourilsky et al.84]. La transcription commence donc par engendrer un ARN pré-messager contenant les introns, puis l'excision de ceux-ci a lieu par un processus appelé *épissage* (figure 1.2).

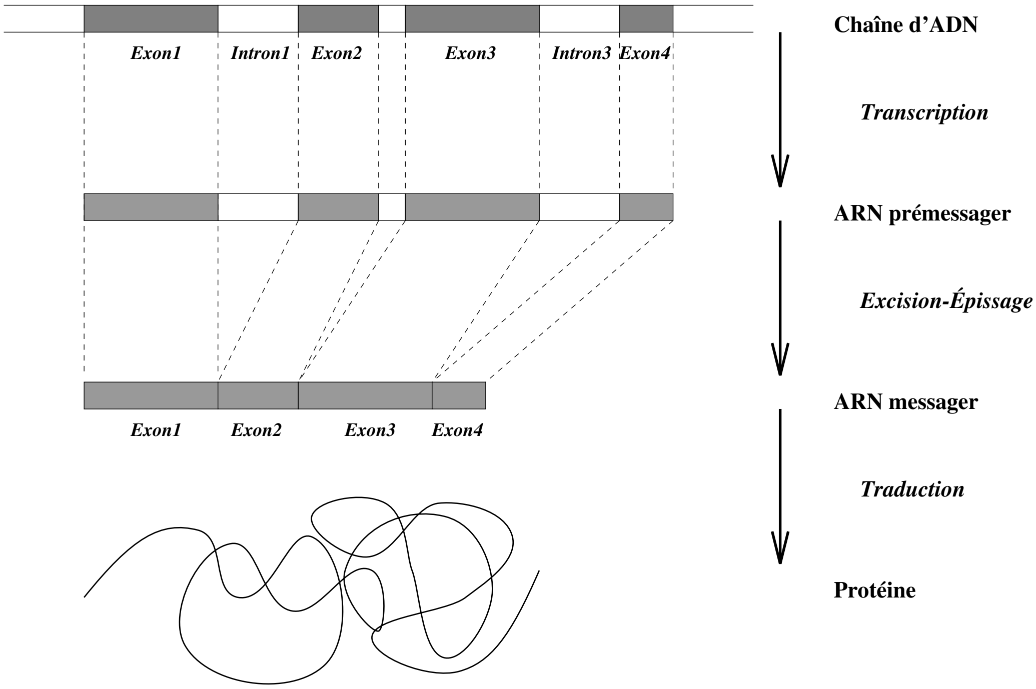


Figure 1.2 - : Mécanismes de transcription et de traduction chez les eucaryotes. La transcription permet de passer de l'ADN à un ARN pré-messager, dont les introns sont excisés et les exons mis bout à bout (épissage), ce qui donne l'ARN messager traduit enfin en protéine.

Il existe ainsi une correspondance 1-1 entre un gène et une protéine[1]. La traduction permet alors de passer des nucléotides constitutifs de l'ARN messager aux acides aminés, au nombre de vingt, par l'intermédiaire du code génétique, associant à chaque triplet de nucléotides, ou *codon*, un acide aminé. Étant donné que trois nucléotides peuvent s'arranger de $4^3 = 64$ manières différentes, le code génétique est dégénéré, et, de manière générale, c'est le dernier codon qui apporte le moins d'information. Ce code intègre également des signes de ponctuation, en trois codons STOP qui arrêtent la traduction. De plus, le premier codon, qui initie la traduction, est presque toujours le codon AUG correspondant à la méthionine (figure 1.3).

Le choix du premier codon définit une *phase de lecture*; une *phase ouverte de lecture* commence à un codon d'initiation et s'arrête avec une occurrence de codon STOP. Un brin d'ARN comporte trois phases de lecture, et seulement trois parce que le brin d'ARN (comme celui d'ADN) est orienté et n'est lisible que dans un sens, de l'extrémité 5' vers l'extrémité 3' ; par conséquent, la molécule d'ADN peut être lue de six manières différentes (figure 1.4).

[1] Cette affirmation est à tempérer puisque la fabrication des anticorps en est un contre-exemple [Rougeon86].

Seconde lettre

Première lettre	U		C		A		G		***Troisième lettre***
U	**UUU Phe**	**UUC Phe**	**UCU Ser**	**UCC Ser**	**UAU Tyr**	**UAC Tyr**	**UGU Cys**	**UGC Cys**	**U C**
	UUA Leu	**UUG Leu**	**UCA Ser**	**UCG Ser**	**UAA**	**UAG *STOP***	**UGA**	**UGG Trp**	**A G**
C	**CUU Leu**	**CUC Leu**	**CCU Pro**	**CCC Pro**	**CAU His**	**CAC His**	**CGU Arg**	**CGC Arg**	**U C**
	CUA Leu	**CUG Leu**	**CCA Pro**	**CCG Pro**	**CAA Gln**	**CAG Gln**	**CGA Arg**	**CGG Arg**	**A G**
A	**AUU Ile**	**AUC Ile**	**ACU Thr**	**ACC Thr**	**AAU Asn**	**AAC Asn**	**AGU Ser**	**AGC Ser**	**U C**
	AUA Ile	**AUG Met**	**ACA Thr**	**ACG Thr**	**AAA Lys**	**AAG Lys**	**AGA Arg**	**AGG Arg**	**A G**
G	**GUU Val**	**GUC Val**	**GCU Ala**	**GCC Ala**	**GAU Asp**	**GAC Asp**	**GGU Gly**	**GGC Gly**	**U C**
	GUA Val	**GUG Val**	**GCA Ala**	**GCG Ala**	**GAA Glu**	**GAG Glu**	**GGA Gly**	**GGG Gly**	**A G**

Figure 1.3 - : Le code génétique (d'après [Hélène84]). De l'alphabet à quatre lettres de l'ADN, on passe à celui à vingt lettres des protéines grâce au code génétique qui associe à tout groupe de trois nucléotides (ou codon) un acide aminé. Ce code est dégénéré, c'est le plus souvent le dernier nucléotide qui apporte le moins d'information. Il existe également des codons STOP qui arrêtent la traduction.

Ces mécanismes sont malheureusement entachés de nombreuses exceptions ; ainsi, le principe de colinéarité des gènes et des protéines, qui énonce que chaque acide aminé d'une protéine est associé au codon correspondant dans le gène, et que des acides aminés consécutifs sont déterminés par des codons consécutifs (aux introns près) dans l'ADN et l'ARN, ce principe est contredit par le mécanisme d'«editing» des ARN messagers modifiant ces ARN pour les rendre intelligibles [Benne et al.92].

Les protéines obtenues par traduction de l'ARN messager ont de nombreuses fonctions [Doolittle85], entre autres celle de gérer l'expression des gènes de l'ADN. Elles sont également (et peut-être avant tout) un support structurel à l'édifice cellulaire. Ce sont des enzymes permettant que des réactions chimiques aient lieu, qu'elles servent aussi bien à synthétiser des molécules (*anabolisme*) qu'à dégrader des constituants (*catabolisme*) tout en fournissant l'énergie nécessaire à ces transformations. La survie d'une cellule – et par voie de conséquence de l'organisme entier – dépend en premier lieu de sa capacité à fabriquer les protéines dont elle a besoin et quand elle en a besoin.

L'activité des différentes molécules, que ce soient des acides nucléiques ou des protéines, dépend fondamentalement de leur structure tridimensionnelle, appelée conformation. Par exemple, une enzyme est une molécule spatiale, qui occupe un certain volume, telle que les substrats de la réaction chimique qu'elle facilite vont s'y rattacher en des endroits adéquats pour que la réaction ait lieu. De même, l'auto-épissage de certains ARN pré-

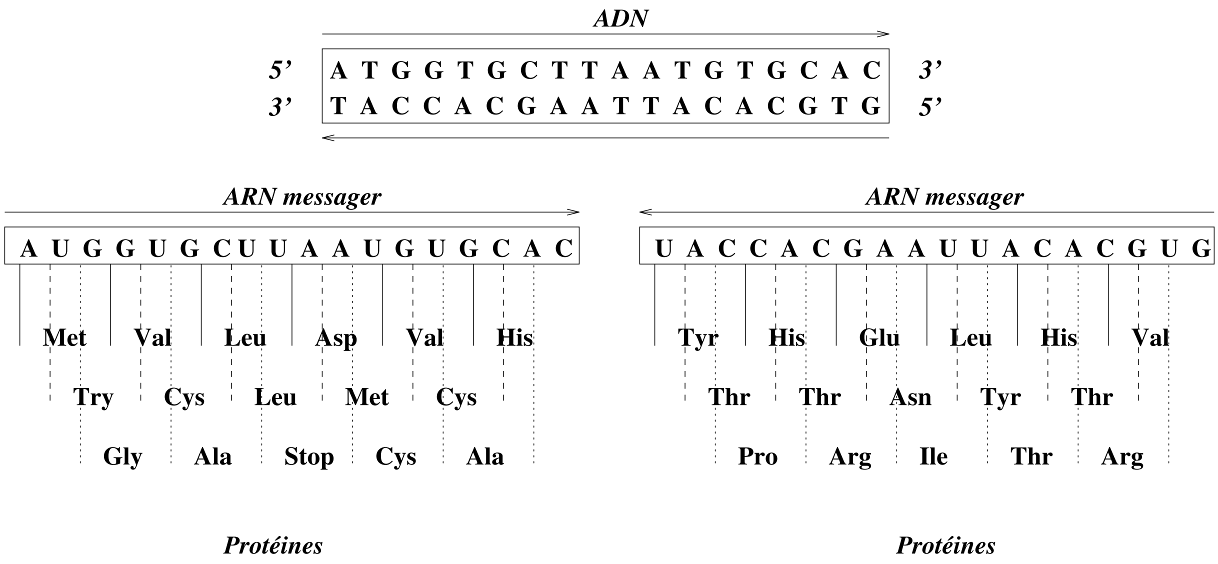


Figure 1.4 - : Les six cadres de lecture de l'ADN. La molécule d'ADN peut être lue de six manières différentes, selon chacun des brins et selon le choix du premier codon.

messagers est dû au fait que des séquences séparées peuvent être proches spatialement ce qui autorise l'excision de l'intron.

1.2.3 Polymorphisme, génotype et phénotype

Les gènes, la plupart du temps, existent sous différentes versions appelées *allèles*. Ce polymorphisme peut parfois être mis en évidence par le *phénotype* de l'organisme (c'est-à-dire l'ensemble des caractères visibles, comme par exemple la couleur des yeux). Le *génotype* est par opposition le codage qui mène au phénotype. Il existe donc pour un même phénotype un grand nombre de génotypes qui le produisent puisque certains changements de la séquence d'ADN ne modifient pas la protéine produite ou la modifient mais celle-ci conserve ses propriétés, liées à sa structure spatiale. La mécanique évolutionnaire modifie ponctuellement le génotype, et opère une sélection (naturelle) sur les phénotypes produits qui tend à ne garder que les plus aptes à la survie [Hunter93]. Certains changements de la séquence nucléotidique d'un gène (ou *mutations*), qui peuvent par exemple apparaître lors de la réplication, sont sans effet sur son rôle (une telle mutation est dite neutre) tandis que d'autres empêchent la génération de la protéine correspondante ou la modifient, engendrant le plus souvent un effet négatif. Dans le cas de cellules diploïdes, il y a moins de chances qu'une mutation soit visible car les gènes sont en double exemplaire, ce qui fait qu'un gène inactif sur un chromosome sera le plus souvent présent sur son homologue et pourra en assurer la fonction.

La réalité biologique est autrement plus compliquée que ce qui a été présenté dans les paragraphes précédents. Les problèmes de régulation génétique n'ont pas été abordés, de même que les mécanismes de conformation spatiale des molécules, etc. Néanmoins, les notions qui ont été traitées, et qui seront étendues dans le chapitre spécifique aux cartes génomiques (chapitre 2), sont suffisantes pour la compréhension du mémoire.

1.2.4 De la taille des génomes

Pour terminer cette présentation, il est intéressant d'avoir une idée de certains ordres de grandeur concernant les tailles des génomes manipulés par les biologistes [Danchin93]. L'unité employée est la base (ou paire de bases, car la molécule d'ADN est composée de deux brins), ou un de ses multiples, la kilobase (kb) ou la mégabase (Mb). Le génome de l'Homme est réparti sur vingt-trois chromosomes constitués de 3300 mégabases ; celui de la souris est un peu plus petit et fait 3000 Mb. La drosophile a un génome de 170 Mb et le nématode 110 Mb. Le génome de la plante *Arabidopsis Thaliana* a 100 Mb, celui de la levure 15 Mb. Les génomes des organismes inférieurs sont plus petits : *Escherichia Coli* possède un génome de 4720 kb ; les virus ont été les premiers génomes à avoir été séquencés dans leur intégralité : celui d'Epstein Barr est de 172 kb, celui de la varicelle de 124 kb.

Nous verrons plus loin (Cf. chapitre 2.1) les différentes techniques biologiques qui permettent à la fois de déterminer la séquence de nucléotides de ces génomes mais aussi de récolter des informations importantes sur les entités qui occupent la molécule d'ADN (les gènes en particulier).

1.3 Informatique et biologie moléculaire

Le mariage de l'informatique et de la biologie moléculaire est destiné à durer car, d'une part, cette dernière a désespérément besoin des techniques informatiques pour résoudre ses problèmes, que ceux-ci se situent au niveau d'algorithmes à implémenter efficacement ou à celui de la gestion des données biologiques [DeLisi88], d'autre part, elle est un terrain inépuisable d'expérimentation pour les chercheurs en informatique, posant des questions non triviales et étant un domaine d'application réel. Avant de parler de la problématique de ce travail, nous passerons en revue divers problèmes biologiques et les techniques informatiques permettant, plus ou moins efficacement, de les résoudre.

1.3.1 Algorithmique biologique

La première utilisation de l'informatique par des biologistes moléculaires a été la programmation d'algorithmes permettant d'analyser la séquence d'ADN obtenue après séquençage. Ces algorithmes vont de la recherche de motifs fonctionnels (indiquant par exemple la présence de gènes) à la comparaison de plusieurs séquences en passant par la détermination de conformation de protéines ou d'acides nucléiques.

Un algorithme classique pour la comparaison de séquences biologiques est celui de programmation dynamique découvert dans le domaine de la biologie moléculaire par Needleman et Wunsch [Waterman89]. Le résultat de l'application de cette méthode est une matrice exprimant pour toutes les sous-séquences des deux séquences à comparer un coût d'édition, c'est-à-dire la somme de coûts élémentaires permettant de passer de l'une à l'autre. En biologie moléculaire, trois coûts sont pris en compte : la substitution d'un élément de la séquence par un autre, l'insertion d'un élément et sa délétion. Cette méthode peut s'appliquer aussi bien pour un acide nucléique que pour une protéine.

Des algorithmes tels que celui présenté sont très utiles aux biologistes ; il en existe une pléthore, de complexité variable. Néanmoins, il reste aux biologistes une charge importante qui consiste en l'évaluation des résultats, car ceux-ci ne sont issus que d'un modèle très approximatif de la réalité biologique. Il manque des techniques plus fines, autorisant des interactions avec un utilisateur, et apportant une modélisation plus fine. L'intelligence artificielle apporte de telles techniques comme cela va être montré brièvement maintenant.

1.3.2 Intelligence artificielle

La biologie moléculaire est devenue un domaine de prédilection pour de nombreuses applications en intelligence artificielle, car elle nécessite la gestion de grandes quantités de données, elle implique de multiples formes de connaissances et constitue ainsi un défi pour la validation de systèmes d'intelligence artificielle [Hunter91, Rawlings et al.94]. C'est pourquoi de plus en plus de systèmes utilisant des techniques propres à l'intelligence artificielle, que ce soit pour représenter ou raisonner, sont appliqués à la biologie moléculaire. Par exemple, la détermination de structures secondaires de molécules est un sujet traité par des techniques de réseaux de neurones [Steeg93, Holbrook et al.93], de satisfaction de contraintes [Hayes-Roth et al.86, Altman et al.94], etc. ; de même, les processus biologiques peuvent se modéliser, par exemple, à l'aide de système de représentation de connaissances et de simulation qualitative ([Karp93] pour l'opéron tryptophane, [Koile et al.89] pour la simulation de l'expression génétique). Un certain nombre de systèmes informatiques d'intelligence artificielle ont eu un impact important du fait de leur application à la biologie moléculaire comme MOLGEN [Stefik81] et GeneSys [Overton et al.90].

MOLGEN

L'article de Stefik introduit une approche de la planification hiérarchique basée sur l'utilisation de contraintes. Ces contraintes sont utilisées pour répondre à trois objectifs : l'élimination de certaines règles, le choix de descriptions partielles à développer et la communication entre les divers sous-problèmes. Le but du système informatique MOLGEN est l'assistance dans la mise en place d'une expérience. Ceci nécessite une modélisation des mécanismes biologiques, contenue dans un système à base de connaissances.

MOLGEN a été étendu par la suite [Friedland et al.85] pour découvrir des théories scientifiques en biologie moléculaire. En particulier, une étude a été réalisée sur la régulation de l'opéron tryptophane, dont le mécanisme est différent de celui, bien connu grâce aux travaux de Jacob et Monod, de l'opéron lactose. À partir de la modélisation de la régulation de l'opéron lactose, MOLGEN est à même d'expliquer des observations dans le cadre de ce mécanisme, de reconnaître des informations contradictoires ou non prédictibles et de construire des hypothèses d'extension ou de correction de cette théorie.

GeneSys

Le but du projet GeneSys est de formaliser la connaissance sur les structures et les fonctions moléculaires, de manière à utiliser des mécanismes d'inférence d'autant plus efficaces que cette formalisation aura été poussée. Ces connaissances sont introduites dans un langage de «frames», permettant leur expression déclarative ; les aspects dynamiques de la biologie moléculaire (en l'occurrence, la régulation des gènes) sont assurés par l'utilisation du langage Prolog. Ce dernier est aussi utilisé pour la représentation de relations complexes comme l'agrégation (qui peut être assimilée à la composition).

Les attributs des frames sont le reflet des structures de données des banques biologiques comme GenBank. Un langage de requêtes permet soit d'accéder directement à ces données, soit à la modélisation dans le langage de frames. GeneSys a pour vocation d'aider à la découverte de relations entre la structure des gènes et leur fonction, à l'aide de la représentation des données et des processus biologiques.

1.4 Problématique des cartes génomiques

Un des objectifs principaux de la cartographie des chromosomes, tout au moins en ce qui concerne l'Homme, est de comprendre les causes génétiques des maladies héréditaires dans le but de les prévenir voire de les guérir [White et al.88]. Déterminer le gène responsable d'une telle maladie (si tant est qu'un seul gène est impliqué[2]) est néanmoins une entreprise de longue haleine. Elle nécessite avant tout d'avoir balisé le génome pour placer ce gène par rapport à des repères bien positionnés, sur les différentes cartes.

1.4.1 Vocation des cartes

Les cartes génomiques sont la représentation de la connaissance accumulée sur le génome d'un organisme, c'est-à-dire sur ses chromosomes. En conséquence, elles sont monodimensionnelles et visent à spécifier, plus ou moins précisément, les positions des entités qui y apparaissent. Il existe trois types de carte, cytogénétique, génétique, physique, qui sont autant de représentations des mêmes chromosomes, mais selon des vues et des objectifs différents.

La carte cytogénétique permet de positionner toutes sortes de marqueurs par rapport aux bandes plus ou moins sombres qui apparaissent lors de la coloration des chromosomes.

La carte génétique a pour vocation de refléter les positions de marqueurs génétiques ; elle se base pour cela sur le polymorphisme de ces marqueurs qu'on met en évidence lors de la transmission des caractères qui leur sont liés. En effet, deux gènes proches sur un même chromosome sont susceptibles d'être souvent transmis ensemble à la descendance ; une étude statistique permet alors d'estimer une proximité entre deux caractères associés à deux gènes, et par extension un ordre. Des marqueurs de plus en plus informatifs (i.e. polymorphiques) sont désormais utilisés et sont moléculaires, c'est-à-dire basés sur la séquence d'ADN elle-même au lieu d'être phénotypiques, ou liés à l'apparence de l'organisme étudié. Une carte génétique des chromosomes humains (la carte génétique de l'ensemble des chromosomes humains a une longueur cumulée de 4000 centiMorgans) de très grande résolution a été atteinte récemment par des chercheurs du Généthon [Weissenbach et al.94] ; elle inclut plus de deux mille marqueurs, ce qui constitue déjà un balisage dense du génome, comparé à la première carte du génome humain entier [Donis-Keller et al.87] dont la résolution était de dix centiMorgans.

[2] La monogénicité des caractères, c'est-à-dire le fait qu'ils soient déterminés par un seul gène, n'est pas du tout systématique [Jordan92].

De la même manière que le phénomène de recombinaison balise le génome de marqueurs génétiques, la cartographie physique le balise avec des marqueurs liés à l'ADN ; une carte complète à 75% du génome humain a été obtenue [Guainville92], elle est constituée de points de repères, qui sont des séquences nucléotidiques particulières souvent rencontrées, espacés de 25 à 100 millions de nucléotides ; certains chromosomes (dont le chromosome 21, qui recèle de nombreux gènes impliqués dans des maladies héréditaires) ont été cartographiés à des résolutions plus grandes encore. Ces repères permettent de positionner facilement les fragments d'ADN qui ont été séquencés, et dans lesquels des gènes ont parfois été mis en évidence. Il est important de noter que les marqueurs se retrouvent très souvent d'un type de carte à un autre.

Les cartes génomiques possèdent une importance extrêmement grande aux yeux des biologistes car elles permettent de représenter, en particulier graphiquement, une grande quantité de connaissances sur les entités biologiques manipulées, faisant ressortir leurs positions relatives et les distances qui les séparent ; elles facilitent ainsi énormément le positionnement d'autres entités (le gène d'une maladie par exemple) par rapport à celles déjà positionnées. De plus, l'intégration des différentes cartes, c'est-à-dire le fait de pouvoir placer une même entité sur les différentes cartes, augmente les informations disponibles puisqu'elles peuvent provenir de différentes sources et se compléter mutuellement.

1.4.2 Problématique

La gestion des cartes se heurte malheureusement à de nombreux problèmes. Le premier est le grand volume des connaissances biologiques à manipuler ; il ne s'agit pas tant du stockage de la séquence des nucléotides que de son traitement et de l'utilisation de toute la connaissance qu'elle contient [Erickson92]. La séquence elle-même n'est informative que dans la mesure où elle a été analysée pour mettre en évidence des entités utiles aux biologistes, c'est-à-dire des gènes, des marqueurs spécifiques, des relations entre ces marqueurs, etc.

Un autre problème est lié à l'incomplétude des données ; en effet, dans le cas de la construction des cartes, celles-ci ne contiennent en général pas toutes les informations nécessaires à l'ordonnancement des entités. Les cartes sont donc souvent incomplètes et d'autres expériences sont souvent indispensables pour positionner précisément une entité par rapport à d'autres.

Enfin, ces données, comme elles proviennent d'expériences, sont entachées d'erreurs, créant de la sorte des incohérences. Là encore, d'autres expériences sont requises pour lever les ambiguïtés.

Ce travail s'attache à répondre à ces problèmes de représentation et de raisonnement sur les cartes à l'aide des techniques d'intelligence artificielle, que sont la représentation des connaissances et le raisonnement temporel.

1.4.3 Comment résoudre le problème de gestion des cartes ?

Une formalisation du problème est une étape nécessaire à l'élaboration d'une solution à ce problème ; deux aspects interviennent dans la justification d'une formalisation : le premier concerne le côté statique des cartes, c'est-à-dire la visualisation de cartes dites consensus, suffisamment figées pour être le reflet de la réalité biologique, le second est davantage lié au problème de la construction des cartes à partir de résultats expérimentaux, qui ne fournissent en général pas une carte définitive disponible immédiatement. Ces deux aspects biologiques sont reliés d'une part à la représentation des connaissances pour le premier, et au raisonnement pour le second.

Tout d'abord, une formalisation d'un problème, quel qu'il soit, permet de mieux le comprendre, car elle en fait une structuration. Formaliser facilite de beaucoup la spécification du logiciel amené à répondre au problème. Ce côté «génie logiciel», malgré sa connotation peut-être terre-à-terre, est très important. Dans le cas qui nous occupe, la formalisation va mettre en évidence les ensembles de données sur lesquels porteront les traitements, de la même manière que la programmation par objets spécifie d'abord les entités du domaine – les données – avant de développer des méthodes sur ces entités. Cette étape a également l'avantage de faciliter toute extension du modèle, justement parce qu'un tel modèle existe. Un programme qui ne se base pas sur une formalisation initiale est *a priori* plus difficile à étendre car les données se trouvent enfouies dans le code lui-même et ne sont donc pas accessibles. Enfin, la spécification des entités manipulées empêche des traitements et actions dépourvues de sens, et limite donc les erreurs lors de la définition des données. Ceci augmente beaucoup la cohérence des données en vue de leur traitement.

La formalisation des cartes génomiques facilite aussi grandement toute la partie algorithmique liée à la construction de celles-ci. En effet, pour bénéficier d'algorithmes avec certaines propriétés (complétude, correction, complexité), il est indispensable de bien définir le langage utilisé par ces algorithmes. Ainsi, on obtient à la fois plus de rigueur en caractérisant les algorithmes et plus de généricité grâce aux possibilités d'extension qui sont facilitées.

Une fois la formalisation effectuée, il est beaucoup plus facile de l'implémenter. Dans notre cas, cette implémentation sera réalisée dans un système de représentation de connaissances disposant de certaines fonctionnalités mises en évidence par la formalisation. La partie algorithmique bénéficiera aussi des avantages de la représentation déclarative offerte par le système de représentation de connaissances.

Enfin, un dernier avantage de la formalisation des cartes est de disposer d'un modèle sur lequel va reposer une interface cartographique indispensable pour représenter graphiquement les résultats biologiques. On peut ainsi obtenir une interface générique dans le but d'avoir, non pas une visualisation figée, mais plutôt un générateur d'interfaces de cartographie modifiable à volonté. Sans la présence d'un modèle sous-jacent, une telle possibilité n'existerait pas.

1.5 Apports de ce travail

La contribution originale de ce travail se situe à différents niveaux. Tout d'abord, nous avons introduit une abstraction du concept de carte génomique, basée sur un formalisme mathématique. Ensuite, cette abstraction a été implémentée dans un logiciel de représentation de connaissances, Tropes, en cours de développement au sein du laboratoire. Cette implémentation a consisté en l'extension effective du méta-modèle de Tropes et en l'écriture d'une base de connaissances représentant les entités introduites dans la formalisation. Une réalisation conjointe a été de développer un algorithme à même de résoudre le problème de construction des cartes ; l'algorithme proposé s'appuie sur des travaux de raisonnement temporel, de manière à profiter des avantages de déclarativité et d'efficacité d'algorithmes qui existent déjà dans ce domaine. Néanmoins, de profonds ajouts et modifications ont dû être introduits pour les adapter à la construction de cartes génomiques. L'algorithme final a été implémenté en Lisp à partir des connaissances présentes dans la base de connaissances et d'algorithmes de traitement de contraintes temporelles, qualitatives et quantitatives, disponibles auprès de leurs auteurs. Enfin, nous avons écrit les spécifications d'un générateur d'interfaces cartographiques, dans un souci de généricité par rapport aux logiciels existants ; ce générateur d'interfaces cartographiques est actuellement en cours d'implémentation par un stagiaire du CNAM.

Nous avons été amenés à faire différentes présentations de synthèse, tant pour introduire un thème particulier que pour justifier notre travail vis-à-vis de recherches antérieures. La première de ces présentations est une description des connaissances biologiques indispensables à la compréhension du mémoire. Nous avons également dû préciser les formalismes de représentation du temps et les raisonnements qui leur sont attachés, avant de voir comment les adapter à notre cas. Un état de l'art sur les logiciels de représentation et de raisonnement cartographiques, ainsi que sur les systèmes de représentation de connaissances, nous a permis de présenter le modèle TROPES et son utilisation. Pour finir, avant de détailler les spécifications du générateur d'interfaces cartographiques, il a paru nécessaire d'étudier les logiciels existants, de façon à se positionner par rapport à eux.

La section suivante montre l'organisation générale du mémoire en reprenant les aspects décrits ci-dessus.

1.6 Plan du mémoire

Ce mémoire est divisé en trois parties, chacune élaborant une des étapes nécessaires à une modélisation correcte et complète des cartes génomiques.

La première partie traite de la modélisation proprement dite des cartes génomiques. Le premier chapitre est un prélude indispensable qui détaille la problématique biologique des cartes génomiques, en abordant divers aspects : quels sont les différents types de carte, que recouvre la notion de carte, quelles sont les relations à exprimer entre les éléments qui forment les cartes, quelles sont les opérations effectuées sur les cartes.

Les chapitres qui suivent détaillent les stades de la modélisation, de la formalisation à l'algorithmique.

- Le chapitre 3 présente l'abstraction qui est faite des cartes génomiques, en introduisant une typologie et en spécifiant les relations qualitatives et quantitatives entre les éléments des cartes.

- Le chapitre 4 précise les requêtes auxquelles la formalisation précédente permet de répondre et termine par un algorithme simplifié de recherche des ensembles d'entités localement ordonnées.

- La théorie décrite dans les deux chapitres précédents est alors instanciée aux cartes génomiques (chapitre 5).

- Le chapitre suivant (chapitre 6) présente une synthèse des travaux sur la représentation du temps et le raisonnement temporel.

- L'algorithmique des cartes génomiques développée au chapitre 7 se fonde sur la présentation du chapitre précédent mais modifie et étend l'algorithmique temporelle pour y inclure les spécificités du raisonnement sur les cartes génomiques.

La seconde partie utilise la formalisation de la partie précédente pour son implémentation dans un système de représentation de connaissances.

- Le chapitre 8 présente un bref état de l'art sur les logiciels existants, dont le but est la représentation des cartes ou leur algorithmique.

- Il est suivi d'une présentation des systèmes de représentation de connaissances (chapitre 9).

- Ensuite est décrit le système Tropes qui servira de modèle de représentation de connaissances (chapitre 10).

- Enfin, l'implémentation proprement dite des éléments de la formalisation dans le modèle sera abordée (chapitre 11).

La troisième et dernière partie abordera les problèmes relatifs à la construction d'une interface homme-machine permettant de représenter les informations cartographiques.

- Le chapitre 12 donnera un aperçu des logiciels d'interface cartographique.

- Puis sera présentée l'approche adoptée dans le but de réaliser, non plus une interface particulière, mais un générateur d'interfaces cartographiques (chapitre 13).

- Les spécifications fonctionnelles de ce générateur seront décrites dans le chapitre 14.

Pour terminer, une conclusion résumera les points abordés dans ce travail et ses apports, tant informatiques que biologiques, et le situera dans un cadre plus général lié à la réalisation d'un projet du GREG sur un système générique d'aide à l'analyse de séquences.

Première partie

Modélisation de cartes génomiques

Chapitre 2

Les cartes génomiques

Ce chapitre est nécessaire à la compréhension du problème des cartes génomiques, et par voie de conséquence à la formalisation qui en est faite au chapitre 3 ; il décrit ce que sont les cartes génomiques tant au niveau de leur structure qu'à celui des relations qui les lient les unes aux autres.

2.1 Les différents types de carte génomique

Les cartes génomiques permettent de représenter la constitution des chromosomes et les relations qui existent entre ces constituants, et ce à des échelles différentes, jusqu'à aboutir à la séquence des nucléotides. Chaque carte est issue d'expériences diverses qui mettent en évidence des entités biologiques différentes, mais qui peuvent éventuellement se correspondre.

2.1.1 Les cartes cytogénétiques

Une carte cytogénétique est obtenue, chez les eucaryotes, par coloration des chromosomes à l'aide de différentes techniques, pendant la division de la cellule. On observe alors des bandes claires et sombres sur ceux-ci, plus ou moins précises selon le colorant utilisé et le moment où se font les mesures des intensités. En effet, suivant la condensation de l'ADN dans le chromosome, qui dépend de la phase dans laquelle se trouve la cellule pendant sa division mitotique, certaines bandes se subdivisent en sous-bandes. Ainsi, durant la prophase, il est possible d'observer jusqu'à 2000 bandes sur le génome humain. À l'exception de régions dites variables et autour du centromère, la variabilité, d'une coloration à l'autre, est très faible. On dispose ainsi de cartes de résolution différente ; chez l'Homme, typiquement, sont utilisées des cartes à 300, 400, 550, 850 bandes sur l'ensemble du génome. La désignation de ces bandes se fait récursivement en ajoutant un numéro dès qu'une bande se décompose en sous-bandes (figure 2.1).

Ces bandes sont utiles tout d'abord pour caractériser les chromosomes d'une espèce. D'autre part, elles permettent de positionner des gènes (ou plus généralement des marqueurs génétiques) directement sur les chromosomes et enfin elles apportent des informations supplémentaires, provenant des méthodes de coloration, comme par exemple le taux en bases C et G. En effet, les bandes claires sont en général des régions riches en C et

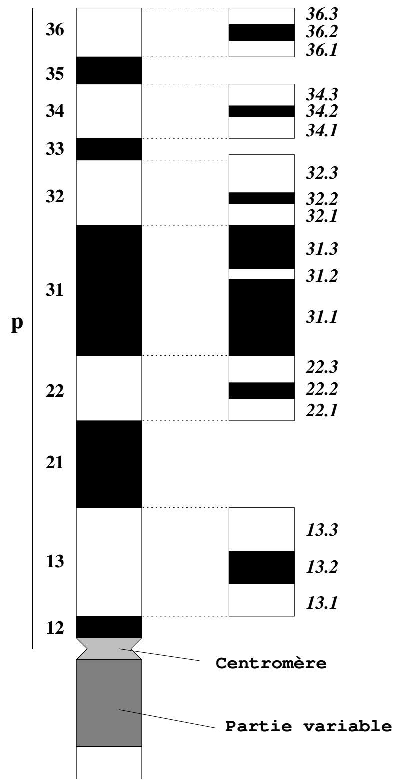


Figure 2.1 - : Subdivision des bandes cytogénétiques. Les bandes 13, 22, 31, 32, 34, 36 du chromosome 1 de l'Homme se subdivisent dans le système à 550 bandes en sous-bandes ; le centromère et la partie variable sont des zones où les bandes ne sont pas conservées d'une expérience de coloration à une autre.

G. De plus, le taux de G+C permet de partitionner le génome en *isochores*, les isochores lourds (i.e. riches en G+C) sont censés avoir une proportion plus importante de gènes [Mouchiroud91].

Les bandes de faible résolution sont issues de techniques plus anciennes, mais il est indispensable de les conserver car certains résultats ne sont disponibles, pour l'instant, que pour un bas niveau de «banding».

2.1.2 Les cartes génétiques

La construction d'une carte génétique est fondée sur ce qu'on appelle l'*analyse de liaison* ; cette analyse est possible chez les êtres sexués pour lesquels se produit le phénomène de méiose[1]. Celle-ci permet la formation des cellules sexuelles pendant laquelle un chromosome échange des segments avec son homologue ; ce phénomène a pour nom *recombinaison* ou *crossing-over*. Ainsi, deux allèles de deux gènes différents peuvent se trouver séparés après la recombinaison. Plus les gènes sont proches et moins ils ont de chance d'être séparés lors de la méiose car le crossing-over peut se produire sur toute la longueur séparant les gènes (figure 2.2). Une analyse de liaison est d'autant plus informative que

[1] Il existe un phénomène de *conjugaison* chez les bactéries mettant en évidence des distances génétiques ; en effet, celles-ci s'échangent du matériel génétique, et il est possible de mesurer la distance séparant deux gènes en interrompant ce processus à un moment donné.

les gènes sont polymorphiques ; en effet, si un allèle de gène est identique dans 99% de la population, il sera peu probable de tomber sur des individus possédant le second allèle. La recombinaison, même si elle a lieu, ne sera donc pas visible.

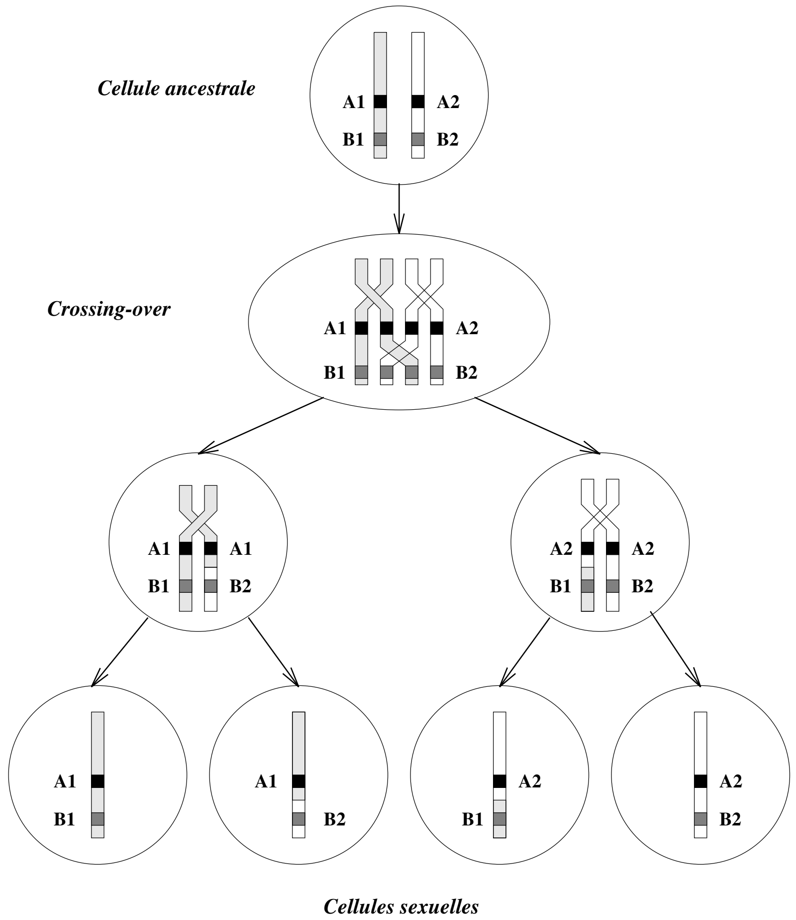


Figure 2.2 - : Formation des cellules sexuelles par recombinaison (crossing-over) pendant la méiose (d'après [White et al.88]). Les deux chromosomes homologues d'un parent comportent deux allèles différents des gènes A et B. Après réplication des chromosomes, ceux-ci échangent des segments entre A et B. Quatre cellules germinales (ou sexuelles) sont obtenues ; deux portent la combinaison allélique parentale et deux contiennent des chromosomes recombinés.

En plus des gènes sont utilisés des marqueurs (i.e. des séquences particulières d'ADN) qui existent également sous des formes différentes et qui sont mis en évidence grâce à des techniques de coupure de segments d'ADN par des enzymes de restriction. Une enzyme coupe un brin d'ADN en des endroits spécifiques fonction de la séquence nucléotidique ; ces sites sont appelés *sites de restriction*. Par exemple, l'enzyme TaqI coupe l'ADN à chaque occurrence de la séquence TCGA. La détermination de ces marqueurs génétiques permet de disposer de repères réguliers le long des chromosomes, pour pouvoir ensuite positionner précisément de nouveaux gènes. Ces polymorphismes de séquence qui provoquent l'apparition ou la disparition de sites de coupure par certaines enzymes de restriction sont appelés *RFLP* pour *restriction fragment length polymorphism* (figure 2.3). Même si l'utilisation d'enzymes de restriction est une étape fondamentale lors de l'élaboration de cartes physiques (Cf. §2.1.3), il s'agit ici de cartes génétiques car elles sont obtenues par analyse de liaison et sont donc liées au phénomène de recombinaison et au polymorphisme.

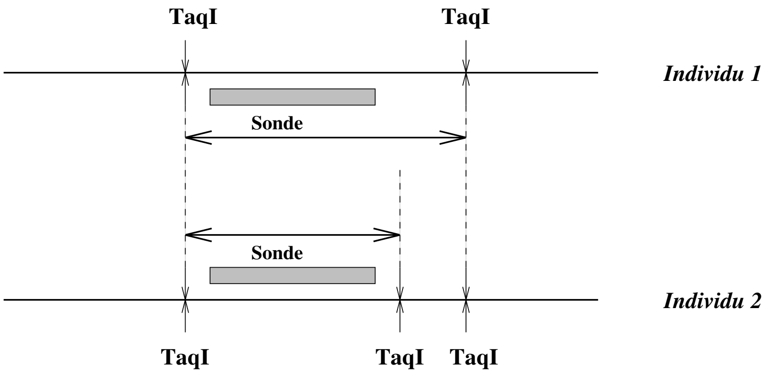


Figure 2.3 - : Utilisation de RFLPs pour l'élaboration d'une carte génétique. L'apparition d'un nouveau site de coupure par l'enzyme de restriction TaqI chez l'individu 2 entraîne la diminution de la longueur d'un des fragments obtenus et mis en évidence par une sonde radioactive. Ce polymorphisme de longueur des fragments de restriction permet de disposer de marqueurs servant à déterminer la position des gènes sur le chromosome.

Enfin, d'autres marqueurs sont utilisés comme les *minisatellites*, répétitions d'un motif de base long d'une vingtaine de nucléotides dont le nombre à une position donnée du génome dépend de l'individu. Plus récemment ont été découverts les *microsatellites*; il s'agit de séquences très simples, répétition d'un motif, qui, chez l'Homme, est le plus souvent CA, d'où le nom de *CA repeats*. Les séquences $(CA)_n$ sont en effet très fréquentes et n varie beaucoup d'un individu à l'autre, ce qui fait des microsatellites des marqueurs extrêmement polymorphiques, donc très informatifs lors des analyses de liaison.

Tous ces marqueurs moléculaires sont mis en évidence par une technique d'analyse des fragments obtenus appelée *Southern blot*. Pour les deux premiers marqueurs mentionnés – les RFLP et les minisatellites, ce sont directement les fragments obtenus par application d'enzymes de restriction. Dans le cas des microsatellites, if faut pouvoir «extirper» du génome un fragment d'une centaine de nucléotides contenant le motif répété, puis en multiplier la quantité à l'aide d'une amplification par PCR (*polymerase chain reaction*), procédé permettant de multiplier jusqu'à un million de fois une petite région d'ADN à partir de deux amorces de part et d'autre de cette région. Ces fragments sont ensuite séparés par Southern blot en fonction de leur longueur en les faisant migrer sur un gel d'agarose grâce à l'application d'un champ électrique (technique d'électrophorèse). Une sonde radioactive spécifique (i.e complémentaire du fragment d'ADN recherché) montre alors, d'un individu à l'autre, les différences de longueur entre les fragments, dues au polymorphisme. Il est clair que le gel à utiliser pour les microsatellites doit être très résolutif car des différences de longueur de deux nucléotides doivent pouvoir être visibles.

Comme le montre la figure 2.2, la moitié au moins des cellules sexuelles conserve le type parental; c'est pourquoi la fréquence de recombinaison (c'est-à-dire le nombre de recombinants divisé par le nombre total de descendants) est inférieur à 50%. En particulier, si deux gènes appartiennent à deux chromosomes différents, ces gènes ne sont pas liés et, statistiquement, la moitié de la descendance présentera des recombinaisons. De même, deux gènes d'un même chromosome, éloignés l'un de l'autre, recombineront en moyenne dans 50% de la descendance. Par contre, deux gènes proches recombineront moins, et la fréquence de recombinaison sera inférieure à 50%.

L'unité de cartographie (exprimée en centiMorgan) est «la distance entre paires de gènes pour laquelle un produit sur cent de la meiose est un recombinant» [Suzuki et al.89]. Cette unité est postulée additive, mais la fréquence de recombinaison ne l'est pas, puisqu'elle ne peut pas dépasser 50%. C'est pourquoi il existe une fonction dite cartographique qui permet de passer de la distance exprimée en centiMorgan à la fréquence de recombinaison (figure 2.4). Malheureusement, cette fonction, même si elle tient compte des crossing-over multiples, qui ne manquent pas de se produire dès que les gènes sont éloignés, suppose que les probabilités d'occurrence des sites de crossing-over suivent un processus de Poisson, ce qui n'est pas le cas. En effet, il n'y a pas indépendance des sites d'échanges de morceaux de chromosomes homologues, mais il se produit un phénomène d'interférence qui fait que l'existence d'un crossing-over à un endroit donné du chromosome diminue la probabilité d'en avoir un autre à proximité. Puisque les distances génétiques sont le reflet des expériences et donc de la fréquence de recombinaison, elles sont telles que l'additivité n'est pas satisfaite, et il est nécessaire, la plupart du temps, de disposer des distances entre entités deux à deux.

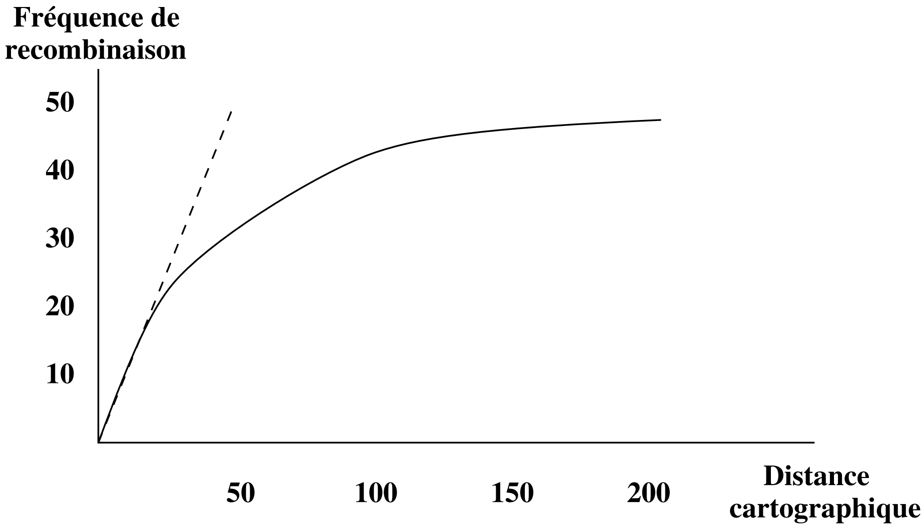


Figure 2.4 - : Fonction cartographique. Cette fonction permet en théorie de faire le passage entre la fréquence de recombinaison (qui ne dépasse pas 50%) et la distance génétique (exprimée en centiMorgan). Cette fonction est linéaire pour des valeurs proches de zéro, c'est-à-dire quand les loci sont liés. La conversion n'est pas parfaite en raison de la non-indépendance des sites de crossing-over les uns par rapport aux autres.

Chez l'animal, les fréquences de recombinaison peuvent être estimées de façon précise en réalisant des croisements-test entre un double *hétérozygote*, c'est-à-dire un organisme pour lequel deux loci homologues portent chacun un allèle différent, et un double *homozygote*, pour lequel les deux loci portent le même allèle[2]. Chez l'Homme, il est indispensable de vérifier la significativité des recombinaisons, par l'utilisation, le plus souvent, d'une fonction dite de «lod score», définie par Morton [Morton88].

Une carte génétique est donc constituée d'un assortiment de gènes et de marqueurs ordonnés par les analyses de liaison. Celles-ci expriment pour différents couples les pour-

[2]Il est également possible de faire des études impliquant trois locus en même temps.

centages de recombinaison exprimés en centiMorgan ; à un centiMorgan correspond environ un million de paires de bases. Cette valeur moyenne peut varier de 5×10^5 à 10^7 paires de bases [Kaplan et al.90]. Une particularité importante des cartes génétiques, qui a déjà été mentionnée et dont il faut se souvenir, est que les fréquences de recombinaison exprimées entre marqueurs ne sont pas additives (ce ne sont donc pas des distances au sens mathématique) ; ainsi, pour trois marqueurs A, B et C dans cet ordre, la distance séparant A et C n'est pas égale à la somme de celles séparant A et B d'une part, B et C d'autre part.

Deux cartes génétiques de l'Homme[3] sont apparues fin 1992 ; la première provient de l'étude des familles gérées par le CEPH et contient de nombreux gènes et des marqueurs divers [Group92] ; la seconde est uniquement constituée de microsatellites régulièrement espacés et s'est progressivement enrichie pour en contenir 2000 [Weissenbach et al.94].

2.1.3 Les cartes physiques

Contrairement à la carte génétique, la cartographie physique ne se base pas sur la transmission de gènes alléliques, mais sur l'étude des jalons contenus physiquement dans l'ADN, que ceux-ci correspondent à des gènes ou non. Les éléments des cartes physiques sont obtenus par des manipulations biochimiques de l'ADN (utilisation d'enzymes de restriction, clonage, séquençage, etc.).

L'hybridation *in situ* est une première étape grossière qui consiste à positionner une sonde radioactive ou fluorescente sur les chromosomes lors d'une des phases de la division cellulaire, la métaphase. Cette sonde s'hybride, c'est-à-dire se lie, avec son brin complémentaire, permettant de la positionner sur le chromosome. La précision obtenue est alors d'une bande ou une demi-bande de la carte cytogénétique.

Comme il est impossible de traiter directement l'ensemble d'un génome, ces cartes sont avant tout basées sur l'utilisation d'enzymes de restriction qui fragmentent l'ADN en morceaux plus petits, plus aisément manipulables. Il est d'abord nécessaire de les séparer en fonction de leur taille ; pour cela, on utilise l'*électrophorèse*, qui les fait migrer sur un gel soumis à un champ électrique ; les molécules les plus grosses se déplacent plus lentement et arrivent donc moins loin que les plus petites. Pour réordonner ces fragments, on les crible avec des sondes, une sonde commune indiquant que deux fragments se recouvrent ; il est également possible de les hybrider entre eux, ou bien d'en établir une carte de restriction, qui consiste à les couper avec différentes enzymes de restriction, puisque les parties communes comporteront les mêmes sites de coupure.

Ensuite, il faut pouvoir disposer de quantités suffisantes des produits. Le *clonage* reproduit en un grand nombre d'exemplaires un fragment d'ADN en le plaçant dans un vecteur de clonage, comme les cosmides, capable de se répliquer dans une bactérie.

Le gros inconvénient des techniques précédentes est qu'elles ne fonctionnent que pour des fragments d'ADN de taille maximale de quelques dizaines de kilobases. Elles induisent ainsi un fossé entre les résultats obtenus grâce à l'analyse de liaison et conduisant à la carte génétique et ceux basés sur l'ADN lui-même (figure 2.5). En particulier, la construction

[3] Remarquons que le sexe intervient sur les distances entre marqueurs génétiques. Ainsi, sur le chromosome 12, la fréquence de recombinaison est supérieure chez la femme. Il faut donc représenter deux cartes génétiques d'un même chromosome, une mâle et une femelle.

de la carte physique qui passe par l'arrangement des morceaux d'ADN, en tirant parti des recouvrements qui existent entre eux, et qui aboutit à la création de *contigs*, va nécessiter un très grand nombre de tels morceaux pour reconstituer ne serait-ce qu'un seul chromosome. Heureusement, de nouvelles techniques sont apparues, permettant de résoudre ce problème.

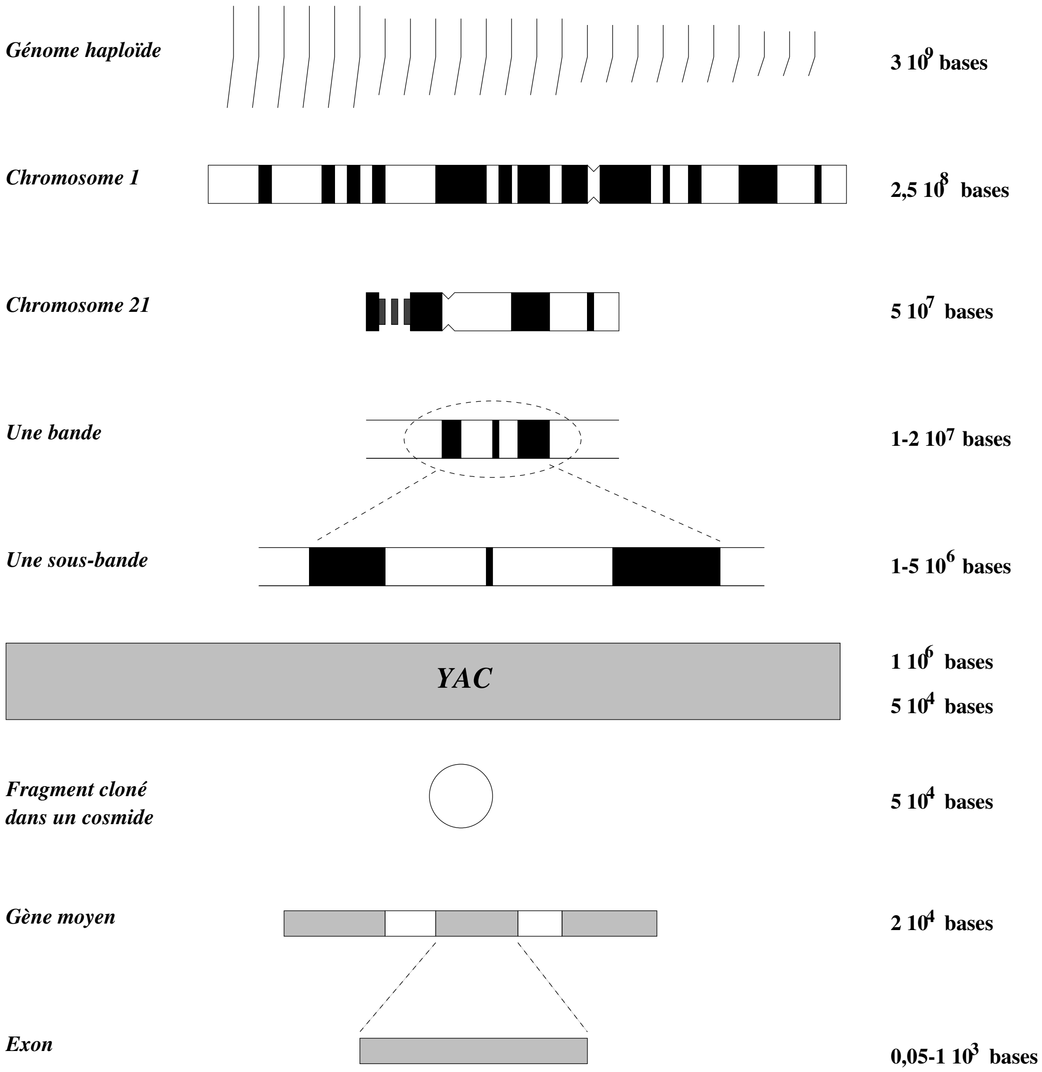


Figure 2.5 - : Le fossé entre l'analyse génétique et l'analyse moléculaire (physique) (d'après [Kaplan et al.90]). Les YAC permettent de combler ce fossé car grâce à eux, il est possible de cloner des fragments d'ADN beaucoup plus grands que ceux dans les cosmides.

L'électrophorèse en champs pulsés est le pendant à l'électrophorèse classique et peut séparer des fragments d'ADN longs de plusieurs mégabases. De tels fragments, étudiés par électrophorèse classique, sont retenues dans le gel par un nombre de pores proportionnel à leur longueur ; comme la force électrique est elle-aussi proportionnelle à la longueur, les molécules d'ADN migrent toutes à la même vitesse. L'électrophorèse en champs pulsés modifie régulièrement l'orientation du champ électrique, ce qui oblige les molécules à se réorienter ; plus elles sont longues et plus elles mettent de temps à le faire.

L'autre technique a été le clonage de grands fragments d'ADN dans des chromosomes artificiels de levure (ou YAC pour *Yeast Artificial Chromosome*). En «déguisant» un fragment d'ADN en chromosome de levure et en l'introduisant dans une cellule de levure, ce fragment sera transmis à la descendance comme les vrais chromosomes de la cellule. La taille des fragments manipulables culmine alors à une, voire deux, mégabases. Malgré les difficultés techniques liées à la manipulation de morceaux d'ADN de grande taille, des banques de YAC se sont progressivement construites. Un problème actuel est la présence dans ces banques de clones dits chimériques, c'est-à-dire formés de la juxtaposition de segments d'ADN provenant de régions différentes du génome.

Enfin, le *séquençage* constitue l'étape ultime de la carte physique puisqu'il aboutit à la suite des nucléotides. Malheureusement, son coût est très élevé, d'autant plus que des erreurs se produisent et qu'une séquence brute demande à être reconfirmée. Le projet de séquençage du génome humain est vu par certains comme inutile, au vu des résultats espérés comparés au coût et au travail demandés, car, d'une part cela risque de se faire au détriment de recherches plus originales et créatives [Jordan89], et d'autre part, il n'y a actuellement pas d'intérêt à séquencer toute la partie non codante des introns [Kaplan et al.90]. Il semble plus intéressant, et plus urgent aussi, de disposer d'un alignement de clones recouvrant une région à étudier, et sur lesquels ont été positionnés des jalons comme les microsatellites, ou les STS (*Sequence Tagged Sites*, c'est-à-dire des séquences uniques dans le génome et qui seraient reproductibles aisément grâce à la PCR à partir des seules extrémités de cette séquence).

2.2 La notion de carte

La section précédente a montré l'existence d'un certain nombre de types de carte. Les descriptions qui en ont été faites permettent de mettre en évidence une certaine concordance entre eux ; on peut en effet décrire ce qu'est une carte *in abstracto*, c'est-à-dire trouver un cadre commun aux trois types de carte.

2.2.1 Le concept biologique de carte

Avant d'aller plus avant dans la description de ce moule commun, il est nécessaire de préciser l'aspect biologique de la carte. Vu l'importance des génomes étudiés (ou des chromosomes seulement), il est clair que très souvent on ne dispose pas d'une carte globale couvrant entièrement un génome ou un chromosome ; c'est pourquoi les biologistes moléculaires travaillent sur des morceaux de carte (qui ne sont bien sûr pas nécessairement positionnés dans une carte de plus haut niveau). De plus, indépendamment de cette contrainte biologique, il peut être intéressant de définir des sous-parties d'une carte importante pour se concentrer sur leur étude, en particulier aussi parce qu'une telle sous-carte peut avoir un sens biologique qui la distingue des autres parties. Néanmoins, au premier abord, rien ne distingue une sous-carte d'une carte (globale) si ce n'est le fait que cette dernière a une identité biologique particulière puisqu'elle recouvre tout un chromosome ou un génome. À un moindre niveau, une sous-carte possède elle aussi une identité propre, expérimentale ou conceptuelle. Quoi qu'il en soit, la structure est conservée d'une carte à ses sous-cartes.

La structure est un élément fondamental du concept de carte. En effet, l'examen des différents types de carte a montré que chaque type est structurellement différent des autres en ce sens que les éléments qui y apparaissent ne sont pas de même nature. Chaque type de carte génomique est constitué de composants qui lui sont propres ; ainsi, une carte génétique est formée de gènes, de RFLP, d'autres marqueurs ; une carte physique se compose également de gènes, mais aussi de séquences, de sites de restriction, etc. ; une carte cytogénétique est constituée de bandes, encore de gènes, etc. Une particularité de la structure des cartes est que n'importe laquelle peut *a priori* se décomposer en sous-cartes. On utilisera désormais le terme *carte* de façon générique en y incluant les sous-cartes, puisqu'il a été établi qu'il n'existait pas de différence structurelle entre les deux. De plus, cela met en évidence la structure récursive du concept. Il est effectivement clair qu'une sous-carte peut elle-même se redécomposer en sous-cartes d'un niveau inférieur.

Une carte n'est évidemment pas seulement l'agrégat de ses composants ; elle comporte aussi un arrangement de ces composants sur l'axe orienté de la carte. Il faut donc ajouter à la notion de carte définie jusqu'à présent une origine et une fin sur cet axe, par rapport auxquelles les composants vont se positionner. Nous aborderons dans la section suivante les problèmes liés au positionnement des composants d'une carte sur son axe. Remarquons que la définition d'une origine et d'une fin implique l'existence d'une orientation, qui provient biologiquement de la structure de l'ADN en double brin ; tout fragment d'ADN peut en effet être lu de deux façons différentes correspondant aux deux brins constitutifs du fragment. Il en est donc de même de tout chromosome, et plus généralement de toute carte. Ceci aura une grande importance lors de la définition des relations entre les constituants de carte (Cf. §2.3).

Pour terminer cette brève énumération des caractéristiques du moule auquel se conforme la notion de carte, il faut ajouter qu'une carte dispose d'une unité. La carte génétique a pour unité le centiMorgan, la carte physique la kilobase, jusqu'à la carte cytogénétique qui possède une unité relative à la taille du chromosome, qui est le pourcentage de longueur du chromosome (c'est ainsi qu'est spécifiée la longueur des bandes).

La figure 2.6 montre une carte typique et met en évidence les éléments mentionnés précédemment (composants, origine, fin, unité).

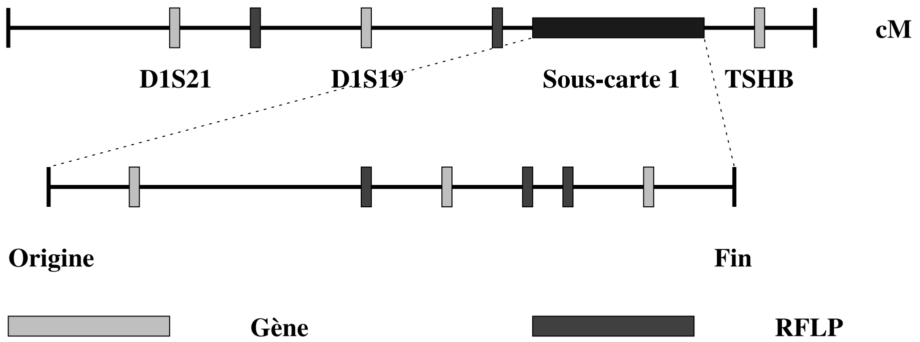


Figure 2.6 - : Description d'une carte typique. Elle est formée d'entités propres, une sous-carte (qui se décompose à nouveau), deux gènes et des RFLP. L'unité est le centiMorgan ; l'origine et la fin sont indiquées.

2.2.2 Les interrelations entre cartes

La section précédente a insisté sur l'intérêt d'une carte intégrée c'est-à-dire telle qu'on puisse passer de l'une à l'autre grâce à des éléments communs. Il est évident que l'existence de points de repères, quels qu'ils soient, partagés par des cartes différentes, autorise à faire le pont entre des résultats d'expérience divers, et de faire profiter une carte des informations d'une autre. C'est pourquoi la découverte de l'électrophorèse en champs pulsés et de l'utilisation de chromosomes de levure pour le clonage a revêtu une si grande importance. Ce sont ces éléments communs qui autorisent les interrelations entre cartes.

La description faite dans le paragraphe précédent ne contredit en rien cette possibilité, puisqu'il suffit qu'une carte puisse avoir pour composant une même entité qu'une autre carte pour créer ce lien. Par exemple, les gènes sont un pont autorisé entre l'ensemble des trois types de carte ; ils appartiennent effectivement à chacune des cartes.

Ces composants partagés ont néanmoins en général des représentations différentes selon la carte considérée. Par exemple, un gène sur une carte génétique est une entité ponctuelle, représentée par un trait, tandis que sur une carte physique, le même gène possède une longueur exprimable en nombre de bases.

De plus, de la même manière que les cartes se décomposent en éléments constitutifs, leurs constituants peuvent se décomposer également. Ainsi, un gène sur une carte physique est composé d'introns, d'exons, de sites d'initiation et de terminaison, etc. (figure 2.7) ; en ce sens, c'est également une carte puisqu'il sert de référence pour y placer des entités. Cette décomposition peut, comme celle des cartes, être récursive ; un exemple d'une telle situation apparaît pour les bandes cytogénétiques qui se divisent en sous-bandes. Plus encore, une même entité peut d'un type de carte à l'autre se décomposer en entités différentes, le meilleur exemple étant celui des cartes elles-mêmes, dont les éléments constitutifs dépendent, comme cela a été vu, du type de la carte.

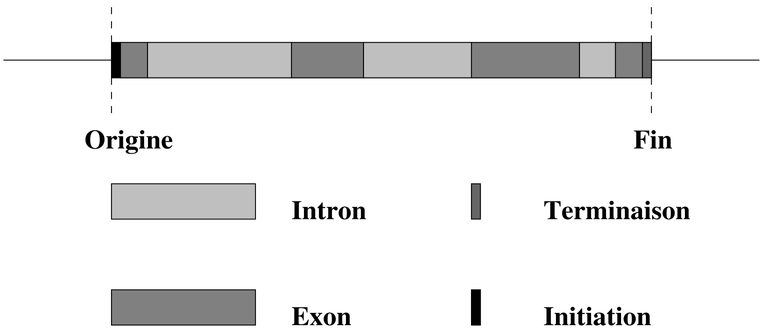


Figure 2.7 - : Décomposition d'un gène sur une carte physique. Les constituants d'une carte peuvent eux-mêmes se décomposer en sous-éléments ; ainsi, un gène d'un organisme eucaryote sur une carte physique est constitué de sites d'initiation et de terminaison, d'introns et d'exons.

On arrive à une définition de la carte qui n'était peut-être pas évidente *a priori*, qui est qu'*une carte est une entité sur laquelle se positionnent des constituants*. L'appellation de carte en biologie moléculaire peut être légèrement différente puisqu'elle n'inclut classiquement que les représentants de plus haut niveau (c'est-à-dire les chromosomes) ou des sous-parties structurellement identiques (même si certains biologistes peuvent considérer un gène, par exemple, comme une carte). Nous conserverons par la suite cette sémantique, étant donné que les autres entités qui acceptent des composants ont d'autres noms qui les définissent mieux ; il est malgré tout important de conserver en mémoire cette définition *stricto sensu* du terme carte.

2.3 Les relations entre éléments de cartes

La section précédente a montré les particularités descriptives des cartes génomiques sans aborder le problème de leur dynamicité (voir aussi à ce propos la section 2.4). Une carte a besoin d'être construite, et, pour cela, on se doit d'exprimer des états incomplets et incertains. On peut même douter de l'existence d'une carte ultime qui ne nécessiterait plus d'être modifiée. Construire une carte, cela signifie récolter les fruits d'expériences qui mettent en évidence des relations entre les entités qui appartiennent à cette carte. Par exemple, une analyse de liaison va montrer que deux gènes sont séparés par une certaine distance génétique (sur laquelle peut se greffer une part d'incertitude) ou que trois gènes sont ordonnés de telle manière. De façon générale, il s'agit de relations qualitatives ou quantitatives, qui conduisent à exprimer soit des ordres, soit des distances entre les entités biologiques manipulées [Graves93, Guidi et al.93]. Nous allons les examiner à tour de rôle.

2.3.1 Les relations qualitatives

Une difficulté majeure lors de l'expression de ces relations est celle de l'orientation [Lee et al.93, Letovsky et al.92]. Quand par exemple on affirme que le gène A est avant le gène B, sur quelle orientation est fondée le «avant» ? Sur celle, globale, du chromosome ? Sur celle, locale, de l'entité qui contient les deux gènes (si tant est qu'une telle entité existe[4]) ? Ce problème sera abordé plus en détail par la suite lors de la formalisation (Cf. §3.4).

La description des relations entre entités biologiques appartenant à des cartes génomiques est un sujet difficile. Même si, à première vue, on remarque une correspondance évidente avec des travaux sur la représentation du temps [Guidi et al.93], on se rend compte que ceux-ci s'en distinguent sur de nombreux points. En effet, quand bien même il existerait un isomorphisme entre une carte génomique sur laquelle on place des entités biologiques et un axe temporel où on positionne des intervalles de temps et des instants, l'adaptation ne serait pas sans risque. Trop offrir n'est pas toujours un avantage et, plutôt que de perdre un biologiste dans les méandres des nombreuses (2^{13}) relations temporelles définies par Allen [Allen83], il vaut mieux ne lui proposer que les relations qui lui sont utiles (comme cela est fait par exemple dans [Lee et al.93] pour la construction de contigs).

De plus, pour des raisons de complexité, la résolution de problèmes de satisfaction de contraintes temporelles impose de se restreindre au niveau de l'expressivité des relations [Freksa92, Vilain et al.86]. Enfin, les liaisons entre les différentes cartes, le besoin de gérer les incertitudes, en bref, les spécificités du problème biologique imposent des modifications. Nous verrons plus en détail les analogies utiles entre la modélisation des relations dans les cartes génomiques et la représentation du temps au chapitre 6. Les relations intéressantes (i.e. qu'un biologiste a envie d'exprimer) ne sont pas très nombreuses ; en plus

[4] En effet, rien n'empêche de penser qu'il puisse exister plusieurs entités qui les contiennent, par exemple le chromosome et une sous-carte.

de certaines relations qui apparaissent dans l'algèbre d'intervalles d'Allen et dont nous aurons à préciser la sémantique (par exemple, en ce qui concerne la pertinence des inégalités strictes), se trouvent deux autres relations qui se rattachent au problème d'orientation soulevé précédemment ; il s'agit des relations *ordre*, et du couple *même_orientation* et *orientation_opposée* (figure 2.8).

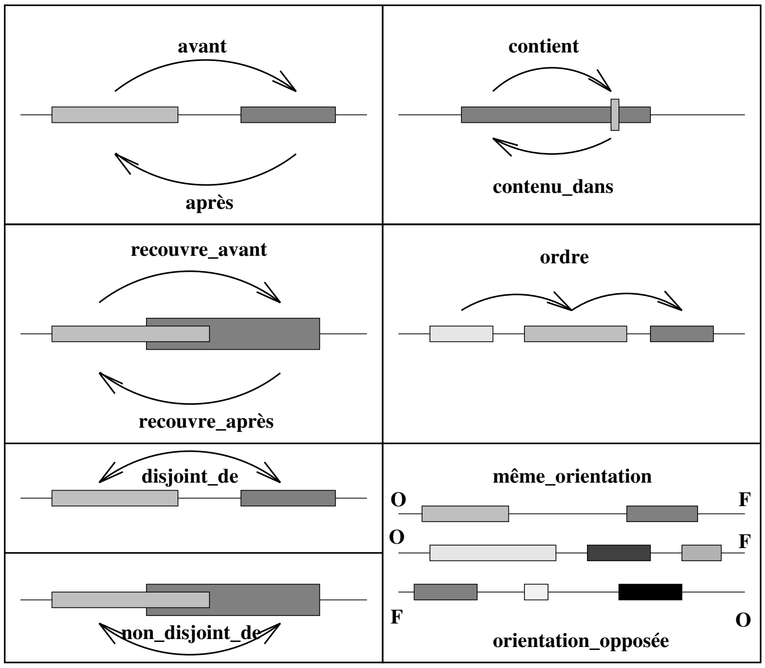


Figure 2.8 - : L'ensemble des relations entre entités biologiques. Elles sont issues d'expériences biologiques, telles que des hybridations, des analyses de liaison ou sont le résultat d'inférences à partir de ces expériences.

Les relations concernant l'orientation sont fondamentales car elles permettent d'expliciter de la connaissance implicite entre entités de cartes. En particulier, elles ont un rôle de propagation des ordres locaux, dans le but d'obtenir au bout du compte un ensemble d'entités ordonnées le plus grand possible ; la figure 2.9 montre les conséquences qu'on peut tirer d'une telle relation entre deux sous-cartes.

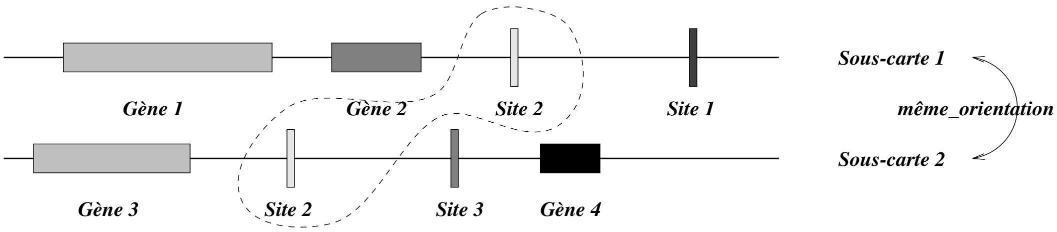


Figure 2.9 - : Utilisation de la relation de même orientation. Les deux sous-cartes sont constituées d'entités ordonnées et sont liées par une relation de même orientation. L'existence d'un élément commun à ces sous-cartes, *Site 2*, permet alors d'inférer des relations d'ordre entre les éléments de part et d'autre de *Site 2* sur chacune des cartes. Ainsi, on peut affirmer que *Gène 1* et *Gène 2* sont avant *Site 3* et *Gène 4* et que *Gène 3* est avant *Site 1*. Par contre, on ne peut rien dire sur la position relative de *Gène 1* et *Gène 3*.

2.3.2 Les relations quantitatives

En ce qui concerne les relations quantitatives, deux notions sont à considérer, celle de distance et celle d'incertitude. Typiquement, il faut pouvoir exprimer quelque chose comme «le gène D1S10 est situé à 6.3 cM du gène D1S21 avec une incertitude de 0.4 cM». De plus, le fait de disposer de plusieurs cartes, qui ne sont peut-être pas liées par un facteur d'échelle, impose d'exprimer plusieurs distances, au plus une par type de carte. Il suffit de considérer les cartes génétique et physique pour s'en convaincre. De même, les cartes physique et cytogénétique ne sont pas liées par un facteur d'échelle global, même au seul niveau d'un chromosome (figure 2.10).

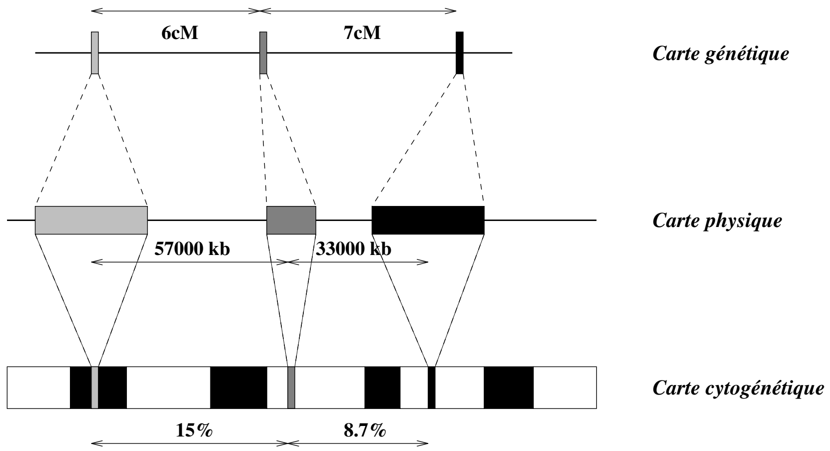


Figure 2.10 - : Multiplicité des distances. Les trois gènes sont séparés par trois distances différentes selon la carte génomique. Il n'y a pas d'échelle fixe entre la carte génétique, la carte physique et la carte cytogénétique. Seuls des correspondances approximatives peuvent être établies, qui ne sont donc pas fiables.

2.4 Les opérations effectuées sur les cartes

Les opérations effectuées sur une carte génomique (ou plus précisément sur un ensemble de cartes génomiques) se séparent en deux groupes : les opérations dynamiques et les opérations statiques. Dans le premier cas, il s'agit de construire une carte à partir d'expériences biologiques qui ont permis d'établir des relations, des ordres partiels entre entités de la carte. Dans le second, les cartes sont supposées avoir atteint un niveau de description suffisamment stable pour qu'on puisse les considérer comme représentatives d'un consensus. On cherche alors à utiliser cette représentation comme un outil de travail en lui appliquant des traitements ou en activant des requêtes, voire simplement en visualisant les cartes et en circulant d'une entité à l'autre grâce aux liens qui les relient.

2.4.1 La construction de cartes

La construction d'une carte nécessite de développer les relations qui ont été exprimées entre les entités constitutives de cette carte de façon à mettre en évidence des sous-ensembles pour lesquels ces entités sont complètement ordonnées. Étant donné l'importance des expériences dans l'établissement de ces relations, il est indispensable de gérer les incohérences éventuelles qui risquent d'apparaître.

Il peut apparaître par exemple des croisements d'une carte à l'autre, dus le plus souvent à de mauvaises estimations de distances génétiques par manque de familles sur lesquels faire une étude statistique, en particulier chez les espèces pour lesquelles on ne peut pas réaliser expérimentalement les croisements (typiquement, l'Homme).

Dans son principe, une telle construction revient à un problème de satisfaction de contraintes identique à celui qu'on aurait dans un réseau de contraintes temporelles mais avec beaucoup moins de relations à gérer. Les travaux de Letovsky et Berlyn [Letovsky et al.92] et ceux de Lee *et al.* [Lee et al.93] utilisent tous les deux des règles d'inférence pour exhiber des ensembles de plus en plus importants d'entités ordonnées localement, les premiers dans le but de construire des cartes génétiques, les seconds pour l'assemblage de fragments d'ADN en contigs. Nous étudierons lors de la formalisation des requêtes (Cf. chapitre 4) l'aspect algorithmique de la résolution de l'ordonnancement des entités biologiques à partir des relations exprimées entre elles ; cet aspect sera étudié plus en détail dans le chapitre 7 grâce à l'utilisation de techniques de raisonnement temporel. La description de ces algorithmes dépend fondamentalement des relations qu'il est possible d'exprimer ; nous nous limiterons alors à celles spécifiées dans la section précédente.

2.4.2 Les opérations statiques

Une carte consensus se trouve dans un état relativement stable, elle n'en est pas moins encore très utile, car elle constitue une représentation structurée de la connaissance disponible, et sur laquelle diverses opérations sont possibles.

La plus simple de ces opérations est la visualisation des cartes et des entités qui leur appartiennent, le parcours des liens entre ces entités, etc. La récupération d'informations sur les cartes est fortement dépendante de l'interface qui la facilite ; nous verrons au chapitre 14 quelles sont les fonctionnalités d'une telle interface. Ensuite, toutes sortes de requêtes peuvent leur être appliquées. Par exemple, la recherche des constituants d'une (sous-)carte permet de récupérer (éventuellement récursivement) ses entités ainsi que leur ordre.

D'autres requêtes concernent davantage les relations permettant de définir les ordres des entités les unes par rapport aux autres. Il s'agit alors de récupérer l'information, exprimée le plus souvent de façon implicite dans les relations, pour tenter de répondre à la requête. Ce genre de requête est à rapprocher de la construction de la carte, puisque cette dernière est le fruit d'un certain nombre de requêtes similaires.

Chapitre 3

Abstraction des cartes génomiques

Nous allons définir désormais une abstraction de la description du concept de carte génomique, c'est-à-dire une représentation formelle et générique (i.e. indépendante du domaine considéré jusqu'alors). Nous verrons après cela de quelle manière cette abstraction s'instancie pour s'appliquer aux cartes génomiques. Nous conserverons le terme de carte, sachant qu'un autre domaine applicatif choisirait un autre terme (par exemple, en représentation temporelle, on parlerait plutôt d'axe du temps ou d'axe temporel). Mais avant cela, il est nécessaire de justifier notre démarche.

3.1 Pourquoi formaliser la notion de carte?

Plusieurs éléments entrent dans la justification d'une formalisation des cartes génomiques, qui peuvent grossièrement se diviser selon deux aspects, un aspect portant sur des considérations de représentation de connaissances, un autre plus orienté vers l'algorithmique et le raisonnement. De manière générale, les travaux sur les cartes concernant ces deux aspects se limitent au développement de systèmes *ad hoc* et ne profitent justement pas des avantages qu'aurait entraînés une formalisation.

D'abord, toute formalisation entraîne une structuration du problème qui aide à sa compréhension. L'aspect «génie logiciel» intervient beaucoup en informatique, et l'élaboration d'un système informatique passe obligatoirement par une phase de spécification. La formalisation est le prélude aux spécifications d'un outil informatique à même de répondre aux interrogations liées au problème.

De plus, formaliser, on va le voir dans les sections suivantes, va définir naturellement de manière déclarative les éléments de la représentation. Ayant spécifié ces éléments-là, la cohérence des données est énormément améliorée, car un utilisateur n'a plus la possibilité d'écrire n'importe quoi, puisque ses actions doivent rester dans le cadre défini par la formalisation.

Enfin, l'extensibilité du modèle est d'autant plus facile que celui-ci a été bien défini, ce qui est directement lié à l'abstraction, plus qu'à la formalisation. Le fait d'avoir établi précisément les entités du problème permet d'en rajouter aux endroits adéquats. Il est beaucoup plus difficile de faire des extensions à un programme, et surtout de comprendre les modifications introduites sur le fonctionnement et le déroulement correct de ce programme.

Concernant le raisonnement, la formalisation permet le développement d'algorithmes plus génériques, en ce sens qu'il est possible de les étendre sans avoir à tout reprendre du début. De plus, il est beaucoup plus aisé de déterminer les caractéristiques de ces algorithmes (correction, complétude, complexité), même après les avoir étendus. C'est par exemple un des gros avantages des algorithmes de satisfaction de contraintes, qui partent d'un modèle d'expression de contraintes pour aboutir à une résolution du problème contraint, et dont la construction peut se faire de façon incrémentale, en ajoutant au fur et à mesure de nouvelles fonctionnalités, du moment que la sémantique des ajouts est intégrée à l'algorithme. On aboutit ainsi à une modularité qui facilite le développement, la maintenance et l'extension des algorithmes.

Une justification peut-être plus proche de l'application aux cartes est que le développement d'une interface homme-machine, indispensable si on veut pouvoir utiliser efficacement les connaissances cartographiques, doit suffisamment «coller» au modèle pour ne pas permettre à un utilisateur toutes les actions possibles. La structuration de l'interface cartographique dépend de celle qu'a fournie la formalisation. Ici encore, des considérations d'extensibilité interviennent. Il est clair que si l'interface repose sur un modèle sous-jacent proprement défini, un ajout minime (comme, par exemple, l'extension d'un ensemble d'éléments) ne va pas modifier l'interface car elle profitera des comportements déjà définis.

3.2 La typologie

Cette section définit de façon arbitraire un certain nombre d'ensembles dans le but d'abstraire les cartes génomiques et de mettre ainsi en évidence ce qu'on pourrait un peu pompeusement appeler l'*ontologie* du concept de carte.

Soit $\mathcal{P}$ l'ensemble des points de vue sous lesquels représenter les cartes ; ces points de vue correspondent aux trois types de carte génomique (cytogénétique, génétique et physique). Soit $\mathcal{T}$ l'ensemble de tous les types des entités qui sont susceptibles d'apparaître sur ces cartes, auquel est ajouté un type pré-défini appelé *carte*, ce qui donne

$$\mathcal{T}^+ = \mathcal{T} \cup \{carte\}.$$

Ces types sont organisés en un arbre de composition dans lequel est associé à un type un ensemble de types constitutifs, éventuellement vide. Sous-carte, gène, séquence, bande, site de restriction, intron, etc., sont les types biologiques qui vont apparaître sur les différentes cartes. L'arbre de composition spécifie, pour chaque point de vue, qu'un de ces types est constitué d'autres entités. Par exemple, une (sous-)carte est formée de gènes, de séquences, etc., dans le point de vue physique ; un gène est composé d'introns et d'exons dans ce même point de vue.

Soit ensuite $\mathcal{D}$, un ensemble de dimensions, constitué de deux éléments[1], *point* noté $\bullet$ et *intervalle* noté $\leftrightarrow$, sur lequel existe la relation d'ordre $\bullet < \leftrightarrow$. Les éléments de cet ensemble

[1] Il est envisageable d'étendre cet ensemble pour y incorporer des dimensions plus compliquées comme les unions d'intervalles, de la même manière que des travaux de représentation temporelle traitent de l'union d'intervalles de temps [Ladkin86] ; cette possibilité d'extension donne son sens à l'ordre partiel défini sur cet ensemble.

permettent de spécifier chaque type dans un point de vue, en décrivant sa longueur ; un type dont la dimension dans un point de vue est un point n'aura pas de longueur (et sera représenté sur la carte par un point), tandis qu'une dimension d'intervalle indiquera une longueur et une représentation par un segment.

Enfin, on définit $\mathcal{U}$ comme l'ensemble des unités correspondant aux points de vue, et $\mathcal{U}^+$ obtenu par l'adjonction d'un élément dénotant l'absence d'unité noté $\odot$:

$$\mathcal{U}^+ = \mathcal{U} \cup \{\odot\}.$$

Définissons alors des fonctions qui s'appliquent à ces ensembles :

$$\begin{array}{llcl}
const: & \mathcal{T}^+ \times \mathcal{P} & \rightarrow & P(\mathcal{T}^+) \\
 & (t,p) & \mapsto & const(t,p), \\
dim: & \mathcal{T}^+ \times \mathcal{P} & \rightarrow & \mathcal{D} \\
 & (t,p) & \mapsto & dim(t,p), \\
unit: & \mathcal{P} & \rightarrow & \mathcal{U}^+ \\
 & p & \mapsto & unit(p), \\
add: & \mathcal{U} & \rightarrow & \{vrai, faux\} \\
 & u & \mapsto & add(u), \\
scale: & \mathcal{U} \times \mathcal{U} & \rightarrow & I\!R^{+*} \\
 & (u_1,u_2) & \mapsto & scale(u_1,u_2).
\end{array}$$

La fonction *const* associe à un type dans un point de vue l'ensemble des types constitutifs de ce type dans ce point de vue ($P(\mathcal{A})$ désigne les parties de $\mathcal{A}$, c'est-à-dire l'ensemble des sous-ensembles de $\mathcal{A}$). Elle définit donc l'arbre de composition mentionné dans le paragraphe précédent. La fonction *dim* donne, pour tout couple (type, point de vue), la dimension de ce type dans ce point de vue. La fonction *unit* lie une unité à un point de vue, et *add* indique si une unité est additive, c'est-à-dire si elle représente bien une distance. C'est le cas pour l'unité *kilobase* du point de vue physique, mais pas du *centiMorgan* du point de vue génétique. Enfin est éventuellement associé à un couple d'unités un facteur d'échelle tel que $scale(u_1, u_2)$ représente le coefficient pour passer d'une valeur exprimée dans l'unité u_1 à une valeur dans l'unité u_2 (globalement sur l'ensemble des deux cartes). Ces cinq fonctions ont leur définition en extension ; il est nécessaire de les valuer pour tout élément de leur domaine de définition. Elles doivent néanmoins vérifier les propriétés suivantes :

1. Une carte peut être composée d'autres cartes quel que soit le point de vue.
$$\forall p \in \mathcal{P}, const(carte, p) \ni carte.$$
2. Une carte est toujours considérée comme un intervalle.
$$\forall p \in \mathcal{P}, dim(carte, p) = \leftrightarrow .$$
3. La dimension d'un constituant est plus petite que celle de l'entité constitutive (inutile avec la propriété 5 et si $\mathcal{D}$ se limite à deux éléments).
$$\forall p \in \mathcal{P}, \forall t \in \mathcal{T}^+, t' \in const(t,p), dim(t',p) \preceq dim(t,p).$$

4. Si un type n'est pas un constituant d'un autre type dans un point de vue donné, alors sa dimension n'est pas définie. Réciproquement, si la dimension d'un type n'est pas définie dans un point de vue donné, alors aucune entité ne l'a comme constituant dans ce point de vue.

$$\forall t \in \mathcal{T}^+, \forall p \in \mathcal{P}, t \notin \bigcup_{t' \in \mathcal{T}^+} const(t', p) \Leftrightarrow (t, p) \notin \mathcal{D}_{def}(dim).$$

5. Un type ayant une dimension ponctuelle dans un point de vue ne peut pas se décomposer dans ce point de vue.

$$\forall p \in \mathcal{P}, t \in \mathcal{T}, dim(t, p) = \bullet \Rightarrow const(t, p) = \emptyset.$$

6. Si un type se décompose dans un point de vue, alors il est un constituant d'un type (éventuellement le même) dans ce même point de vue (si ce n'en était pas un, à quoi servirait-il de le décomposer puisqu'il n'apparaîtrait jamais dans ce point de vue ?).

$$\forall t \in \mathcal{T}^+, \forall p \in \mathcal{P}, const(t, p) \neq \emptyset \Rightarrow \exists t' \in \mathcal{T} \text{ tel que } t \in const(t', p).$$

7. La fonction *scale* est réflexive.

$$(u_1, u_1) \in \mathcal{D}_{def}(scale) \wedge scale(u_1, u_1) = 1.$$

8. Tout élément appartenant au domaine de définition de *scale* possède un inverse.

$$scale(u_1, u_2) = s_{12} \Rightarrow (u_2, u_1) \in \mathcal{D}_{def}(scale) \wedge scale(u_2, u_1) = 1/s_{12}.$$

9. Le facteur d'échelle *scale* possède la propriété de transitivité.

$$\begin{aligned} & scale(u_1, u_2) = s_{12} \wedge scale(u_2, u_3) = s_{23} \\ & \quad \Rightarrow (u_1, u_3) \in \mathcal{D}_{def}(scale) \wedge scale(u_1, u_3) = s_{12} \times s_{23}. \end{aligned}$$

3.3 Les cartes elles-mêmes

Après une description conceptuelle générale des entités qui forment les cartes se pose la question de ce qu'est une carte particulière. Soit alors $\mathcal{E}_t$, l'ensemble des entités de type t, pour tout $t \in \mathcal{T}^+$ et $\mathcal{E}$ défini par

$$\mathcal{E} = \cup_{t \in \mathcal{T}^+} \mathcal{E}_t.$$

Associée à cette définition est la fonction suivante :

$$\begin{array}{rccc} type: & \mathcal{E} & \rightarrow & \mathcal{T}^+ \\ & e & \mapsto & t. \end{array}$$

Les éléments de ces ensembles sont les noms (ou identificateurs) des entités du type considéré. Il leur est associé une description par l'intermédiaire d'une fonction *desc*, dont la définition est liée à celle de la fonction *const* (qui donne pour chaque couple (type, point de vue) l'ensemble des types constitutifs de ce type dans ce point de vue).

$$\begin{array}{rccc} desc: & \mathcal{E} \times \mathcal{P} & \rightarrow & P(\mathcal{E}) \\ & (e, p) & \mapsto & \{e_1, e_2, \ldots, e_n\}, \end{array}$$

avec la propriété que

$$\forall p \in \mathcal{P}, \forall e \in \mathcal{E}, \forall e' \in desc(e, p), type(e') \in const(type(e), p),$$

qui impose que les constituants de l'élément e aient un type en accord avec la spécification de la fonction *const*.

Enfin, on crée également une fonction *orig* et une fonction *fin* qui donnent respectivement l'origine et la fin d'une entité ; elles prennent leur valeur dans un ensemble de positions noté *Pos*. Les deux fonctions sont bijectives de $\mathcal{E}$ dans Pos^- et Pos^+ respectivement ; si une entité a une dimension nulle dans tous les points de vue où elle est définie, son origine et sa fin coïncident, tout en étant distinctes. La fonction *entit* permet de passer d'une position, origine ou fin, à l'entité à laquelle elle appartient (figure 3.1).

$$\begin{array}{rccc} orig: & \mathcal{E} & \rightarrow & Pos^- \\ & e & \mapsto & position \\ fin: & \mathcal{E} & \rightarrow & Pos^+ \\ & e & \mapsto & position \\ entit: & Pos & \rightarrow & \mathcal{E} \\ & position & \mapsto & e \end{array}$$

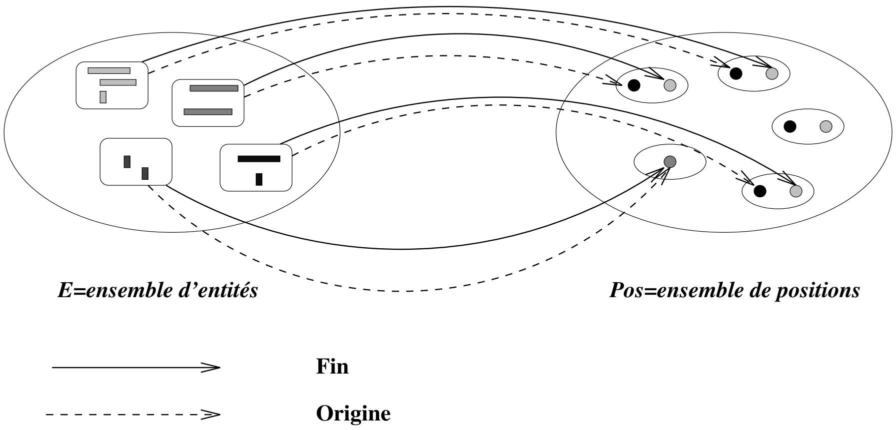


Figure 3.1 - : Description des fonctions *orig* et *fin*. Toute entité possède une origine et une fin appartenant à un ensemble de positions ; elles sont confondues si la dimension de l'entité est nulle dans tous les points de vue où elle est définie.

Ces fonctions permettent d'une part la spécification d'une orientation propre à une entité, allant de son origine à sa fin, d'autre part de définir les distances séparant les entités les unes des autres.

3.4 Les relations qualitatives

Soit, par définition, $\mathcal{Q}_l$ l'ensemble des types de relation qualitative (i.e. non numériques). Chaque élément de $\mathcal{Q}_l$ constitue un moule générique s'instanciant pour construire des relations particulières entre entités. Soit alors la fonction *arit* qui à tout élément de $\mathcal{Q}_l$ associe ses arités possibles, c'est-à-dire un ensemble de nombres d'éléments liés par cette relation (il est en effet possible qu'une relation lie un nombre variable d'entités, même si la plupart ont une arité réduite à un seul élément).

$$\begin{array}{rccl} arit: & \mathcal{Q}_l & \rightarrow & P(I\!N^*) \\ & t & \mapsto & arit(t). \end{array}$$

Parmi l'ensemble des relations qualitatives considérées, il est nécessaire de distinguer les types de relation qui expriment un ordre (et qui ont conséquemment besoin d'une référence) de ceux exempts de cette contrainte. On a vu en effet (Cf. §2.3.1) que des relations comme *avant* ou *après* ont besoin d'une entité de référence parce que la molécule d'ADN autorise deux sens de lecture ; une telle relation nécessite donc d'être projetée sur une entité donnant l'orientation.

Soient alors $\mathcal{Q}_l^+$ l'ensemble des types de relation qualitative orientée et $\mathcal{Q}_l^-$ les autres. Ces deux ensembles forment une partition de $\mathcal{Q}_l$. Nous verrons plus loin (Cf. §4.1), lors de l'application du formalisme au problème des cartes génomiques, quels sont les éléments de ces ensembles.

De la même manière, l'ensemble des relations qualitatives valides (i.e. tous les éléments de cet ensemble représentent des relations qualitatives vraies, qu'elles aient été énoncées comme des faits, ou qu'elles soient le fruit d'inférences à partir des faits et en utilisant les mécanismes d'inférence de la section 4.2), noté $\mathcal{R}_{quali}$, se décompose en deux sous-ensembles, l'un contenant les relations dites orientées et l'autre pas :

$$\mathcal{R}_{quali} = \mathcal{R}_{quali}^+ \cup \mathcal{R}_{quali}^-.$$

Une relation qualitative orientée est alors un triplet formé d'un élément de $\mathcal{Q}_l^+$, d'un ensemble de deux entités liées par cet élément (car c'est une relation binaire) et d'une entité dont les entités précédentes sont des constituants. Il est en effet indispensable que les éléments liés par la relation appartiennent à une même entité qui définit une orientation locale pour la relation ; comme cette entité peut ne pas être unique, il faut la spécifier explicitement. Une relation qualitative non orientée est un couple formé des deux premiers éléments donnés pour la définition des relations qualitatives orientées. Formellement,

$$\mathcal{R}_{quali}^+ \subseteq \mathcal{Q}_l^+ \times (\mathcal{E}, \mathcal{E}) \times \mathcal{E},$$

avec

$$\forall r = [rel, E, e] \in \mathcal{R}_{quali}^+, \exists p \in \mathcal{P} / \forall e' \in E, e' \in desc(e, p).$$

Ceci indique qu'on ne peut comparer des entités que si elles appartiennent à une même entité d'un même point de vue (néanmoins, les entités de E peuvent se retrouver dans plus d'un point de vue). Ceci suppose implicitement que les relations qualitatives sont conservées d'un point de vue à l'autre. Nous reviendrons sur cette hypothèse par la suite.

$$\mathcal{R}^-_{quali} \subseteq \mathcal{Q}^-_l \times (\mathcal{E}, \mathcal{E}),$$

avec

$$\forall r = [rel, E] \in \mathcal{R}^-_{quali}, \exists p \in \mathcal{P} / \forall e \in E, e \in \bigcup_{e' \in \mathcal{E}} desc(e', p).$$

Dans ce cas aussi, il faut qu'il existe un point de vue regroupant toutes les entités mises en relation.

Avant d'aller plus loin, nous allons introduire une nouvelle notation plus condensée exprimant l'appartenance d'une relation à $\mathcal{R}_{quali}$:

$$\begin{aligned}
&\forall t \in \mathcal{Q}^+_l \text{ tel que } arit(t) = \{2\}, (e_1\, t_e\, e_2) \stackrel{def}{\Longleftrightarrow} [t, (e_1, e_2), e] \in \mathcal{R}^+_{quali};\\
&\forall t \in \mathcal{Q}^-_l \text{ tel que } arit(t) = \{2\}, (e_1\, t\, e_2) \stackrel{def}{\Longleftrightarrow} [t, (e_1, e_2)] \in \mathcal{R}^-_{quali};\\
&\forall t \in \mathcal{Q}^-_l \text{ tel que } 3 \in arit(t), t(e_1, \ldots, e_n) \stackrel{def}{\Longleftrightarrow} [t, (e_1, \ldots, e_n)] \in \mathcal{R}^-_{quali}.
\end{aligned}$$

Que r soit une relation de $\mathcal{R}^+_{quali}$ ou de $\mathcal{R}^-_{quali}$, le nombre des entités liées par r doit correspondre à celui défini par la fonction $arit$:

$$\forall r = [rel, E\{, e\}] \in \mathcal{R}_{quali}, card(E) \in arit(rel).$$

Enfin, toutes les relations binaires ont un inverse défini par :

$$\forall t \in \mathcal{Q}^+_l, (e_1\, t_e\, e_2) \Leftrightarrow (e_2\, t^{-1}_e\, e_1),$$

$$\forall t \in \mathcal{Q}^-_l, (e_1\, t\, e_2) \Leftrightarrow (e_2\, t^{-1}\, e_1).$$

De plus, l'inverse d'une relation qualitative orientée en est aussi une. De même, l'inverse d'une relation de $\mathcal{Q}^-_l$ appartient à $\mathcal{Q}^-_l$. Pour des raisons de cohérence, les inverses appartiennent à ces ensembles même si les algorithmes, pour des raisons de simplicité, pourront n'utiliser qu'un élément du couple (relation, relation inverse), sachant que chacune peut s'exprimer en fonction de l'autre grâce à la relation de définition.

3.5 Les relations quantitatives

Les relations quantitatives expriment des distances entre les positions des entités, qui peuvent différer selon le point de vue, s'il n'existe pas de facteur d'échelle entre eux (figure 3.2).

Appelons $\mathcal{R}_{quanti}$ l'ensemble des relations quantitatives valides, c'est-à-dire, comme pour les relations qualitatives, les relations énoncées comme des faits ou celles inférées ; un élément de $\mathcal{R}_{quanti}$ est un quadruplet liant deux positions à une distance et une incertitude[2], dans un point de vue donné. On se limitera dans cette formalisation à exprimer des incertitudes absolues, i.e. non liées à la distance qui sépare les éléments.

[2]Cette incertitude correspond à la marge de déplacement d'une extrémité d'un côté ou de l'autre de sa position médiane ; elle vaut donc la moitié de la valeur indiquée par la double flèche de la figure 3.2.

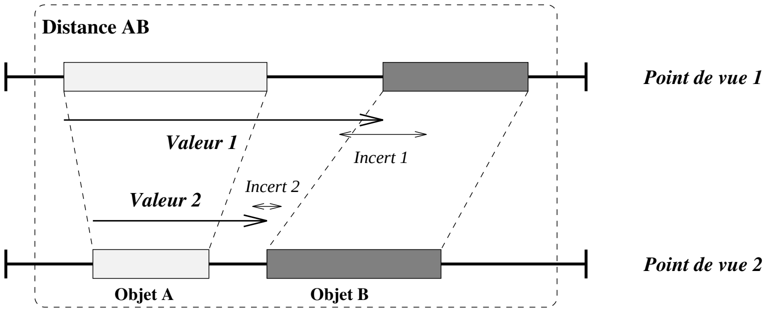


Figure 3.2 - : Deux valeurs pour une même distance. La distance séparant les objets A et B diffère selon le point de vue ; il en est de même de l'incertitude sur cette distance.

La correspondance avec des incertitudes relatives est immédiate ; à partir d'une incertitude relative, l'incertitude absolue correspondante s'obtient en multipliant la distance par l'incertitude (figure 3.3).

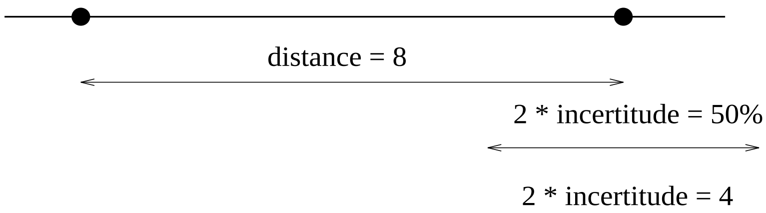


Figure 3.3 - : Passage d'une incertitude relative à une incertitude absolue. La relation suivante les lie : $incert_{rel} = incert_{abs}/d$.

$$\mathcal{R}_{quanti} \subseteq (Pos, Pos) \times I\!R^+ \times I\!R^+ \times \mathcal{P},$$

où

$$\forall r = [P, dist, incert, p], \forall pos \in P, entit(pos) \in \bigcup_{e \in \mathcal{E}} desc(e, p).$$

Cette propriété vérifie que les entités désignées par les positions appartiennent au point de vue indiqué dans la relation.

De plus, on supposera que l'incertitude d'une distance entre deux positions est systématiquement inférieure à cette distance. Cette restriction implique seulement qu'on puisse différencier les deux positions ; elle comporte l'avantage important que la distance sera seule à considérer pour ordonner un ensemble de positions (Cf. §4.2.1).

S'il existe un facteur d'échelle fixe d'une unité d'un point de vue à une autre unité d'un autre point de vue, alors les valeurs des distances dans ces deux points de vue sont liées par la valeur de la fonction $scale$ en ces unités.

On utilisera le plus souvent une notation simplifiée pour exprimer l'appartenance d'un élément à $\mathcal{R}_{quanti}$:

$$r = [(P_i, P_j), dist, incert, p] \in \mathcal{R}_{quanti} \overset{def}{\iff} (P_i \leftrightarrow_p P_j = [dist, incert]).$$

On omettra le point de vue p quand celui-ci ne sera pas indispensable à la compréhension et qu'il n'y aura pas d'ambiguïté.

Chapitre 4

Les requêtes

Ce chapitre décrit comment répondre à des requêtes simples, dont la réponse est donnée explicitement par la description des entités dans le formalisme précédent, et précise les relations qualitatives utiles en cartographie, tant du point de vue de leur sémantique que des inférences qu'elles permettent de réaliser. Il s'achève par une description d'un algorithme simplifié de recherche d'ensembles ordonnés d'entités.

4.1 Quelques requêtes simples

Les fonctions qui ont été définies permettent de répondre aisément à un certain nombre de requêtes. Par exemple, la requête «Quels sont les constituants de l'entité e dans le point de vue p?» a pour réponse la valeur de $desc(e,p)$. La notation $e \in_p e'$ indiquera que e appartient à e' dans le point de vue p. La requête «L'entité e appartient-elle à l'entité e'?», qui sera noté $e \in e'$, a pour réponse la valeur du test «$e \in \cup_{p \in \mathcal{P}} desc(e',p)$». Étant donné la récursivité des imbrications, une requête plus compliquée est «L'entité e appartient-elle *récursivement* à l'entité e'?», noté $e \in_{rec} e'$. La réponse à cette question est fonction de la véracité de l'expression suivante :

$$\exists e_1, \ldots, e_n \in \mathcal{E}, p \in \mathcal{P} \text{ tels que } e_1 = e', e_n = e, \forall i \in [1, n-1], e_{i+1} \in_p e_i.$$

On s'impose ici de trouver un point de vue unique contenant tous les éléments de la chaîne liant e à e'.
De même, on notera $e \in p$ pour indiquer qu'une entité est définie dans un point de vue, i.e. $e \in \cup_{e' \in \mathcal{E}} desc(e',p)$.

Par contre, les requêtes sur les relations entre entités nécessitent le plus souvent des calculs puisqu'une grande partie de la connaissance est implicite. Par exemple, pour déterminer l'ordre d'un certain nombre d'entités, il va falloir parcourir de nombreuses relations, naviguer d'un point de vue à l'autre, etc., pour extraire une relation explicite. Dans le but de détailler certains mécanismes permettant de répondre à de telles requêtes, nous allons devoir préciser les relations exprimables. Pour cette raison, nous ne considérerons que les relations intéressantes en cartographie, qui sont énumérées à la figure 4.1. L'étude de telles requêtes complexes est repoussée dans un premier temps à la section 4.2.3 pour un algorithme simplifié de recherche d'entités localement ordonnées, et au chapitre 6, après l'examen d'algorithmes de satisfaction de contraintes temporelles, dont nous nous inspirerons.

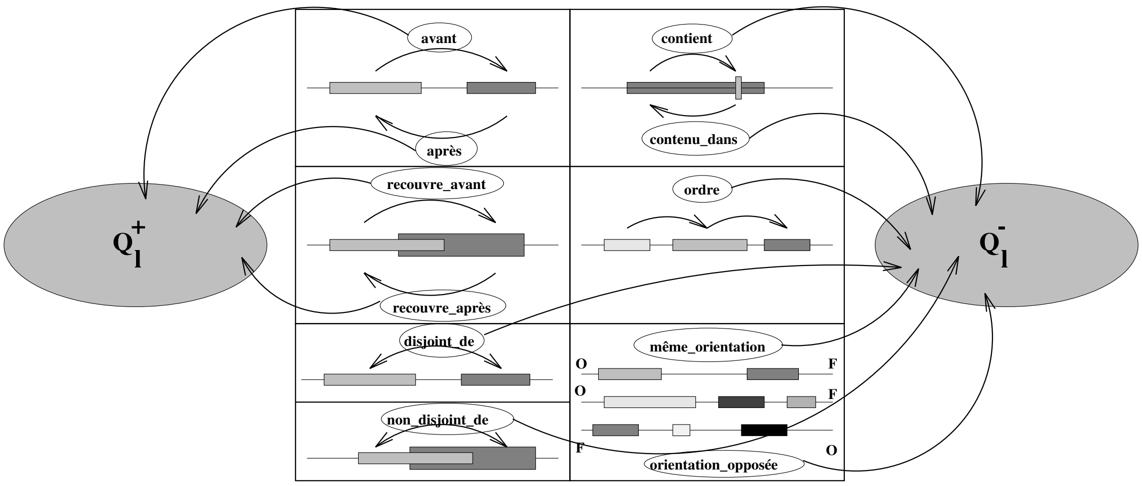


Figure 4.1 - : Ensemble des relations qualitatives entre entités biologiques. Celles-ci se subdivisent en deux sous-ensembles, $\mathcal{Q}_l^+$ pour les relations exprimant un ordre et nécessitant une référence et $\mathcal{Q}_l^-$ pour les autres. Remarquons que la relation *ordre*, n'ayant pas besoin de référence, appartient à $\mathcal{Q}_l^-$.

4.2 Les relations qualitatives exprimables en cartographie

Les ensembles de relations sont définis, dans le cadre des cartes génomiques, par les égalités suivantes :
$\mathcal{Q}_l^+ = \{$avant, après, recouvre_avant, recouvre_après$\}$.
$\mathcal{Q}_l^- = \{$contient, contenu_dans, disjoint_de, non_disjoint_de, ordre, même_orientation, orientation_opposée$\}$.

Les valeurs des fonctions *arit* et inv_{rel} pour les relations qualitatives sont données dans le tableau 4.1.

4.2.1 Sémantique et propriétés

La sémantique des relations est définie précisément par la donnée des positions relatives des entités mises en relation. Pour cela, on utilisera la fonction *dist* qui donne la distance réelle (même si elle est inconnue) entre deux positions.

Nous allons donc détailler chacune des relations qualitatives, et ajouter pour certaines quelques propriétés. On remarquera que les définitions qui vont suivre sont correctes aussi bien pour les intervalles que pour les points, pour peu que, lorsqu'une entité n'a pas de dimension, on confonde son origine et sa fin.

	arit	inv_{rel}
avant	{2}	après
après	{2}	avant
recouvre_avant	{2}	recouvre_après
recouvre_après	{2}	recouvre_avant
contient	{2}	contenu_dans
contenu_dans	{2}	contient
disjoint_de	{2}	disjoint_de
non_disjoint_de	{2}	disjoint_de
ordre	$[3, +\infty]$	
même_orientation	{2}	même_orientation
orientation_opposée	{2}	orientation_opposée

Tableau 4.1 - : Arités et inverses des relations qualitatives utilisées en cartographie génomique. Seule la relation *ordre* n'est pas binaire ; elle nécessite la donnée d'au moins trois entités.

Un choix important est fait dans l'expression de ces relations, qui est que la biologie moléculaire ne s'embarrasse pas d'inégalités strictes. En effet, on ne distingue pas une relation séparant strictement les extrémités des intervalles d'une obtenue continûment en les joignant. C'est pourquoi dans les descriptions qui vont suivre n'apparaîtra pas d'inégalité stricte.

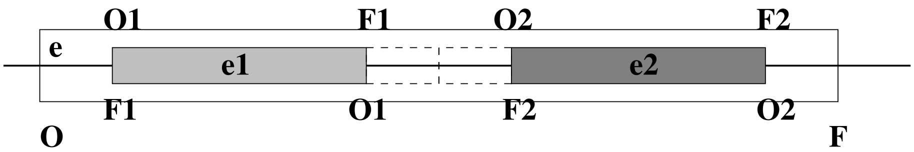
En particulier, les relations *avant* et *après* de $\mathcal{Q}_l^+$ sont des relations d'ordre et en vérifient donc les propriétés, à savoir :

$$\forall t \in \mathcal{Q}_l^+,$$
$$(e_1\, t_e\, e_1), \tag{4.1}$$
$$(e_1\, t_e\, e_2) \wedge (e_2\, t_e\, e_1) \Rightarrow e_1 = e_2, \tag{4.2}$$
$$(e_1\, t_e\, e_2) \wedge (e_2\, t_e\, e_3) \Rightarrow (e_1\, t_e\, e_3). \tag{4.3}$$

Il n'est pas possible d'utiliser des positions sur l'axe, car celui-ci n'est pas orienté ; il faut donc se servir des distances entre les extrémités des intervalles, ce qui oblige à une gymnastique d'autant plus compliquée qu'il faut traiter les deux «brins» de l'ADN, et, pour les relations qualitatives orientées, utiliser l'entité englobante. L'intérêt d'utiliser cette fonction *dist* est qu'elle nous permettra de développer des algorithmes passant d'une représentation qualitative à une représentation quantitative. Elle est équivalente à une relation quantitative avec une incertitude nulle.

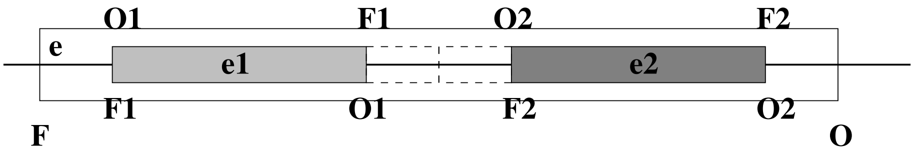
avant :

$$(e_1 \text{ avant}_e\, e_2) \Longleftrightarrow \max(dist(O,O_1), dist(O,F_1)) \leq \min(dist(O,O_2), dist(O,F_2)).$$



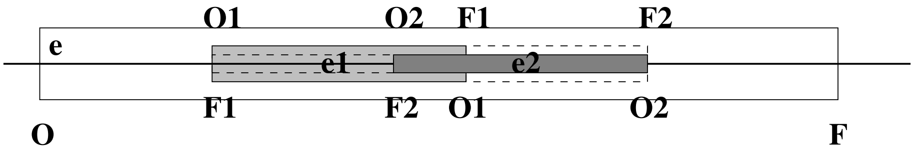
après :

$$(e_1 \text{ après}_e\ e_2) \Longleftrightarrow \min(dist(O,O_1), dist(O,F_1)) \geq \max(dist(O,O_2), dist(O,F_2)).$$



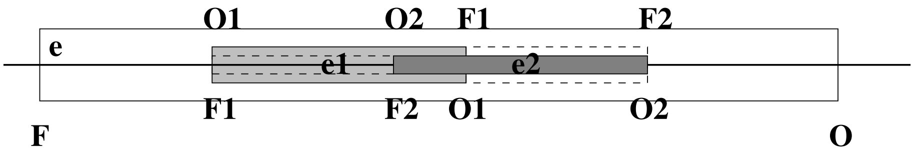
recouvre_avant :

$$\begin{aligned}&(e_1 \text{ recouvre\_avant}_e\ e_2) \Longleftrightarrow\\ &\quad \min(dist(O,O_1), dist(O,F_1)) \leq \min(dist(O,O_2), dist(O,F_2))\\ &\quad \leq \max(dist(O,O_1), dist(O,F_1)) \leq \max(dist(O,O_2), dist(O,F_2)).\end{aligned}$$



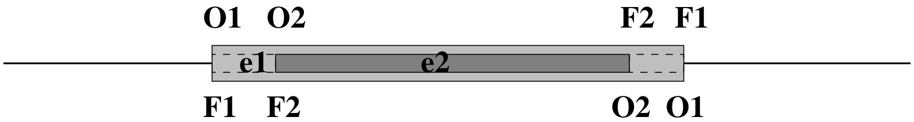
recouvre_après :

$$\begin{aligned}&(e_1 \text{ recouvre\_après}_e\ e_2) \Longleftrightarrow\\ &\quad \max(dist(O,O_1), dist(O,F_1)) \geq \max(dist(O,O_2), dist(O,F_2))\\ &\quad \geq \min(dist(O,O_1), dist(O,F_1)) \geq \min(dist(O,O_2), dist(O,F_2)).\end{aligned}$$



contient :

$$\begin{aligned}&(e_1 \text{ contient } e_2) \Longleftrightarrow\\ &\quad dist(O_1,F_1) \geq dist(O_1,O_2)\\ &\quad \wedge dist(O_1,F_1) \geq dist(O_1,F_2)\\ &\quad \wedge dist(O_1,F_1) \geq dist(F_1,O_2)\\ &\quad \wedge dist(O_1,F_1) \geq dist(F_1,F_2).\end{aligned}$$



contenu_dans :

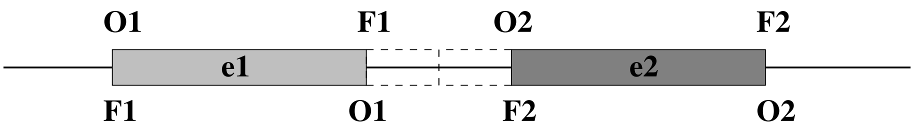
$$\begin{aligned}
&(e_1 \text{ contenu_dans } e_2) \Longleftrightarrow \\
&\quad dist(O_2, F_2) \geq dist(O_2, O_1) \\
&\quad \wedge dist(O_2, F_2) \geq dist(O_2, F_1) \\
&\quad \wedge dist(O_2, F_2) \geq dist(F_2, O_1) \\
&\quad \wedge dist(O_2, F_2) \geq dist(F_2, F_1).
\end{aligned}$$

De plus, toute entité qui appartient récursivement à une autre entité est contenue dans cette dernière entité.

$$\forall e, e' \in \mathcal{E}, e \in_{rec} e' \Rightarrow (e \text{ contenu\_dans } e') \wedge (e' \text{ contient } e) \tag{4.4}$$

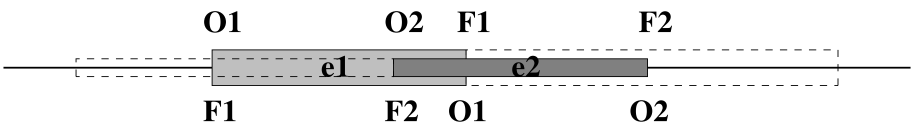
disjoint_de : Pour cela, il faut éviter qu'une extrémité d'une entité se trouve à l'intérieur de l'autre, c'est-à-dire qu'au moins une des distances séparant les extrémités des entités soit plus grande que la longueur d'une entité.

$$\begin{aligned}
&(e_1 \text{ disjoint_de } e_2) \Longleftrightarrow \\
&\quad \min(dist(O_1, O_2), dist(O_1, F_2)) \geq dist(O_1, F_1) \\
&\quad \vee \min(dist(F_1, O_2), dist(F_1, F_2)) \geq dist(O_1, F_1).
\end{aligned}$$



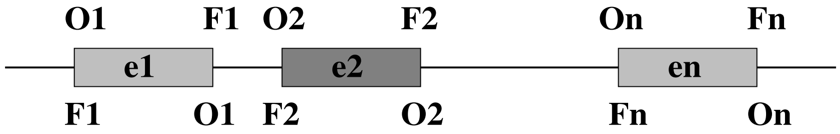
non_disjoint_de : Contrairement à la relation précédente, il faut désormais que les deux distances minimales soient inférieures à la longueur d'une entité. Remarquons que ces deux relations, *disjoint_de* et *non_disjoint_de*, ne sont pas disjointes puisque les égalités sont autorisées dans les deux cas.

$$\begin{aligned}
&(e_1 \text{ non_disjoint_de } e_2) \Longleftrightarrow \\
&\quad \min(dist(O_1, O_2), dist(O_1, F_2)) \leq dist(O_1, F_1) \\
&\quad \wedge \min(dist(F_1, O_2), dist(F_1, F_2)) \leq dist(O_1, F_1).
\end{aligned}$$



ordre : Pour la définition de la relation *ordre*, introduisons la fonction d_{max} entre deux entités, qui est la plus grande distance existant entre les quatre extrémités.

$$\begin{aligned}
&\text{ordre } (e_1, \ldots, e_n) \Longleftrightarrow \\
&\quad \forall i, j \in [1, n], i \neq j, (e_i \text{ disjoint_de } e_j) \\
&\quad \wedge \forall i \in [1, n-2], d_{max}(e_i, e_{i+1}) \leq d_{max}(e_i, e_{i+2}) \\
&\quad \wedge \forall i \in [2, n-1], d_{max}(e_i, e_{i+1}) \leq d_{max}(e_{i-1}, e_{i+1})
\end{aligned}$$



Une propriété de la relation *ordre* permet à partir de deux relations ayant en commun deux entités, d'en engendrer d'autres, comme l'indiquent la formule suivante et la figure 4.2 ; cette propriété sera appelée *additivité*.

$$\begin{aligned}
&\forall e_1, \ldots, e_n, e'_1, \ldots, e'_m \in \mathcal{E}, i, j, k, l \in I\!N, i < j, k < l, \\
&\quad ordre(e_1, \ldots, e_n) \wedge ordre(e'_1, \ldots, e'_m) \wedge e_i = e'_k \wedge e_j = e'_l \\
&\quad \Rightarrow ordre(e_1, \ldots, e_i, e'_{k+1}, \ldots, e'_m) \wedge ordre(e'_1, \ldots, e'_k, e_{j+1}, \ldots, e_n) \\
&\quad \wedge ordre(e_1, \ldots, e_j, e'_{l+1}, \ldots, e'_m) \wedge ordre(e'_1, \ldots, e'_l, e_{k+1}, \ldots, e_n).
\end{aligned} \tag{4.5}$$

Il ne faut néanmoins pas oublier que toutes les entités reliées par une relation *ordre* doivent appartenir au même point de vue.

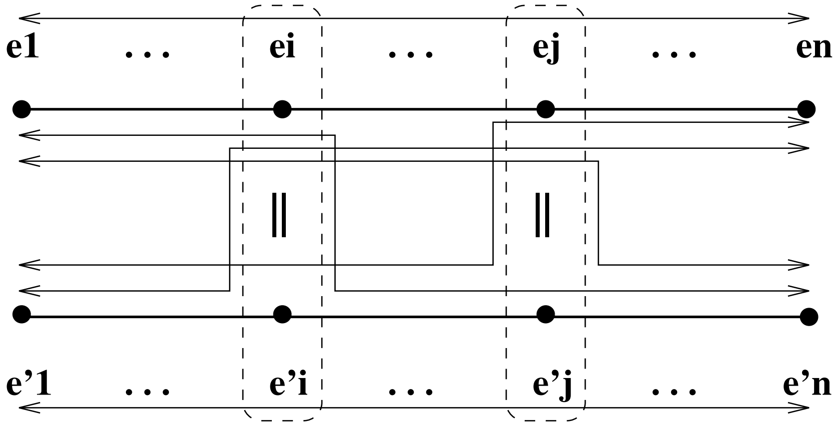


Figure 4.2 - : Inférence d'ordres. Deux relations *ordre* partageant deux entités impliquent quatre autres relations *ordre*, obtenues en suivant les flèches de la figure.

même_orientation, orientation_opposée : Ces deux relations sont intrinsèquement liées ; les contraintes sur les distances permettant de les distinguer sont plus compliquées à décrire. La figure 4.3 montre que, suivant les positions respectives de F_1 par rapport à O_2 et de F_2 par rapport à O_1, il faut rajouter des conditions adéquates pour préciser les positions de F_1 et F_2 par rapport à O_1 et O_2.

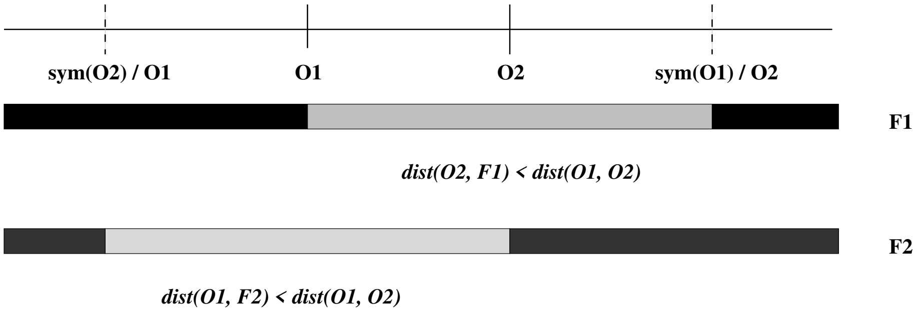


Figure 4.3 - : Comment positionner F_1 et F_2 par rapport à O_1 et O_2 ? Si, par exemple, F_1 est plus proche de O_2 que O_1 (partie claire en haut), on connaît immédiatement l'orientation de la première entité (car F_1 est toujours à droite de O_1) ; si alors, par contre, F_2 est plus loin de O_1 que O_2 (partie sombre en bas), il faut rajouter une condition sur la distance relative entre F_2 et O_2, et entre F_2 et O_1.

Les résultats sont résumés dans le tableau suivant (tableau 4.2).

$dist(O_1, F_2) - dist(O_1, O_2)$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$	$+$
$dist(O_2, F_1) - dist(O_1, O_2)$	$-$	$+$	$+$	$-$	$-$	$+$	$+$	$+$	$+$
$dist(O_1, F_1) - dist(O_2, F_1)$	?	$+$	$-$	?	?	$+$	$+$	$-$	$-$
$dist(O_2, F_2) - dist(O_1, F_2)$	?	?	?	$+$	$-$	$+$	$-$	$+$	$-$
orientation	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$+$	$-$

Tableau 4.2 - : Expression des relations d'orientation à l'aide de distances. Suivant le signe des différences entre distances de la première colonne ($-$, $+$, ou ? si le signe ne modifie pas le résultat), l'orientation est la même ($-$) ou est l'opposé ($+$).

La relation *même_orientation* est une relation d'équivalence ; elle possède en effet les propriétés de réflexivité, de symétrie et de transitivité :

$$(e \text{ même\_orientation } e), \tag{4.6}$$

$$(e \text{ même\_orientation } e') \Rightarrow (e' \text{ même\_orientation } e), \tag{4.7}$$

$$(e \text{ même\_orientation } e') \wedge (e' \text{ même\_orientation } e'') \Rightarrow (e \text{ même\_orientation } e''). \tag{4.8}$$

Des propriétés évidentes de la relation *orientation_opposée* sont :

$$(e \text{ orientation\_opposée } e') \Rightarrow (e' \text{ orientation\_opposée } e), \tag{4.9}$$

$$(e \text{ orientation\_opposée } e') \wedge (e' \text{ orientation\_opposée } e'') \Rightarrow (e \text{ même\_orientation } e''). \tag{4.10}$$

Les propriétés précédentes ne sont valides que pour les entités pour lesquelles l'orientation a un sens, c'est-à-dire pour celles qui, dans au moins un point de vue, ont une dimension non nulle.

Enfin, les relations qualitatives orientées vérifient deux propriétés importantes en liaison avec les relations de *même_orientation* et d'*orientation_opposée* :

$$\forall t \in \mathcal{Q}_l^+, \forall e, e', e_1, e_2 \in \mathcal{E}, \text{ tels que } (e_1\, t_e\, e_2) \wedge \exists p \in \mathcal{P}\; e_1, e_2 \in_p e'$$

$$(e \text{ même\_orientation } e') \Longleftrightarrow (e_1\, t_{e'}\, e_2) \tag{4.11}$$

$$(e \text{ orientation\_opposée } e') \Longleftrightarrow (e_1\, t_{e'}^{-1}\, e_2) \tag{4.12}$$

La propriété précédente est valide pour tout élément de $\mathcal{Q}_l^+$ car les relations inférées appartiennent au même point de vue que celui de la relation initiale, ceci parce que, si e et e' sont en relation, ils appartiennent à (au moins) un même point de vue.

4.2.2 Quelques règles d'inférence

Les équivalences suivantes permettent, à partir d'un nombre restreint de règles d'inférence, d'engendrer toutes celles qui utilisent les inverses et des orientations différentes. Soient $t, t' \in \mathcal{Q}_l^+$ et $\bar{t} \in \mathcal{T}$,

$$\begin{array}{ll}
 & \left\{ \begin{array}{l} (e_1\, t_e\, e_2) \wedge (e_2\, t'_{e'}\, e_3) \wedge (e \text{ même_orientation } e') \\ \qquad \Rightarrow (e_1 \bar{t}_e e_3) \text{ ou } (e_1 \bar{t}_{e'} e_3) \text{ ou } \bar{t}(e_1, e_2, e_3). \end{array} \right. \\
\Longleftrightarrow & \left\{ \begin{array}{l} (e_1\, t_e\, e_2) \wedge (e_2\, t'^{-1}_{e'}\, e_3) \wedge (e \text{ orientation_opposée } e') \\ \qquad \Rightarrow (e_1 \bar{t}_e e_3) \text{ ou } (e_1 \bar{t}^{-1}_{e'} e_3) \text{ ou } \bar{t}(e_1, e_2, e_3). \end{array} \right. \\
\Longleftrightarrow & \left\{ \begin{array}{l} (e_1\, t_e^{-1}\, e_2) \wedge (e_2\, t'_{e'}\, e_3) \wedge (e \text{ orientation_opposée } e') \\ \qquad \Rightarrow (e_1 \bar{t}^{-1}_e e_3) \text{ ou } (e_1 \bar{t}_{e'} e_3) \text{ ou } \bar{t}(e_1, e_2, e_3). \end{array} \right. \\
\Longleftrightarrow & \left\{ \begin{array}{l} (e_1\, t_e^{-1}\, e_2) \wedge (e_2\, t'^{-1}_{e'}\, e_3) \wedge (e \text{ même_orientation } e') \\ \qquad \Rightarrow (e_1 \bar{t}^{-1}_e e_3) \text{ ou } (e_1 \bar{t}^{-1}_{e'} e_3) \text{ ou } \bar{t}(e_1, e_2, e_3). \end{array} \right.
\end{array}$$

Preuve : Supposons que la première proposition soit vérifiée et que le premier membre de la seconde le soit aussi et montrons que le conséquent est vrai. Pour cela, considérons e'', élément de $\mathcal{E}$ identique à e' à la différence que $O(e'') = F(e')$ et $F(e'') = O(e')$. Alors, e'' vérifie : (e' orientation_opposée e''), donc (e même_orientation e''). De plus, $(e_2\, t'^{-1}_{e'}\, e_3)$ implique $(e_2\, t'_{e''}\, e_3)$ en vertu de l'équation 4.12. On peut donc appliquer la première proposition et déduire $(e_1 \bar{t}_e e_3)$ ou $(e_1 \bar{t}_{e''} e_3) \Leftrightarrow (e_1 \bar{t}^{-1}_{e'} e_3)$ en appliquant à nouveau l'équation 4.12, ou $[\bar{t}, E] \in \mathcal{R}^-_{quali}$. □

Quelles sont alors les instanciations de t_e, $t'_{e'}$ et $\bar{t}$ qui vérifient une des quatre propositions équivalentes ? Si le but est d'obtenir des ordres locaux, il est intéressant de remplacer dans la première implication t_e et t'_e par la relation *avant*, et $\bar{t}$ par la relation *ordre*(e_1, e_2, e_3), ce qui donne les quatre inférences de la figure 4.4.

Si, en particulier, on pose $e = e'$, alors toute instanciation de la première proposition va engendrer une proposition équivalente avec les inverses des relations employées ; on obtient alors une table de transitivité incomplète. Les instanciations possibles sont données dans le tableau 4.3, dans lequel ont été ajoutées les inférences avec les relations *contenu_dans* et *contient*.

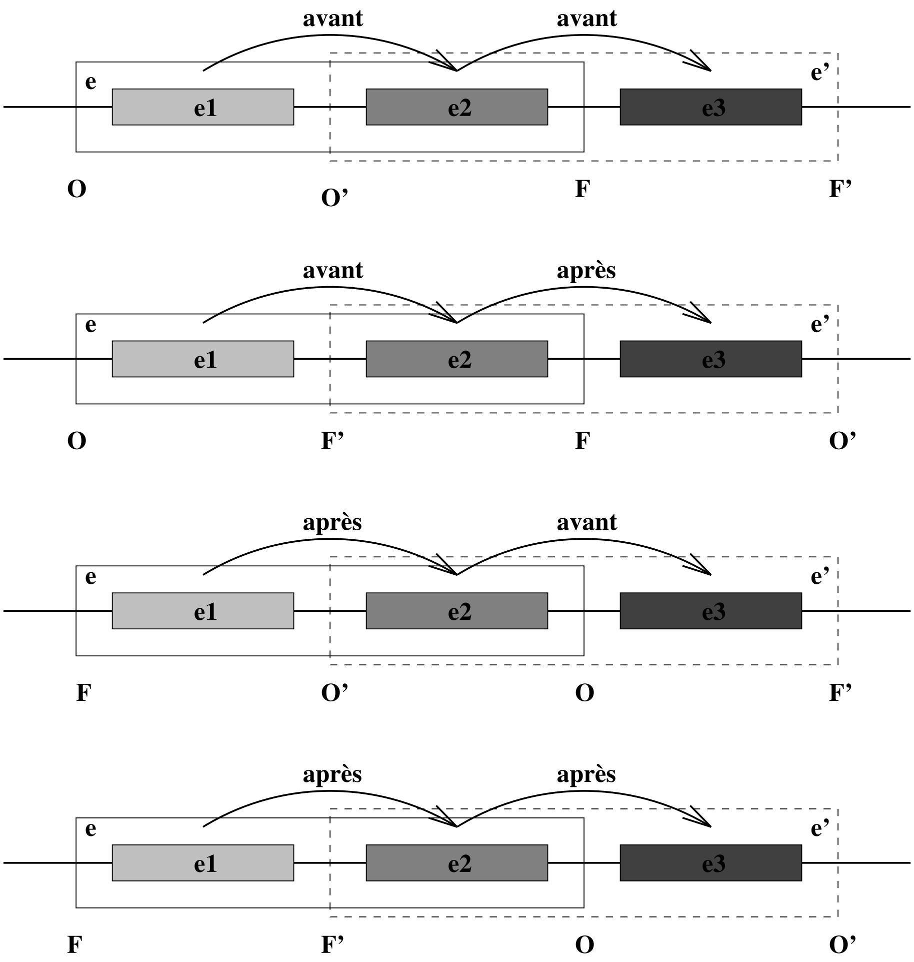


Figure 4.4 - : Quatre inférences possibles permettant de déterminer des ordres locaux. Elles impliquent toutes la relation d'ordre ordre(e_1, e_2, e_3).

Il existe également un certain nombre d'inférences liées aux relations quantitatives, éventuellement impliquant des relations qualitatives. Elles sont détaillées ici (rappelons que par définition $d_{ij} = dist(P_i, P_j)$ et $i_{ij} = incert(P_i, P_j)$ dans l'expression $P_i \leftrightarrow P_j = (d_{ij}, i_{ij})$).

Le but des règles qui suivent est de pouvoir ordonner les positions de façon à ce que l'additivité des distances fonctionne ; en particulier, ceci permettra d'appliquer des algorithmes efficaces de raisonnement temporel quantitatif (Cf. §7.3.3). Définissons alors la fonction ord sur les positions, telle que $ord(P_1, \ldots, P_n)$ signifie que l'ordre des positions est $P_1, \ldots, P_n$.

- $\forall P_1, P_2, P_3, P_1 \leftrightarrow P_2 = (d_{12}, i_{12}), P_1 \leftrightarrow P_3 = (d_{13}, i_{13}), P_2 \leftrightarrow P_3 = (d_{23}, i_{23}), d_{13} \geq d_{12} + d_{23}$, on a : $P_1 \leftrightarrow P_3 = (d_{12} + d_{23}, i_{12} + i_{23}) = (d^c_{13}, i^c_{13})$, d'où le remplacement des deux relations impliquant P_1 et P_3 par $P_1 \leftrightarrow P_3 = (max(d_{13} - i_{13}, d^c_{13} - i^c_{13}) + min(d_{13} + i_{13}, d^c_{13} + i^c_{13}))/2, (min(d_{13} + i_{13}, d^c_{13} + i^c_{13}) - max(d_{13} - i_{13}, d^c_{13} - i^c_{13}))/2)$ (figure 4.5).

$e_1\ t\ e_2$	$e_2\ t'\ e_3$	$e_1\ \bar{t}\ e_3$
avant après	avant après	avant après
avant après	recouvre_avant recouvre_après	avant après
recouvre_avant recouvre_après	avant après	avant après
avant	contient	avant
après	contient	après
contenu_dans	avant	avant
contenu_dans	après	après

Tableau 4.3 - : Résultat de l'instanciation des inférences précédentes.

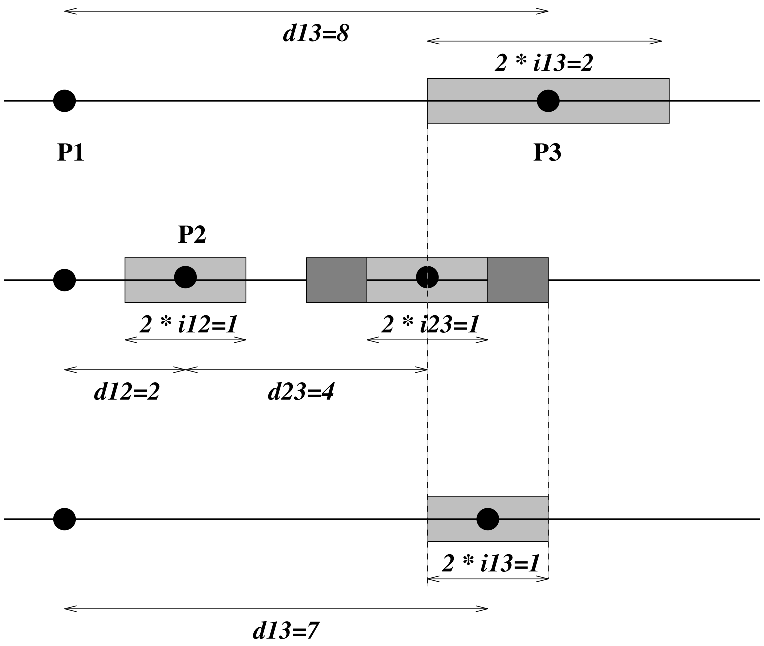


Figure 4.5 - : Règle d'inférence calculant l'intersection des domaines d'une position quand plusieurs distances sont disponibles. La distance d_{13} est calculée en utilisant l'information directe et celle calculée grâce aux distances d_{12} et d_{23}.

Il apparaît une incohérence si l'intersection des deux intervalles est vide c'est-à-dire si $[d_{13} - i_{13}, d_{13} + i_{13}] \cap [d^c_{13} - i^c_{13}, d^c_{13} + i^c_{13}] = \emptyset$.

- La présence d'une relation qualitative et d'une relation d'orientation permet d'ordonner les extrémités des entités en relation, et donc d'appliquer des propriétés d'additivité sur les distances ; les inférences de ces ordres sont résumés dans le tableau 4.4.

  Ces inférences sont valides même quand une entité devient un point (si tant est que

$(e_1\ r_e\ e_2)$	$(e_1$ même_orientation $e_2)$	$(e_1$ orientation_opposée $e_2)$
avant	$ord(O, O_1, F_1, O_2, F_2, F)$ $ord(O, F_1, O_1, F_2, O_2, F)$	$ord(O, O_1, F_1, F_2, O_2, F)$ $ord(O, F_1, O_1, O_2, F_2, F)$
recouvre_avant	$ord(O, O_1, O_2, F_1, F_2, F)$ $ord(O, F_1, F_2, O_1, O_2, F)$	$ord(O, O_1, F_2, F_1, O_2, F)$ $ord(O, F_1, O_2, O_1, F_2, F)$
contient	$ord(O_1, O_2, F_2, F_1)$	$ord(O_1, F_2, O_2, F_1)$

Tableau 4.4 - : Ordonner les extrémités des entités à partir de relations qualitatives. Pour trois relations (*avant*, *recouvre_avant* et *contient*), suivant l'orientation relative de e_1 et e_2, le tableau donne l'ordre des extrémités en fonction encore de l'orientation relative de e par rapport à e_1 ; le premier élément de la colonne est valide quand e et e_1 ont la même orientation.

la relation a toujours un sens). Pour les relations inverses, il suffit de se ramener à un cas traité en utilisant la définition de l'inverse.

- $\forall e_1, \ldots, e_n \in \mathcal{E}, ordre(e_1, \ldots, e_n) \Rightarrow ord(O_1, \ldots, O_n) \wedge ord(F_1, \ldots, F_n)$.

- $\forall P_1, P_2, P_3, d_{12} + i_{12} \leq d_{13} - i_{13} \Rightarrow ord(P_1, P_2, P_3)$.

4.2.3 Un algorithme de génération d'ordres locaux

Le nombre des inférences possibles à partir des relations de $\mathcal{Q}_l$ est trop important pour les écrire ici, voire les traiter toutes. Nous allons néanmoins présenter un algorithme dont le but est de trouver les ordres locaux maximaux, i.e., à partir des relations qualitatives exprimées, exhiber les ensembles de $\mathcal{E}$ liés par une relation d'ordre tels que quels que soient deux de ces ensembles, il n'existe pas de relation d'ordre contenant leur union (c'est en ceci que ces ordres sont appelés maximaux).

L'algorithme présenté est volontairement simple et n'inclut pas d'information quantitative ; d'autres algorithmes plus complexes seront développés grâce à l'utilisation de techniques de raisonnement temporel (Cf. chapitre 7).

Il utilise les inférences précédemment mises en évidence de façon à découvrir les ordres locaux qui existent entre les entités décrites à partir des relations exprimées initialement. Implicitement, cet algorithme se sert de résultats dans les différents points de vue, puisque toute nouvelle relation, fruit d'inférences antérieures, peut servir à n'importe quel point de vue, pourvu que les entités mises en relation appartiennent au moins à un point de vue commun.

Avant de parler de l'algorithme lui-même, nous allons simplifier la représentation en ne gardant qu'un seul élément de chaque couple (relation, relation inverse), et en modifiant les règles d'inférence pour qu'elles n'engendrent que des représentants de la relation choisie. Les règles d'inférence utilisées sont récapitulées ici :

1. Génération de la relation *contient* à partir des relations d'appartenance (Cf. équation 4.4) ;

2. Propriétés des relations *même_orientation* et *orientation_opposée* (Cf. équations 4.6, 4.7, 4.8, 4.9 et 4.10) ;
3. Génération des relations *même_orientation* et *orientation_opposée* (Cf. équations 4.11 et 4.12) ;
4. Transitivité des relations (Cf. équation 4.3 et tableau 4.3) ;
5. Génération de relations à partir de propriétés de *même_orientation* et *orientation_opposée* (Cf. équations 4.11 et 4.12) ;
6. Génération de relations *ordre* à partir de relations (Cf. figure 4.4) ;
7. Additivité des relations *ordre* (Cf. équation 4.5 et figure 4.2).

L'algorithme fonctionne de la manière suivante :

1. Engendrer toutes les relations *même_orientation* et *orientation_opposée* (propriété 2) ;
2. Engendrer toutes les relations *contient* (propriété 1) ;
3. Utiliser la table de transitivité pour déterminer toutes les nouvelles relations (propriété 4) ;
4. Déterminer les relations de *même_orientation* et *orientation_opposée* (propriété 3) ;
5. Pour toutes les nouvelles relations découvertes, répéter la séquence des deux actions précédentes jusqu'à stabilité ;
6. Engendrer toutes les relations *ordre* (propriété 6) ;
7. Appliquer l'additivité sur les ordres découverts (propriété 7).

Cet algorithme est correct (car il n'utilise que des inférences établies) et complet (toutes les relations *ordre* sont découvertes, en particulier grâce à l'étape de répétition jusqu'à stabilité) ; malheureusement, le problème est intrinsèquement NP-complet puisqu'il inclut la résolution d'un problème général de satisfaction de contraintes temporelles.

L'étude du raisonnement temporel (Cf. chapitre 6) nous permettra de développer des algorithmes non limités à la recherche d'ordres et fondés sur un formalisme plus propre (Cf. chapitre 7).

Chapitre 5

Instanciation du formalisme aux cartes génomiques

Ce chapitre permet de boucler sur la biologie moléculaire et montre, s'il en était besoin, que la formalisation des cartes peut être réalisée dans ce formalisme ; il montre surtout, à partir d'un exemple, comment cette instanciation se fait.

5.1 Partie descriptive

Les formules et tableaux qui suivent montre une correspondance possible (et non exhaustive) entre les entités définies dans la section précédente et la réalité biologique. Définissons d'abord les ensembles de la typologie, $\mathcal{P}$ l'ensemble des points de vue, $\mathcal{T}$ l'ensemble des types et $\mathcal{U}$ l'ensemble des unités :
$\mathcal{P}$ = {génétique, physique, cytogénétique}
$\mathcal{T}$ = {gène, marqueur, séquence, site de restriction, bande cytogénétique, intron, exon, ...}
$\mathcal{U}$ = {centiMorgan, kilobase, % de longueur du chromosome}

La fonction *const* se décrit par un tableau à double entrée : elle indique les types constitutifs associés à tout type et à tout point de vue.

const	génétique	physique	cytogénétique
carte	{gène, marqueur, carte}	{gène, séquence, site, carte}	{gène, bande, carte}
gène	$\emptyset$	{intron, exon}	$\emptyset$
marqueur	$\emptyset$	$\emptyset$	$\emptyset$
séquence	$\emptyset$	{...}	$\emptyset$
site	$\emptyset$	$\emptyset$	$\emptyset$
bande	$\emptyset$	$\emptyset$	{bande}
intron	$\emptyset$	$\emptyset$	$\emptyset$
exon	$\emptyset$	$\emptyset$	$\emptyset$

De même, la fonction *dim* est décrite dans le tableau suivant ; le symbole $\bullet$ indique une dimension ponctuelle du type considéré dans un point de vue donné, le signe $\leftrightarrow$ un intervalle. Quand aucun symbole n'est visible, le type n'a pas d'existence dans le point de vue.

dim	génétique	physique	cytogénétique
carte	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
gène	$\bullet$	$\leftrightarrow$	$\bullet$
marqueur	$\bullet$		
séquence		$\leftrightarrow$	
site		$\bullet$	
bande			$\leftrightarrow$
intron		$\leftrightarrow$	
exon		$\leftrightarrow$	

Enfin, les fonctions *unit* et *add* sont telles que :
unit(génétique) = centiMorgan ;
unit(physique) = kilobase ;
unit(cytogénétique) = % de longueur du chromosome ;
add(centiMorgan) = faux ;
add(kilobase) = vrai ;
add(% de longueur du chromosome) = vrai.

Les ensembles de relations ont été définis dans la section 4.1 sur les requêtes ; rappelons les pour mémoire :
$\mathcal{Q}_l^+$ = {avant, après, recouvre_avant, recouvre_après}.
$\mathcal{Q}_l^-$ = {contient, contenu_dans, disjoint_de, non_disjoint_de, ordre, même_orientation, orientation_opposée}.

Considérons alors les cartes de la figure 5.1 ; le chromosome se décompose dans les différents points de vue. La partie entourée d'un cercle correspond à la sous-carte *carte1*. Celle-ci se décompose également en diverses entités. La carte cytogénétique comporte des éléments qui lui sont propres, les bandes cytogénétiques, qui se décomposent récursivement en sous-bandes.

Le tableau suivant indique la décomposition effective des entités visibles sur la carte, en donnant les valeurs de la fonction *desc*. Les types des entités satisfont bien sûr les contraintes descriptives de la fonction *const*. Ainsi, par exemple, le gène D1S21 n'a pour constituants dans le point de vue physique que des entités des types intron et exon.

5.2 Les relations

Les cartes de la figure 5.1 contiennent des informations qualitatives et quantitatives. Les premières ne permettent pas d'ordonner intégralement les entités, il existe en effet

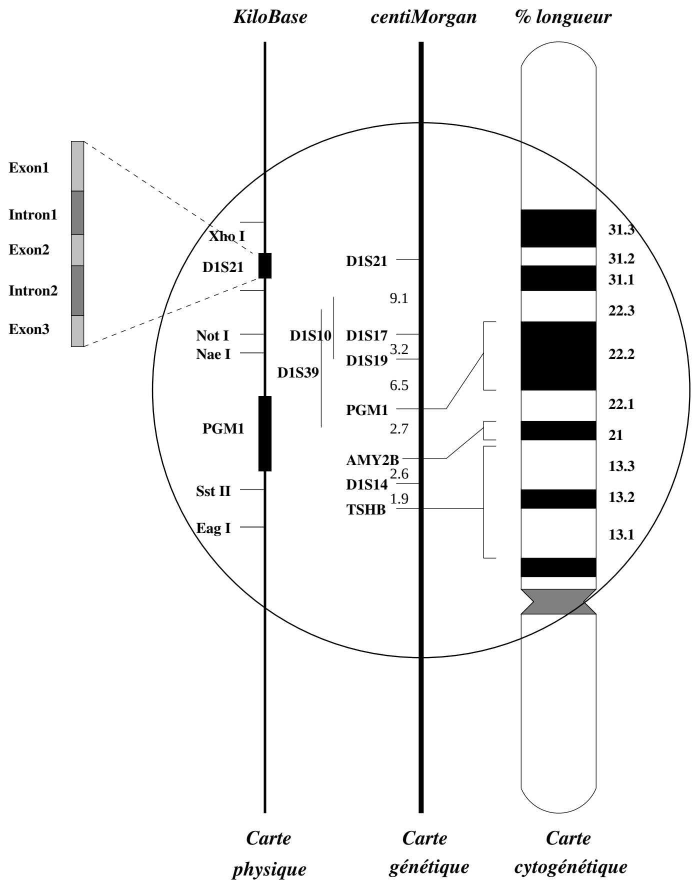


Figure 5.1 - : Intégration de différentes cartes génomiques. Comment représenter, à l'aide du formalisme proposé, leur organisation et les relations qui permettent de positionner les entités ?

des incertitudes sur les positions de D1S10 et D1S39. De même, la position de TSHB sur la carte cytogénétique n'a pour résolution qu'une bande (la bande 13) ; la connaissance disponible ne permet pas de décider à quelle sous-bande ce gène appartient. Les relations quantitatives visibles sont limitées à la carte génétique ; d'autres qui n'apparaissent pas explicitement sont les longueurs des bandes vis-à-vis de la longueur totale du chromosome.

desc	génétique	physique	cytogénétique
chromosome	{carte1, ...}	{carte1, ...}	{carte1, ..., 13, 21, 22, 31, ...}
carte1	{D1S10, D1S14, D1S17, D1S19, D1S21, D1S39, PGM1, AMY2B, TSHB}	{D1S21, PGM1, XhoI, NotI, NaeI, SstII, EagI}	{PGM1, AMY2B, TSHB}
D1S21		{intron1, intron2, exon1, exon2, exon3}	
D1S14	...	...	...
...		...	
13			{13.1, 13.2, 13.3}
22			{22.1, 22.2, 22.3}
...			...

Les relations qualitatives orientées, éléments de $\mathcal{R}^+_{quali}$, sont les triplets suivants :

t $\in \mathcal{Q}_l^+$	e_1, $e_2 \in \mathcal{E}$	$e \in \mathcal{E}$
avant	(TSHB, D1S14)	carte1
avant	(D1S14, AMY2B)	carte1
recouvre_avant	(D1S39, D1S10)	carte1
après	(D1S21, NotI)	carte1
après	(NotI, PGM1)	carte1
après	(PGM1, AMY2B)	carte1
avant	(EagI, SstII)	carte1
avant	(SstII, PGM1)	carte1
avant	(exon1, intron1)	D1S21
avant	(intron1, exon2)	D1S21
avant	(exon2, intron2)	D1S21
avant	(intron2, exon3)	D1S21
...		

Les éléments de $\mathcal{R}^-_{quali}$ apparaissent dans le tableau suivant :

t $\in \mathcal{Q}_l^-$	$e_1, e_2 \in \mathcal{E}$
contenu_dans	(22.1, 22)
contenu_dans	(22.2, 22)
contenu_dans	(22.3, 22)
contenu_dans	(TSHB, 13)
contenu_dans	(AMY2B, 21)
contenu_dans	(PGM1, 22.2)

Enfin, pour finir, les relations quantitatives se retrouvent dans le tableau suivant :

$p_1, p_2 \in Pos$	dist	incert	point de vue
(orig(D1S21), orig(D1S17))	9.1	0	génétique
(orig(D1S17), orig(D1S19))	3.2	0	génétique
(orig(D1S21), orig(D1S10))	8.5	3.1	génétique

5.3 Quelques requêtes

Un exemple de requête sur la composition de la carte peut être : «L'intron 2 appartient-il récursivement à la sous-carte 1 ?», ce qu'on peut écrire «intron2 $\in_{rec}$ carte1». Cette requête aura pour réponse *vrai* puisqu'il existe un chemin de composition partant de carte1 et menant à intron2, qui transite par D1S21.

Une requête plus complexe, qui nécessite d'utiliser un mécanisme d'inférence, est : «Quelle est la position de D1S21 par rapport a AMY2B ?». Pour répondre à cette interrogation, il faut introduire deux entités auxiliaires et utiliser la table de transitivité ; en effet D1S21 se trouve après NotI, qui lui-même est après PGM1, qui se situe après AMY2B. En appliquant deux fois la table de transitivité, on infère que D1S21 est après AMY2B. Remarquons que les inférences ont été réalisées sur les différents points de vue, et que la connaissance sur la carte génétique seule n'aurait pas suffi. Il a fallu introduire le site de restriction NotI pour faire le lien entre D1S21 et PGM1. C'est pour de telles inférences que l'intégration des différentes cartes est profitable. L'algorithme de génération d'ordres locaux (Cf. section 4.2.3) aurait trouvé cette relation puisqu'il applique la table de transitivité jusqu'à ce que plus aucune nouvelle relation ne soit inférée.

Par contre, il est des requêtes auxquelles l'algorithme ne pourra pas répondre ; ce sont toutes celles qui font intervenir des relations quantitatives. Par exemple, il est manifeste que D1S10 est après D1S19, car ce dernier est à une distance de $9.1 + 3.2 = 12.3$ de D1S21, tandis la distance séparant D1S21 et D1S10 est au plus de $8.5 + 3.1 = 11.6$. Pour réaliser cette inférence, il faut non seulement utiliser des informations quantitatives, mais également se servir de ces informations quantitatives pour déduire des informations qualitatives.

Les limitations de l'algorithme nous ont conduits à en définir un à même d'intégrer les connaissances provenant des relations qualitatives et quantitatives. Pour ce faire, nous nous sommes inspirés de techniques de raisonnement temporel, dont le formalisme se rapproche de celui des cartes génomiques. Après une présentation de ces techniques (chapitre 6), nous détaillerons le fonctionnement de l'algorithme global de construction de cartes (chapitre 7).

Chapitre 6

Représentation et raisonnement temporels

Nous avons déjà remarqué dans la section 2.3.1 la correspondance qui existe entre la représentation du temps et celle des cartes génomiques et qui provient du fait que toutes deux sont liées à un axe sur lequel se positionnent des entités ayant une longueur ou non (une durée dans le cas de la représentation du temps). Néanmoins, de profondes différences sont apparues concernant à la fois les relations mises en jeu et la possibilité de pouvoir «remonter dans le temps» quand on représente des cartes génomiques ; en effet, le premier point fait référence à la restriction des relations temporelles d'Allen tout en leur adjoignant des relations propres à la modélisation des cartes génomiques, en liaison avec le second point qui provient de la double orientation du brin d'ADN, alors que l'axe temporel n'en a qu'une, du passé vers l'avenir.

Malgré ces différences, essentiellement représentationnelles, de nombreux traitements se retrouvent d'une représentation à l'autre. En particulier, au niveau algorithmique, les travaux qui ont été faits sur le raisonnement temporel peuvent nous être très utiles, même s'ils nécessitent des adaptations pour prendre en compte les spécificités de la représentation de cartes génomiques.

Après une présentation des formalismes de représentation du temps, puis une étude des mécanismes de raisonnement sur ces formalismes, nous montrerons les similitudes et différences rencontrées du point de vue algorithmique, avant de préciser, dans le chapitre suivant, les algorithmes spécifiques à la modélisation de cartes génomiques.

6.1 Représenter le temps

La représentation du temps se heurte à de nombreuses difficultés, tant du point de vue du formalisme employé et de l'intégration de ces formalismes en un seul, que de celui du compromis à trouver entre expressivité et calculabilité. Parmi les différents formalismes, on trouve celui d'Allen qui permet de représenter l'ensemble des relations qualitatives entre intervalles, ainsi que des algèbres de points, auxquels il faut ajouter des sous-ensembles définis la plupart du temps pour des raisons de calculabilité (en particulier, les relations continues et les relations pointisables) ; à ces formalismes, il convient d'adjoindre toutes les représentations de relations quantitatives.

6.1.1 L'algèbre d'intervalles d'Allen

Allen [Allen83] a introduit un formalisme de représentation des intervalles suggérant l'utilisation de techniques de satisfaction de contraintes pour le raisonnement sur ces relations, qui a eu énormément d'impact et sur lequel de nombreux chercheurs dans ce domaine travaillent encore.

Un intervalle I est une paire de nombres réels (I^-, I^+) tels que $I^- < I^+$. Ces deux nombres sont interprétés comme des points sur l'axe temporel des réels. Une interprétation d'un intervalle est donc une correspondance de celui-ci vers l'axe temporel. L'ensemble des intervalles sera noté $\mathcal{I}$. L'énumération de toutes les relations binaires entre de telles paires permet d'en exhiber treize, détaillées dans le tableau 6.1. L'ensemble de ces treize relations sera nommé $\mathcal{R}_{Allen}$, et chacune peut être définie en précisant les relations arithmétiques sur les extrémités des intervalles. Aucune métrique n'est considérée sur cet ensemble. Les treize relations sont disjointes et forment une partition de la relation universelle. Une formule atomique $I_1 r I_2$ est satisfaite par une interprétation si et seulement si l'interprétation des intervalles I_1 et I_2 satisfait les relations sur les extrémités définies dans le tableau 6.1.

Relation atomique	Symbole	Exemple	Relations sur les extrémités
X avant Y	b (before)	xxxx	$X^- < Y^-, X^- < Y^+$
Y après X	$\breve{b}$	yyyy	$X^+ < Y^-, X^+ < Y^+$
X joint-avant Y	m (meets)	xxxxx	$X^- < Y^-, X^- < Y^+$
Y joint-après X	$\breve{m}$	yyyyy	$X^+ = Y^-, X^+ < Y^+$
X recouvre Y	o (overlaps)	xxxxx	$X^- < Y^-, X^- < Y^+$
Y recouvert-par X	$\breve{o}$	yyyyy	$X^+ > Y^-, X^+ < Y^+$
X pendant Y	d (during)	xxx	$X^- > Y^-, X^- < Y^+$
Y contient X	$\breve{d}$	yyyyyy	$X^+ > Y^-, X^+ < Y^+$
X commence Y	s (starts)	xxx	$X^- = Y^-, X^- < Y^+$
Y commencé-par X	$\breve{s}$	yyyyyyy	$X^+ > Y^-, X^+ < Y^+$
X finit Y	f (finishes)	xxx	$X^- > Y^-, X^- < Y^+$
Y fini-par X	$\breve{f}$	yyyyyy	$X^+ > Y^-, X^+ = Y^+$
X égale Y	$=$	xxxxx yyyyy	$X^- = Y^-, X^- < Y^+$ $X^+ > Y^-, X^+ = Y^+$

Tableau 6.1 - : Les treize relations d'Allen. Ces relations sont disjointes deux à deux, et expriment l'intégralité des positions relatives d'un intervalle par rapport à un autre. Une relation est satisfaite si les interprétations des intervalles qu'elle lie satisfont les relations sur leurs extrémités. La notation $\breve{}$ représente l'inverse de la relation et est définie un peu plus loin.

De manière à représenter des informations imprécises, on autorise l'expression d'unions (ou de disjonctions) sur les relations de $\mathcal{R}_{Allen}$. Ainsi, la formule $I_1(r_1, \ldots, r_n)I_2$ est satisfaite si et seulement si il existe un i tel que $I_1 r_i I_2$ est satisfaite. De même, plus généralement, un ensemble Θ de formules est satisfaisable si et seulement si il existe une interprétation satisfaisant chaque formule de Θ. Une telle interprétation est appelée un modèle de Θ. Enfin, une formule ϕ satisfaite par tout modèle d'un ensemble de formules Θ est dite logiquement induite par Θ, ce qui est noté $\Theta \models \phi$.

Le nombre total de relations considérées est alors de 2^{13} c'est-à-dire le nombre d'applications d'un ensemble de 13 éléments dans un ensemble de 2 (soit on prend la relation, soit on la laisse de côté dans la disjonction). Cet ensemble noté $\mathcal{A}$ contient la relation vide insatisfaisable $\perp$ et la relation universelle $\top = \bigcup_{\mathcal{R}_{Allen}} r$.

L'algèbre de relations d'Allen est construite sur l'ensemble $\mathcal{A}$ auquel on associe les opérations d'inversion (notée $\breve{}$), d'intersection (notée $\cap$) et de composition (notée o) définies de la manière suivante :

$$\begin{aligned}
&\forall I_1, I_2 \in \mathcal{I}, \forall r \in \mathcal{A},\\
&\quad I_1 \breve{r} I_2 \stackrel{def}{\Longleftrightarrow} I_2 r I_1;\\
&\quad I_1 (r \cap s) I_2 \stackrel{def}{\Longleftrightarrow} I_1\ r\ I_2 \wedge I_1\ s\ I_2;\\
&\quad I_1 (r\ o\ s) I_2 \stackrel{def}{\Longleftrightarrow} \exists I_3, I_1\ r\ I_3 \wedge I_3\ s\ I_2.
\end{aligned}$$

De ces définitions, il résulte que l'inverse d'une disjonction est égal à la disjonction des inverses, que l'intersection de deux relations est égale à l'intersection ensembliste des relations atomiques de chacune des deux relations (parce que les treize relations atomiques sont disjointes deux à deux) et que, pour finir, la composition de deux relations est égale à l'union de toutes les compositions d'un élément de la première relation avec un élément de la seconde. On se ramène ainsi à des opérations sur les relations atomiques de $\mathcal{R}_{Allen}$; la composition de deux relations de $\mathcal{R}_{Allen}$ est déterminée en regardant les relations sur les extrémités et Allen en donne la table détaillée pour les 13×13 compositions possibles. La composition de deux relations de $\mathcal{A}$ peut alors être calculée grâce à cette table et en utilisant la propriété de distributivité de la composition sur l'union [Ladkin90]. Il est clair que l'inconvénient d'un tel procédé est qu'il nécessite de recalculer la composition de relations qui pourraient être mises dans une table si celle-ci ne demandait pas autant de place ($2^{13} \times 2^{13}$) [Ladkin et al.92].

Les problèmes typiques de raisonnement qui se posent alors sont les suivants :

À partir d'un ensemble de formules Θ,

1. Déterminer s'il existe un modèle de Θ (problème de satisfaisabilité) ;

2. Déterminer pour chaque couple d'intervalles I_1, I_2 la relation la plus forte impliquée par Θ, c'est-à-dire l'ensemble R le plus petit tel que $\Theta \models (I_1 R I_2)$ (problème de fermeture déductive [Vilain et al.86] ou d'étiquetage minimum [vanBeek90b]) ;

3. Golumbic et Shamir [Golumbic et al.93] ajoutent à ces deux problèmes un autre plus général encore qui consiste à trouver *toutes* les solutions consistantes, puisque cet ensemble est différent de l'énumération du produit cartésien des solutions du problème précédent.

Étant donné que la résolution de ces deux problèmes est dans le cas général difficile, des sous-ensembles de $\mathcal{A}$ ont été définis pour les rendre solubles en temps polynomial ; ils sont détaillés au paragraphe suivant.

6.1.2 Les relations continues et les relations pointisables

Une alternative à la représentation du temps à l'aide d'intervalles est l'utilisation d'instants (ou points) de l'axe temporel [Vilain et al.86]. Comme précédemment, deux points sont reliés par un certain nombre de relations atomiques, qui sont *précède* $(<)$, *égale*$(=)$ et *suit* $(>)$. Il est également possible de définir l'union et la composition de ces relations atomiques de manière à exprimer des connaissances incertaines et spécifier des relations d'inférence. On obtient par l'union un ensemble de huit relations, dont la composition peut être exprimée, soit grâce aux propriétés de distributivité de la composition sur l'union, soit directement comme cela est fait dans le tableau 6.2 (la relation insatisfaisable $\perp$ a été omise).

o	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$	$\top$
$<$	$<$	$<$	$\top$	$\top$	$<$	$\top$	$\top$
$\leq$	$<$	$\leq$	$\top$	$\top$	$\leq$	$\top$	$\top$
$>$	$\top$	$\top$	$>$	$>$	$>$	$\top$	$\top$
$\geq$	$\top$	$\top$	$>$	$\geq$	$\geq$	$\top$	$\top$
$=$	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$	$\top$
$\neq$	$\top$	$\top$	$\top$	$\top$	$\neq$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

Tableau 6.2 - : La table de transitivité des relations entre points. Les unions sont notées de manière condensée ; ainsi $\leq$ est équivalent à l'union de $<$ et de $=$, la relation universelle $\top$ équivaut à $\{<,=,>\}$, etc.

Toute relation de $\mathcal{R}_{Allen}$ peut être exprimée comme un ensemble (conjonction) de relations sur les extrémités des intervalles. Par contre, ce n'est pas le cas des éléments de $\mathcal{A}$, c'est-à-dire des unions d'éléments de $\mathcal{R}_{Allen}$. En particulier, la disjonction d'intervalles, qui est égale à l'union de *avant* et *après*, ne peut pas se mettre sous la forme d'union de relations sur les extrémités des intervalles. Le sous-ensemble de $\mathcal{A}$, noté $\mathcal{P}$, qui a la propriété d'avoir une traduction dans l'algèbre de points précédemment définie, contient les relations *pointisables* et bénéficie de toutes les propriétés de cette algèbre en ce qui concerne la complexité (Cf. §6.2). Cet ensemble, si on inclut la relation insatisfaisable, contient 188 relations énumérées dans [vanBeek et al.90] et dans [vanBeek90a].

Si, à l'ensemble des huit relations entre points, on enlève l'inégalité (c'est-à-dire l'union de *précède* et *suit*), on aboutit par traduction au sous-ensemble de $\mathcal{A}$ appelé *relations continues*, noté $\mathcal{C}$, qui contient quatre-vingt trois relations [Nebel et al.93]. Le fait d'enlever l'inégalité rend l'ensemble obtenu convexe (au sens où, pour toute relation r de $\mathcal{C}$, l'ensemble des points satisfaisant r est convexe, c'est-à-dire, quels que soient deux points solutions, il existe un chemin de l'un à l'autre contenu dans l'ensemble des points solutions), et lui confère d'autres propriétés.

Par exemple, la relation $\{d, o, s\}$ peut être traduite dans l'algèbre de points sans l'inégalité, et appartient donc à l'ensemble des relations continues ; ce n'est pas le cas de

$\{d, o\}$ qui nécessite une relation avec une inégalité (figure 6.1).

$$X\{d,o,s\}Y \equiv \{(X^- < X^+),(Y^- < Y^+),\\ (Y^- < X^+),(X^+ < Y^+)\};\\ X\{d,o\}Y \equiv \{(X^- < X^+),(Y^- < Y^+),\\ (Y^- < X^+),(X^+ < Y^+),(X^- \neq Y^-)\}.$$

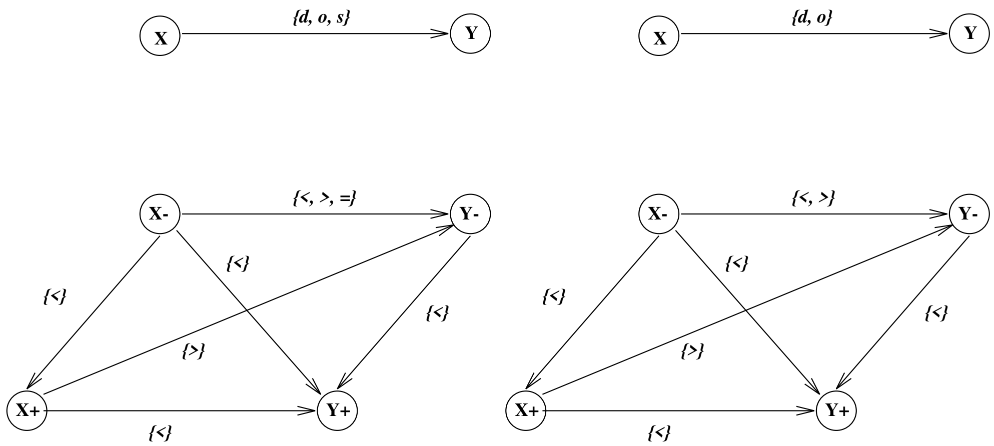


Figure 6.1 - : Traduction de l'algèbre d'intervalles vers les algèbres de points. La relation du premier réseau $\{d, o, s\}$ peut être traduite dans l'algèbre de points sans utilisation de la relation $\neq$ (qui est égale à $\{<, >\}$), celle de l'autre réseau $\{d, o\}$ ne le peut pas.

Nebel et Bürckert [Nebel et al.93] ont défini un ensemble de relations plus grand appelé la sous-classe *ORD-Horn* (noté $\mathcal{H}$), et qu'ils ont prouvé être le plus grand sous-ensemble de $\mathcal{A}$ permettant de résoudre de manière calculable les deux problèmes classiques en représentation du temps à savoir la satisfaisabilité et la détermination des relations maximales (Cf. §6.2). Cet ensemble est également construit à partir des relations de l'algèbre de points, à la différence que chaque élément d'une relation sur les extrémités doit inclure au plus un littéral positif (comme $=$ ou $\leq$) et un nombre arbitraire de littéraux négatifs de la forme $\neq$. Par exemple, la relation $\{o, s, \breve{f}\}$ appartient à cet ensemble car elle peut s'écrire dans l'algèbre de points de la façon suivante :

$$X\{o,s,\breve{f}\}Y \equiv \{(X^- < X^+),(Y^- < Y^+),\\ (X^- \leq Y^-),(X^- < Y^+),(Y^- < X^+),(X^+ \leq Y^+),(X^- \neq Y^- \wedge X^+ \neq Y^+)\}.$$

Dans le cas qui nous intéresse, il est important de noter que même pour l'ensemble composé des trois relations $\{\{\cap\}, \{b, \breve{b}\}, \{b, \cap, \breve{b}\}\}$ où $\cap$ est la relation *intersecte*, c'est-à-dire $\{m, \breve{m}, o, \breve{o}, s, \breve{s}, f, \breve{f}, d, \breve{d}, =\}$, qui représente donc intersecte, disjoint et la relation universelle, le problème de satisfaisabilité d'un réseau de contraintes est déjà NP-complet [Golumbic et al.93].

6.1.3 Relations quantitatives

Le formalisme suivant a été développé par Dechter, Meiri et Pearl [Dechter et al.91] et permet d'exprimer des inégalités simples entre points de l'axe temporel. Nous verrons plus loin des tentatives pour fusionner les représentations qualitative et quantitative du temps, dans le but de pouvoir tirer parti des inférences propres à chaque représentation.

Désormais, une variable X_i représente un point de l'axe temporel. Une contrainte est donnée à l'aide d'un ensemble d'intervalles $\{I_1, \ldots, I_n\} = \{[a_1, b_1], \ldots, [a_n, b_n]\}$, et peut être soit unaire, soit binaire. Dans le premier cas, la variable X_i sur laquelle porte la contrainte vérifie la disjonction suivante :

$$(a_1 \leq X_i \leq b_1) \vee \ldots \vee (a_n \leq X_i \leq b_n);$$

si la contrainte est binaire, les variables X_i et X_j vérifient la disjonction :

$$(a_1 \leq X_j - X_i \leq b_1) \vee \ldots \vee (a_n \leq X_j - X_i \leq b_n).$$

De la même manière que pour les relations temporelles qualitatives, il existe typiquement deux problèmes à résoudre : la satisfaisabilité du réseau de contraintes et la détermination de l'ensemble minimal des valeurs possibles pour chacune des variables (appelé son domaine minimal).

Si on introduit un point de référence X_0, appelé le «commencement du monde», les contraintes unaires peuvent être traitées comme des contraintes binaires par rapport à cette référence en remplaçant T_i par T_{0i}, ce qui est immédiat si on suppose, pour des raisons de simplicité, que $X_0 = 0$. La figure 6.2 montre un exemple de graphe de contraintes quantitatives sous ce formalisme.

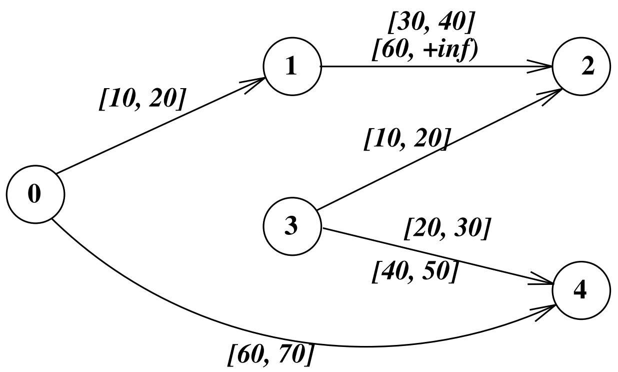


Figure 6.2 - : Exemple de graphe de contraintes. Ce graphe représente les contraintes suivantes : $30 \leq X_2 - X_1 \leq 40$ ou $X_2 - X_1 \geq 60, 10 \leq X_2 - X_3 \leq 20, 20 \leq X_4 - X_3 \leq 30$ ou $40 \leq X_4 - X_3 \leq 50, 10 \leq X_1 - X_0 \leq 20, 60 \leq X_4 - X_0 \leq 70$.

Il est également possible de définir les mêmes opérations que celles qui l'ont été sur les relations qualitatives, à savoir l'union, l'intersection et la composition de ces contraintes. Soient alors $T = \{I_1, \ldots, I_l\}$ et $S = \{J_1, \ldots, J_m\}$, deux contraintes portant sur une variable t, les opérations sont définies ainsi :

$T \cup S = \{I_1, \ldots, I_l, J_1, \ldots, J_m\}$.
$T \cap S = \{K_1, \ldots, K_n\}$ où $\forall i, j, K_k = I_i \cap J_j$.
$T \; o \; S = \{K_1, \ldots, K_n\}$ où $\forall i, j, K_k = [a + y, b + z]$ avec $I_i = [a, b], J_j = [y, z]$.

La figure 6.3 montre un exemple d'intersection et de composition.

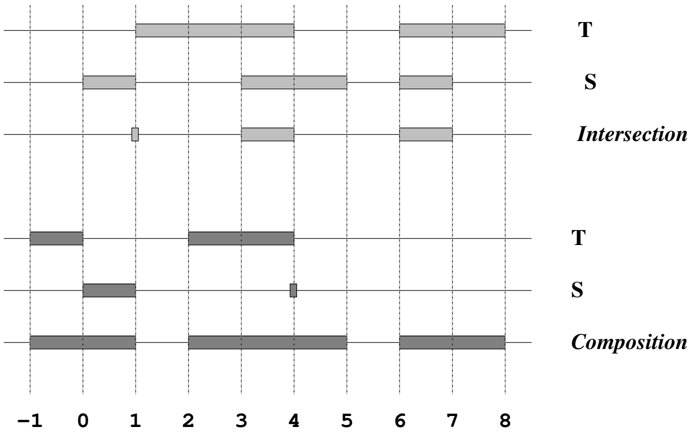


Figure 6.3 - : Intersection et composition de contraintes (d'après [Dechter et al.91]). Chaque contrainte est une union d'intervalles ; l'intersection des contraintes T et S est l'intersection ensembliste des unions d'intervalles ; la composition de T et S est l'union des couples d'intervalles pris dans T et S.

Il existe une relation d'ordre partiel naturelle entre deux contraintes T et S ; T est plus contrainte que S (noté $T \subseteq S$) si chaque paire de valeurs autorisée par T l'est également par S, ce qui implique que pour tout intervalle $I \in T$, il existe $J \in S$ tel que $I \subseteq J$. La contrainte la plus forte est la contrainte vide $\emptyset$, la plus lâche est la contrainte universelle $(-\infty, +\infty)$.

Cet ordre partiel peut être étendu aux réseaux de contraintes binaires ayant le même ensemble de variables de la façon suivante : un réseau $T = \{T_{ij}\}$ de contraintes est plus contraint qu'un réseau $S = \{S_{ij}\}$ si pour tout i et j, $T_{ij} \subseteq S_{ij}$. Alors, parmi tous les réseaux équivalents, c'est-à-dire qui ont le même ensemble de solutions, il en existe un unique minimal vis-à-vis de la relation $\subseteq$.

Un cas particulier de ces problèmes de satisfaction de contraintes temporelles quantitatives est celui où les contraintes ne spécifient qu'un seul intervalle $[a_{ij}, b_{ij}]$ (et non plus une union). Chaque arête i, j est donc étiquetée par la double inégalité $a_{ij} \leq X_j - X_i \leq b_{ij}$. Le problème se représente alors plutôt par un graphe de distances dans lequel l'arête (i, j) a pour valeur a_{ij} représentant la contrainte $X_j - X_i \leq a_{ij}$. La figure 6.4 montre le graphe de distances associé aux contraintes de l'exemple précédent en enlevant les unions d'intervalles.

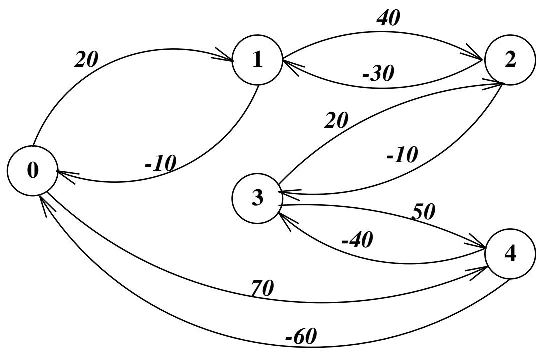


Figure 6.4 - : Exemple de graphe de distances. Ce graphe représente les contraintes suivantes : $30 \leq X_2 - X_1 \leq 40, 10 \leq X_2 - X_3 \leq 20, 40 \leq X_4 - X_3 \leq 50, 10 \leq X_1 - X_0 \leq 20, 60 \leq X_4 - X_0 \leq 70$.

Nous verrons dans le paragraphe 6.2.2 les mécanismes de raisonnement permettant de résoudre le réseau de contraintes et, plus généralement, de répondre à des requêtes sur les variables ou les couples de variables.

6.2 Raisonner sur le temps

6.2.1 Algèbres d'intervalles

Il est possible de représenter un ensemble de formules Θ de $\mathcal{A}$ par un réseau de contraintes binaires, dans lequel ne sont pas affichées les relations universelles entre deux intervalles ; les nœuds du réseau sont les intervalles et les arcs représentent les contraintes entre deux de ces intervalles (figure 6.5).

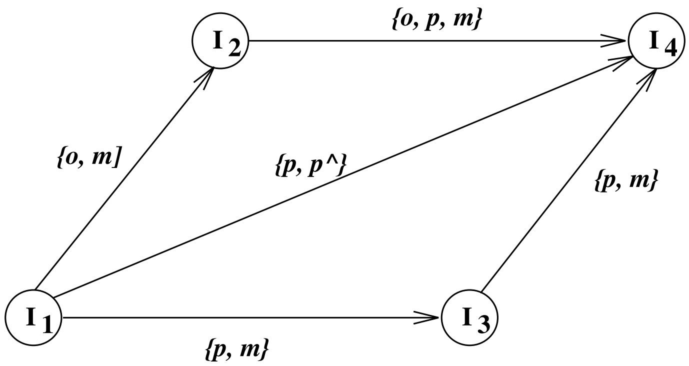


Figure 6.5 - : Réseau de contraintes temporelles. Chaque nœud du réseau est un intervalle, chaque arc entre deux nœuds symbolise une relation temporelle de $\mathcal{A}$; ainsi, l'intervalle I_1 soit rencontre, soit recouvre l'intervalle I_2, soit précède, soit rencontre I_3, soit précède, soit est précédé par I_4. Le réseau de contraintes présenté ici est dit *régulier* [Ladkin90] car il n'existe qu'un arc entre deux variables (il suffit pour obtenir un graphe régulier de remplacer deux arcs orientés différemment par l'intersection de l'un avec l'inverse de l'autre).

Les deux problèmes mentionnés avant (satisfaisabilité et détermination du réseau minimal) sont NP-complets pour $\mathcal{A}$. Allen [Allen83] a décrit un algorithme de consistance de chemin[1] qui, dans le cas présent de réseaux de contraintes binaires, est équivalent à la 3-consistance [Montanari74] ; un réseau est chemin-consistant si, pour tout chemin du réseau, il existe une instanciation des variables qui satisfait les contraintes sur le chemin ; la 3-consistance se limite à un chemin de longueur égale à 3. Il est intéressant de noter que la formalisation des réseaux de contraintes temporelles (ou TCSP pour *Temporal Constraint Satisfaction Problem*) est différente de celle des problèmes de satisfaction de contraintes (ou CSP pour *Constraint Satisfaction Problem*) ; en effet, il existe *a priori* une infinité d'instanciations possibles pour les intervalles, mais, intuitivement, un nombre fini de classes d'instanciation équivalentes. En réduisant le domaine de la relation liant deux variables, on limite implicitement le domaine (infini) des valeurs des extrémités des intervalles [vanBeek et al.90]. On peut également transcrire un TCSP en CSP en posant que les variables ne sont plus les intervalles, mais les relations entre eux, et que le problème

[1] La consistance d'arc est toujours satisfaite dans le cas des réseaux de contraintes temporelles [Ladkin90, Ladkin et al.94, vanBeek et al.90].

à résoudre est de réduire les domaines de ces variables en éliminant progressivement des relations de la disjonction en se basant sur les propriétés de transitivité. Pour plus de renseignements sur les CSP, voir [Freuder78, Mackworth77].

L'algorithme de consistancede chemin décrit par Allen fonctionne de la manière suivante: pour chaque triangle (i, j, k) du réseau de contraintes temporelles, pour chaque arête (i, j) de ce triangle, on remplace la contrainte P_{ij} par l'intersection de P_{ij} et de $P_{ik}\ o\ P_{kj}$; cette opération est itérée jusqu'à ce que plus aucune contrainte n'ait été modifiée. La figure 6.6 montre le résultat de l'application de l'algorithme au réseau de la figure 6.5. Un réseau chemin-consistant vérifie: $\forall i, j, k, P_{ij} \subseteq P_{ik}\ o\ P_{kj}$.

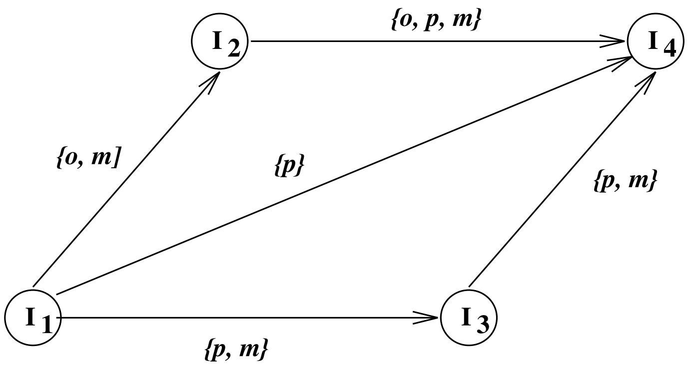


Figure 6.6 - : Réseau chemin-consistant de contraintes temporelles. L'application de l'algorithme de consistance de chemin a permis de réduire le domaine de la relation liant I_1 à I_3.

Une implémentation itérative de cet algorithme est en $O(n^3)$; malheureusement, la consistance de chemin n'implique pas même la satisfaisabilité du graphe de contraintes temporelles. En d'autres termes, cet algorithme est bien sûr correct, mais il n'est pas complet. Allen [Allen83] donne un exemple de graphe consistant mais insatisfaisable (figure 6.7). Ladkin [Ladkin90] présente une version de cet algorithme utilisant la recherche d'un point fixe dans une matrice représentant le réseau de contraintes ; Nebel et Bürckert [Nebel et al.93] donnent un algorithme de calcul de la fermeture réduite d'un réseau Θ obtenu en inférant toutes les relations possibles à partir du réseau initial en se servant de la relation inverse, de l'intersection et de la composition, puis en prenant parmi toutes les relations entre deux variables la plus forte ; le réseau $\hat{\Theta}$ obtenu est chemin-consistant.

Il apparaît donc que la consistance d'un TCSP dans l'algèbre d'intervalles $\mathcal{A}$ est un problème NP-complet [Vilain et al.86]. De même, la recherche de l'étiquetage minimum est également un problème NP-complet.

Si par contre, on passe aux sous-algèbres des ensembles des relations pointisables, la satisfaisabilité du réseau peut être connue en temps polynomial (c'est également vrai pour l'ensemble des relations continues puisque $\mathcal{C} \subseteq \mathcal{P}$). En particulier, la chemin-consistance (en $O(n^3)$) assure cette satisfaisabilité [Vilain et al.86]. En fait, il existe même un algorithme en $O(n^2)$ qui n'est pas basé sur la chemin-consistance et qui vérifie la satisfaisabilité du réseau [vanBeek90b]. Vilain et Kautz [Vilain et al.86] ont prétendu que la chemin-consistance donnait également une solution au problème d'étiquetage minimum ; en réalité, van Beek [vanBeek89] a démontré que ce n'était pas le cas, et a décrit un algorithme réalisant ce travail en $O(n^4)$. Par contre, cette propriété est vraie pour l'ensemble des relations continues.

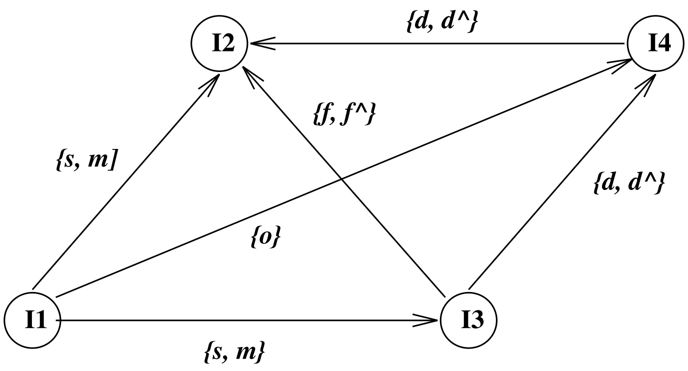


Figure 6.7 - : Exemple de réseau de contraintes temporelles chemin-consistant mais insatisfaisable. On le voit en instanciant successivement la relation entre I_1 et I_4 à f et $\breve{f}$.

Pour terminer sur ce sujet, ajoutons simplement que Nebel et Bürckert [Nebel et al.93] ont exhibé le plus grand sous-ensemble de $\mathcal{A}$ pour lequel le problème de satisfaisabilité se résolvait en temps polynomial (en $O(n^3)$ également). En conséquence, l'étiquetage minimum se résout en $O(n^5)$, car, pour l'algèbre de points, le passage d'un réseau chemin-consistant à la résolution du problème d'étiquetage minimum se fait en $O(n^2)$ [Ladkin90]. De plus, comme cela a déjà été dit, le problème de satisfaisabilité pour l'ensemble Δ_0 des trois relations {intersecte, disjoint, relation universelle}, est NP-complet [Golumbic et al.93].

Tous ces résultats sont résumés dans le tableau 6.3.

Ensemble	Satisfaisabilité	Étiquetage minimum
$\mathcal{A}$	NP-complet	NP-complet
$\mathcal{H}$	$O(n^3)$	$O(n^5)$
$\mathcal{P}$	$O(n^2)$	$O(n^4)$
$\mathcal{C}$	$O(n^2)$	$O(n^3)$
Δ_0	NP-complet	NP-complet

Tableau 6.3 - : Résumé des caractéristiques des problèmes classiques de raisonnement temporel pour différents ensembles.

6.2.2 Raisonnement quantitatif

Nous ne nous attacherons dans cette section qu'aux graphes simplifiés pour lesquels un seul intervalle apparaît dans le réseau et plus précisément à leur traduction en graphe de distances. Mentionnons juste que la consistance d'un réseau de contraintes temporelles dans le cas général est un problème NP-complet.

Le cas simplifié est directement soluble en $O(n^3)$; l'algorithme de Floyd-Warshall utilisé calcule les distances des chemins les plus courts pour tous les couples de nœuds du graphe, et permet par la même occasion de vérifier la consistance du graphe. On obtient au bout du compte les domaines minimaux des variables. L'application de l'algorithme au graphe de la figure 6.4 donne le réseau minimal du tableau 6.4.

	0	1	2	3	4
0	0 [0]	20 [10, 20]	50 [40, 50]	30 [20, 30]	70 [60, 70]
1	-10 [-20, -10]	0 [0]	40 [30, 40]	20 [10, 20]	60 [50, 60]
2	-40 [-50, -40]	-30 [-40, -30]	0 [0]	-10 [-20, -10]	30 [20, 30]
3	-20 [-30, -20]	-10 [-20, -10]	20 [10, 20]	0 [0]	50 [40, 50]
4	-60 [-70, -60]	-50 [-60, -50]	-20 [-30, -20]	-40 [-50, -40]	0 [0]

Tableau 6.4 - : Longueurs des chemins les plus courts dans le graphe de distance et réseau minimal. La première ligne donne la valeur la plus faible de la contrainte entre deux nœuds ; la seconde ligne donne directement l'intervalle de valeurs pour ces deux nœuds (on observe alors une symétrie dans les valeurs).

6.2.3 Intégration des formalismes

Dans cette section, nous parlerons de deux intégrations de contraintes qualitatives et quantitatives, chacune ayant des spécificités intéressantes. L'intérêt évident d'une telle intégration est de disposer de mécanismes de représentation et de raisonnement orthogonaux qui se complètent mutuellement. La première intégration [Kautz et al.91] définit une double traduction entre l'algèbre d'Allen et les relations quantitatives simples définies par [Dechter et al.91] (c'est-à-dire en excluant les unions d'intervalles) ; la seconde [Meiri91] décrit plutôt un formalisme commun permettant d'y exprimer à la fois des relations qualitatives (incluant celles entre intervalles et points et entre deux points) et des relations quantitatives avec union d'intervalles. Néanmoins, ce second formalisme semble moins puissant dans ses capacités déductives comme cela sera montré.

Kautz et Ladkin intègrent donc les 2^{13} disjonctions de l'algèbre d'intervalles d'Allen $\mathcal{A}$ et les relations quantitatives simples de la forme $m \leq x - y \leq n$ où x et y sont des extrémités d'intervalles éventuellement différents. À partir de deux réseaux de contraintes initiaux, leur algorithme calcule les réseaux minimaux (ou la seule consistance de chemin dans le cas du réseau de contraintes qualitatives, puisque la détermination du réseau minimal est un problème NP-complet) et traduit chacune des relations d'un formalisme dans l'autre, et répète ce processus jusqu'à l'obtention de la stabilité. L'algorithme est précisé dans le tableau 6.5.

Il s'agit alors de définir les procédures de traduction d'un formalisme à l'autre en perdant le moins d'information possible. La traduction des relations du réseau de contraintes quantitatives vers des relations qualitatives est plus compliquée qu'il n'y paraît, car, contrairement à l'intuition première, il ne suffit pas de considérer les relations impliquant les extrémités d'un même intervalle. En effet, si $3 < I^+ - I^- < \infty$ et $-\infty < J^+ - J^- < 2$, les relations qualitatives inférées ne contiennent pas celle qui dit que I ne peut pas être pendant J puisque I dure plus longtemps que J. Heureusement, il suffit en fait de considérer les couples d'intervalles (et non pas n'importe quel n-uplet) pour que les plus fortes contraintes qualitatives soient inférées. L'algorithme *quanti* $\rightarrow$ *quali* du tableau 6.6 fait ce travail de traduction en $O(n^3)$, n étant le nombre d'intervalles.

Algorithme 1 *Allen+métrique(M, A)*

Entrées : un réseau métrique simple M et un réseau de relations entre intervalles A
Sorties : les réseaux M' et A' impliqués par $M \cup A$

Répéter
$A' := quanti \rightarrow quali(M) \cup A$
$M' := quali \rightarrow quanti \cup M$
$M := M'; A := A'$
Jusqu'à ce que $A = A'$ *et* $M = M'$
Rendre A' *et* M'

Tableau 6.5 - : Comment combiner les relations d'Allen avec des relations quantitatives simples? À partir de deux réseaux de contraintes, l'un qualitatif A et l'autre quantitatif M, on opère les traductions de ces réseaux jusqu'à la stabilité.

Réciproquement, la traduction des relations du réseau de contraintes qualitatives vers des relations quantitatives est un problème NP-complet (pour la simple raison que la consistance du premier est NP-complet et celle du second polynomiale). L'algorithme présenté énumère les couples d'intervalles reliés par une relation complexe et ne garde que les relations quantitatives de la forme $x - y < 0$ consistantes avec chacune des relations atomiques de la relation complexe (tableau 6.7). Malheureusement, cet algorithme n'est complet que si le réseau qualitatif minimal a été déterminé, ce qui demande un temps exponentiel.

L'article de Meiri [Meiri91] a pour objectif la définition d'un formalisme qui unifie les différentes représentations du temps, à savoir l'algèbre d'intervalles d'Allen, l'algèbre de points de Vilain et Kautz, ainsi que les relations point-intervalle et point-point, et l'ensemble des relations quantitatives définies par Dechter *et al.* (en ne se limitant donc pas aux réseaux simples comme précédemment).

La première étape de la formalisation consiste en la définition d'une algèbre intégrant toutes les relations qualitatives, possédant en particulier une table de transitivité complète, adjoignant à celles d'Allen et de Vilain et Kautz celle pour les relations entre point et intervalle. La seconde est la détermination de règles d'inférence permettant de passer de relations qualitatives aux relations quantitatives, et réciproquement. Celles-ci sont très simples et se limitent à celles impliquant les relations quantitatives et les relations qualitatives entre deux points (tableau 6.8). En particulier, elles ne semblent pas tenir compte du point soulevé avant sur la nécessité d'intégrer plusieurs relations métriques pour inférer l'ensemble des relations qualitatives impliquées.

La représentation de l'ensemble des contraintes, qualitatives ou quantitatives, se fait dans un réseau de contraintes comme celui de la figure 6.8 dans lequel les contraintes entre points ont systématiquement été transformées en contraintes quantitatives grâce aux règles du tableau 6.8. De plus, il existe des contraintes internes exprimant que l'origine d'un intervalle commence cet intervalle et que la fin le termine.

Algorithme 2 *quanti → quali(M)*

Entrées : un réseau métrique simple M
Sorties : le réseau de contraintes d'intervalles le plus fort impliqué par M

Soit M' le réseau minimal associé à M
$A_M := \emptyset$
Pour *toute paire d'intervalles I, J,* **faire**
  Soit $S = \{I^-, I^+, J^-, J^+\}$, un sous-réseau de M'
  $R := \emptyset$
  Pour *chaque relation d'Allen r,* **faire**
    $S' := S \cup \{m$ *tels que m est une inégalité impliquée par r dans la traduction de r*$\}$
    Si *S' est consistent,* **alors** $R := R \cup \{r\}$
    Fin Si
  Fin Pour
  $A_M := A_M \cup \{I(R)J\}$
Fin Pour
Rendre A_M

Tableau 6.6 - : Conversion de contraintes métriques en relations d'Allen. Cet algorithme qui explore chaque couple d'intervalles en ne gardant que les relations qualitatives non contradictoires avec le réseau de contraintes quantitatives, est complet et a une complexité en $O(n^3)$.

Meiri ne s'intéresse alors qu'aux réseaux de contraintes dont la résolution n'est pas un problème NP-complet, de manière à développer des algorithmes polynomiaux. Il réalise cela en restreignant le langage de représentation des contraintes et en ne calculant que des consistances d'arc et de chemin. Nous n'entrerons pas dans les détails, d'autant plus que nous nous inspirerons de l'intégration précédente, plus complète et plus aisément applicable aux cartes génomiques (Cf. section 7.3.3).

6.3 Temps et cartes génomiques

La description des mécanismes de représentation et de raisonnement temporels qui précède permet de mieux comprendre les similitudes, et surtout les différences, qui existent entre la modélisation du temps et celle des cartes génomiques. Nous allons désormais en faire la liste, pour déterminer les éléments dont il sera possible de se servir.

6.3.1 Les similitudes

- Expression de relations qualitatives : la modélisation des cartes génomiques implique la définition d'un certain nombre de relations qualitatives entre entités d'un axe orienté. Ainsi, on retrouve un sous-ensemble des relations qualitatives d'Allen, et

Algorithme 3 *quali → quanti(M)*

Entrées : un réseau de relations entre intervalles A
Sorties : le réseau métrique simple le plus fort impliqué par A

Soit A' le réseau minimal associé à A
$M_A := \emptyset$
Pour *toute paire d'intervalles I, J,* **faire**
  Soit R la relation liant I à J dans A'
  Soit $S = \{m$ tels que $m = x - y < 0, x, y \in \{I^-, I^+, J^-, J^+\}\}$
  Pour *tout élément r de R,* **faire**
    $S := S \cap \{m$ *tels que m est une inégalité impliquée par* $(IrJ)\}$
  Fin Pour
  $M_A := M_A \cup S$
Fin Pour
Rendre M_A

Tableau 6.7 - : Conversion de contraintes d'Allen en contraintes métriques. Pour chaque relation entre deux intervalles, la relation métrique impliquée est celle qui satisfait chacune des relations atomiques contenues dans la relation entre ces deux intervalles. Si on ne part pas du réseau minimal, l'algorithme (en $O(n^2)$) n'est pas complet.

aussi des relations entre extrémités de ces intervalles. En conséquence, l'ensemble des relations qualitatives exprimées sur l'ensemble des entités et l'ensemble des points de vue peut naturellement se représenter dans un réseau de contraintes.

- Expression de relations quantitatives : une relation quantitative sur les cartes génomiques nécessite les données de distance et d'incertitude ; celles-ci se transcrivent aisément dans le formalisme de Dechter *et al.* En effet, on a l'équivalence suivante : $(P_i \leftrightarrow P_j = [d_{ij}, i_{ij}]) \Leftrightarrow d_{ij} - i_{ij} \leq P_j - P_i \leq d_{ij} + i_{ij}$. Comme précédemment, mais à l'intérieur d'un même point de vue, les relations quantitatives peuvent être représentées à l'aide d'un réseau de contraintes semblable à ceux qui modélisent le temps. Il faudra de plus travailler sur des ensembles ordonnés de positions pour bénéficier de l'additivité des distances.

Les travaux sur la représentation du temps ont souvent été dirigés par les formalismes, c'est-à-dire la recherche d'ensembles de relations tels que les problèmes classiques de satisfaisabilité et de résolution soient calculables en temps polynomial. Par contre, dans le cas présent, le problème est soumis à l'application biologique, et c'est elle qui fixe les relations à exprimer et qui guide la formalisation. Avant de disposer de résultats mathématiques, il est primordial de coller au problème réel. C'est pour cette raison que les différences énoncées ci-après doivent être prises en compte pour modéliser les cartes génomiques.

C est une contrainte quantitative entre deux points P_i et P_j représentée par une union d'intervalles $\{I_1, \ldots, I_k\}$.
$Qual(C)$ est la relation qualitative impliquée par C.

- Si $0 \in \{I_1, \ldots, I_k\}$, alors $= \in Qual(C)$;
- Si $\exists\, v > 0$ tel que $v \in \{I_1, \ldots, I_k\}$, alors $< \in Qual(C)$;
- Si $\exists\, v < 0$ tel que $v \in \{I_1, \ldots, I_k\}$, alors $> \in Qual(C)$;

C est une contrainte qualitative entre deux points P_i et P_j représentée par une union de relations R.
$Quan(C)$ est la relation quantitative impliquée par C.

- Si $< \in R$, alors $(0, +\infty) \in Quan(C)$;
- Si $= \in R$, alors $(0) \in Quan(C)$;
- Si $> \in R$, alors $(-\infty, 0) \in Quan(C)$;

Tableau 6.8 - : Règles d'inférence entre une relation quantitative entre deux points et une relation qualitative entre ces mêmes points. Suivant qu'une valeur, à positionner par rapport à 0, appartient ou non à la contrainte quantitative, la contrainte qualitative impliquée contient une des relations atomiques $<, >, =$; réciproquement, selon qu'une contrainte qualitative contient ou non une des relations atomiques $<, >, =$, la contrainte quantitative contient un intervalle contenant 0 ou non.

6.3.2 Les différences

- Un axe non orienté : cette grande différence a déjà été soulignée ; elle est très importante car elle implique d'énormes modifications au formalisme de représentation du temps. Comme les cartes génomiques sont dépourvues d'orientation globale, contrairement au temps qui s'écoule dans un sens, un certain nombre de relations qualitatives doivent contenir une indication de leur orientation par rapport à une autre entité, comme cela a été défini dans le chapitre 2.3. De plus, les relations quantitatives sont également concernées par ce point puisque les distances aussi ne sont pas orientées, et qu'une valeur (toujours positive) n'indique donc pas de positionnement précis par rapport à une orientation donnée.

- Les points de vue : la gestion des points de vue est une charge importante dans la modélisation des cartes génomiques, et ce sur deux points particuliers : d'abord, le fait que les entités qui se trouvent sur plusieurs points de vue peuvent passer d'intervalle à point et réciproquement, ce qui oblige lors de la définition de relations qualitatives, à définir les changements autorisés d'une représentation de l'entité à l'autre (problème de granularité [Euzenat93, Euzenat94]) ; ensuite, les distances ne sont valides qu'à l'intérieur d'un même point de vue, en l'absence de facteur d'échelle entre points de vue. Il est donc parfois obligatoire de travailler à l'intérieur d'un point

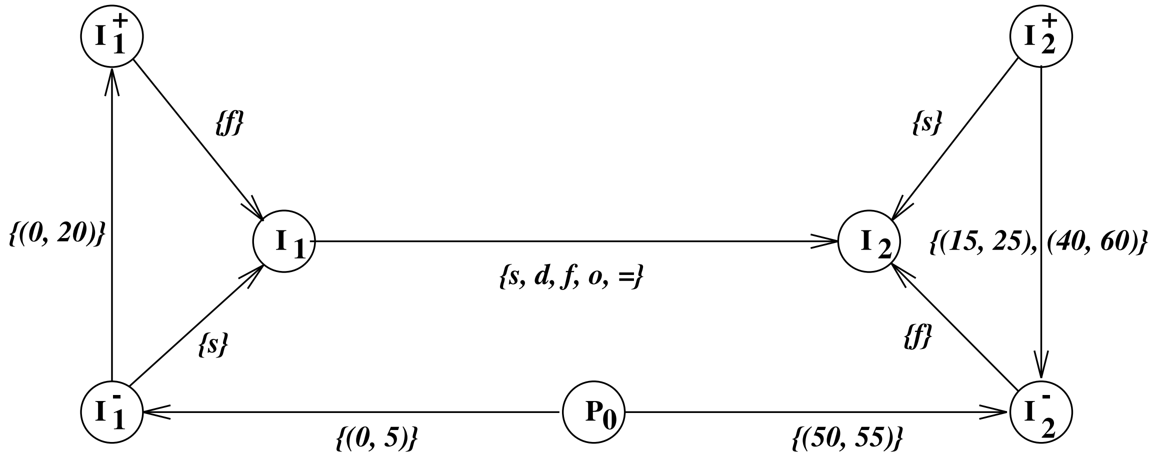


Figure 6.8 - : Exemple de réseau de contraintes temporelles intégrant des informations quantitatives et qualitatives. Les relations liant un intervalle à ses extrémités sont systématiques.

de vue et parfois de propager des résultats d'un point de vue à un autre.

- Un sous-ensemble de $\mathcal{A}$: seul un certain nombre de relations de l'algèbre d'Allen sont autorisées. Cela peut modifier les algorithmes utilisés en représentation du temps, en particulier si les propriétés de ces algorithmes se basent sur les caractéristiques de $\mathcal{A}$, que l'ensemble des relations qualitatives sur les cartes génomiques ne possède plus, comme par exemple le fait que ce soit une algèbre. Nous verrons dans le chapitre suivant comment traiter ce problème.

- Les incertitudes expérimentales : contrairement aux problèmes de représentation du temps, tels qu'ils sont spécifiés, les cartes génomiques, parce qu'elles sont issues d'expériences, sont entachées d'erreur. Conséquemment, les réseaux de contraintes susceptibles d'être extraits des informations expérimentales sont probablement inconsistants. Comment alors gérer ces inconsistances tout en continuant à raisonner ? Nous verrons comment assurer des consistances locales à des ensembles d'entités.

Chapitre 7

Algorithmique des cartes génomiques

Ce chapitre s'inspire de techniques de raisonnement temporel pour développer des algorithmes de construction de cartes, fondés sur le formalisme présenté. Pour cela, il faut préciser les mécanismes d'inférence liés aux relations qu'il est possible d'exprimer entre les entités ; la première section détaillera une table de transitivité sur les relations cartographiques, ainsi que des règles de passage entre les représentations point-intervalle. La seconde section traitera des relations quantitatives et des moyens de les regrouper de telle façon que l'additivité des distances soit assurée. Enfin, un algorithme sera présenté.

7.1 Expression des relations qualitatives entre entités cartographiques

Exprimer des relations qualitatives entre entités cartographiques comporte une complexité intrinsèque liée à l'existence de différents points de vue sur lesquels une même entité dispose de représentations éventuellement distinctes (point ou intervalle). Néanmoins, comme les relations qualitatives sont conservées d'un point de vue à l'autre, le passage d'une représentation par un intervalle à celle par un point (et réciproquement) est conditionné par des règles.

Dans un premier temps, nous détaillerons les multiples relations (intervalle-intervalle, point-intervalle et point-point) entre entités dans un unique point de vue, puis nous préciserons les règles permettant de passer d'une relation dans une représentation à une relation dans l'autre.

7.1.1 Relations intervalle-intervalle

Nous avons vu à la section 4.2.1 que les relations qu'il était intéressant d'exprimer en cartographie ne tenaient pas compte de la position précise des extrémités les unes par rapport aux autres, c'est-à-dire que les inégalités étaient toujours larges. Le tableau 7.1 montre les relations cartographiques en les exprimant dans le formalisme d'Allen. Il s'appuie sur le réseau de Nökel [Nökel88] qui met en évidence les passages continus d'une relation d'Allen à une autre (figure 7.1).

Relation carto	Équivalent Allen	Inverse	Inverse Allen
avant $\preceq$	$\{b, m\}$	après $\succeq$	$\{\breve{b}, \breve{m}\}$
recouvre_avant $\ll$	$\{m, o, s, \breve{f}, =\}$	recouvre_après $\gg$	$\{\breve{m}, \breve{o}, \breve{s}, f, =\}$
contenu $\sqsubseteq$	$\{d, s, f, =\}$	contient $\sqsupseteq$	$\{\breve{d}, \breve{s}, \breve{f}, =\}$
disjoint_de $\neq$	$\{b, \breve{b}, m, \breve{m}\}$	disjoint_de $\neq$	$\{b, \breve{b}, m, \breve{m}\}$
non_disjoint_de $\bowtie$	$\{o, \breve{o}, m, \breve{m}, s, \breve{s}, d, \breve{d}, f, \breve{f}, =\}$	non_disjoint_de $\bowtie$	$\{o, \breve{o}, m, \breve{m}, s, \breve{s}, d, \breve{d}, f, \breve{f}, =\}$

Tableau 7.1 - : Les relations cartographiques en termes de celles d'Allen. Les deux premières relations (et leur inverse) nécessitent une entité constitutive définissant un ordre local. Les deux dernières relations *disjoint_de* et *non_disjoint_de* peuvent être exprimées comme des disjonctions des précédentes ; en effet, $\neq = \{\preceq, \succeq\}$ et $\bowtie = \{\ll, \gg, \sqsubseteq, \sqsupseteq\}$. Il est important de noter alors que $\neq$ et $\bowtie$ sont définies à l'aide de relations qualitatives orientées, alors qu'elles-mêmes n'en sont pas. Cela signifie que la définition est valide pour toute entité de référence contenant les entités en relation. Cette propriété vient du fait que les deux relations incluent une relation qualitative orientées et son inverse.

Ainsi, une relation cartographique part d'une relation d'Allen exprimée avec des inégalités strictes sur les extrémités et est prolongée en lui ajoutant les relations d'Allen obtenues en étendant les extrémités des deux intervalles jusqu'à ce que les inégalités strictes deviennent des égalités. Comme on s'autorise à prolonger les extrémités des deux intervalles, on va plus loin que le réseau de Nökel, qui n'en prolonge qu'une à la fois. Les relations de départ sont mises en gras et les ensembles obtenus par ce processus sont entourés d'une ligne pointillée dans la figure 7.1.

L'ensemble des six relations est noté $\mathcal{R}_{carto}$ par la suite. L'ensemble des disjonctions de ces relations «atomiques» sera appelé $\mathcal{C}$. Il contient 2^6 relations, puisque $\mathcal{R}_{carto}$ en contient six.

Les relations de $\mathcal{R}_{carto}$ ne sont pas disjointes ; on a par exemple $\preceq \cap \ll = \{m\}$. Néanmoins, si on ne tient pas compte des relations impliquant une égalité entre les extrémités, on peut s'accorder sur la disjonction ; en effet, l'intersection de deux relations cartographiques est une relation d'Allen (ou une disjonction de relations d'Allen) qui met en jeu des égalités sur les extrémités. De plus, l'union de toutes les relations est bien la relation universelle $\top$. Sous ces conditions, $\mathcal{C}$ est stable sous les opérations d'union et d'intersection. Qu'en est-il de la composition ? Pour le savoir, on peut utiliser la table de composition d'Allen et voir si le résultat obtenu se trouve toujours dans $\mathcal{C}$. Ces résultats sont montrés dans le tableau 7.2.

On remarque qu'il n'y pas stabilité dans la composition. Néanmoins, toutes les relations d'Allen qui apparaissent contiennent une relation d'égalité au moins sur les extrémités (comme pour l'intersection), et on peut étendre le résultat trouvé par des relations en continuité avec celles vérifiées, comme par exemple lors de la composition de $\preceq$ avec lui-même (qui élimine la relation m) ; il suffit dans ce cas de remplacer la relation b par celle de $\mathcal{R}_{carto}$ qui la contient, à savoir $\preceq$. De même pour la composition de $\preceq$ et de $\sqsubseteq$, pour laquelle il faut ajouter f, $\breve{f}$ et $=$. On obtient alors la table de composition suivante (tableau 7.3), qui n'est pas exacte, mais qui n'infère jamais quelque chose de faux ; elle

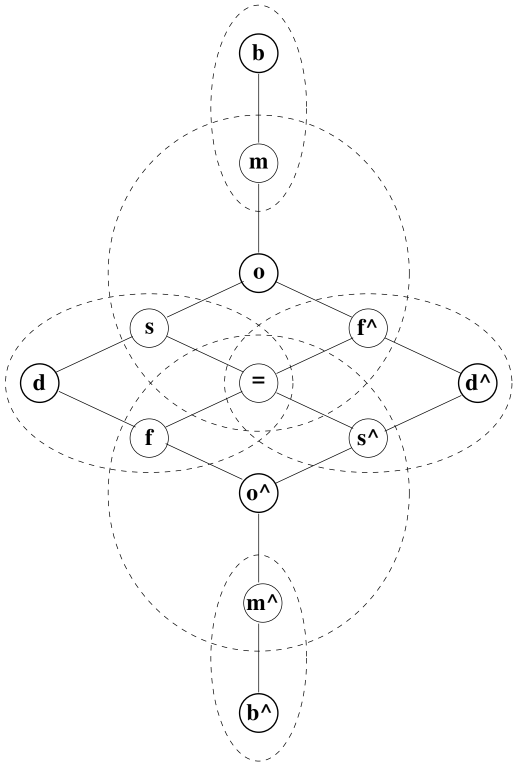


Figure 7.1 - : Le réseau de Nökel pour $\mathcal{R}_{Allen}$. Il montre des propriétés de convexité sur les relations temporelles, telles qu'une relation est liée à une autre si elle peut être obtenue en déplaçant continûment une extrémité d'un intervalle en relation. Par exemple, la relation m est liée à b car il suffit de déplacer la fin du premier intervalle vers la droite (ou le début du second vers la gauche) pour atteindre le début du second (ou la fin du premier), et obtenir ainsi cette dernière relation. Les relations entourées d'un cercle sont les treize relations d'Allen ; les relations cartographiques sont les disjonctions des relations entourées d'une ligne pointillée.

est imprécise dans certains cas. Elle a l'avantage de donner une relation de composition stable pour $\mathcal{R}_{carto}$.

7.1.2 Relations point-intervalle et intervalle-point

Pour des raisons de simplification des mécanismes de raisonnement, un *point* sera dorénavant une entité ayant une dimension nulle dans chaque point de vue où elle est définie ; un *intervalle* est une entité qui a une dimension non nulle dans au moins un des points de vue où elle apparaît. Sans cette simplification, on est contraint de définir des relations particulières liant intervalle à intervalle, point à intervalle et intervalle à point, et pour finir, point à point. Désormais, les mêmes symboles de relation sont conservés pour exprimer toutes ces relations, sachant que leur comportement dépend en partie des dimensions des entités liées.

De plus, comme il n'existe qu'une relation qualitative entre deux entités, indépendamment de leurs représentations dans différents points de vue, il est raisonnable de choisir les représentations les plus fines, c'est-à-dire les intervalles à chaque fois que c'est possible. Néanmoins, il doit toujours être possible d'exprimer une relation qualitative sur un point

o	$\preceq$	$\succeq$	$\ll$	$\gg$	$\sqsubseteq$	$\sqsupseteq$
$\preceq$	$\{b\}$	$\top$	$\{b, m\}$	$\{b, m, o, d, s, f, \breve{f}, =\}$	$\{b, m, o, d, s\}$	$\{b, m\}$
$\succeq$	$\top$	$\{\breve{b}\}$	$\{\breve{b}, \breve{m}, \breve{o}, d, s, \breve{s}, f, =\}$	$\{\breve{b}, \breve{m}\}$	$\{\breve{b}, \breve{m}, \breve{o}, d, f\}$	$\{\breve{b}, \breve{m}\}$
$\ll$	$\{b, m\}$	$\{\breve{b}, \breve{m}, \breve{o}, \breve{d}, \breve{s}, f, \breve{f}, =\}$	$\{b, m, o, s, \breve{f}, =\}$	$\{m, \breve{m}, o, \breve{o}, d, \breve{d}, s, \breve{s}, f, \breve{f}, =\}$	$\{m, o, d, s, f, \breve{f}, =\}$	$\{b, m, o, \breve{d}, s, \breve{s}, \breve{f}, =\}$
$\gg$	$\{b, m, o, \breve{d}, s, \breve{s}, \breve{f}, =\}$	$\{\breve{b}, \breve{m}\}$	$\{m, \breve{m}, o, \breve{o}, d, \breve{d}, s, \breve{s}, f, \breve{f}, =\}$	$\{\breve{b}, \breve{m}, \breve{o}, \breve{s}, f, =\}$	$\{\breve{m}, \breve{o}, d, s, \breve{s}, f, =\}$	$\{\breve{b}, \breve{m}, \breve{o}, \breve{d}, \breve{s}, f, \breve{f}, =\}$
$\sqsubseteq$	$\{b, m\}$	$\{\breve{b}, \breve{m}\}$	$\{b, m, o, d, s, f, \breve{f}, =\}$	$\{\breve{b}, \breve{m}, \breve{o}, d, s, \breve{s}, f, =\}$	$\{d, s, f, =\}$	$\top$
$\sqsupseteq$	$\{b, m, o, \breve{d}, \breve{f}\}$	$\{\breve{b}, \breve{m}, \breve{o}, \breve{d}, \breve{s}\}$	$\{m, o, \breve{d}, s, \breve{s}, \breve{f}, =\}$	$\{\breve{m}, \breve{o}, \breve{d}, \breve{s}, f, \breve{f}, =\}$	$\{o, \breve{o}, d, \breve{d}, s, \breve{s}, f, \breve{f}, =\}$	$\{\breve{d}, \breve{s}, \breve{f}, =\}$

Tableau 7.2 - : Table de composition pour les relations cartographiques «atomiques». Pour le moment, on ne s'occupe pas du problème d'orientation, qui sera traité par la suite. Cette table est donc valide pour peu que les relations aient une entité de référence englobante commune.

de vue particulier, parce que l'expérience qui l'engendre est d'un type particulier. Pour assurer cela, on définira des règles de passage qui, d'une relation entre deux entités de dimensions données, associeront la relation induite sur des entités de dimensions différentes (Cf. §7.1.4).

L'ensemble des relations point-intervalle se limite à trois éléments qui expriment que le point est avant, après ou est contenu dans l'intervalle. De même, un intervalle est avant, après ou contient un point. Nous conserverons les notations précédentes, c'est-à-dire $P\{\preceq, \succeq, \sqsubseteq\}I$ et $I\{\preceq, \succeq, \sqsupseteq\}P$.

Qu'en est-il alors de la table de transitivité ? Il apparaît que, quelle que soit la dimension des entités en relation, la table est utilisable en conjonction avec les règles de passage (Cf. §7.1.4). Chaque relation donnée par la table est remplacée par celle(s) donnée(s) par la règle correspondante. Il suffit après d'enlever les doublons de l'ensemble final.

Par exemple, si les relations sont $e_1 \preceq e_2$ et $e_2 \sqsubseteq e_3$, alors la relation qui lie e_1 et e_3 dépend de leur dimension : si e_1 et e_3 sont tous deux des intervalles, on a bien sûr la relation de la table $\{\preceq, \ll, \sqsubseteq\}$; si e_1 est un point et e_3 un intervalle, l'application des règles pour chacune des relations donne respectivement $\preceq$, $\{\preceq, \sqsubseteq\}$ et $\sqsubseteq$; on obtient donc pour finir l'ensemble $\{\preceq, \sqsubseteq\}$ (figure 7.2). Par contre, il n'est pas possible que e_1 soit un intervalle et e_3 un point, car aucune dimension pour e_2 n'aurait convenu pour exprimer les relations.

7.1.3 Relations point-point

L'ensemble des relations point-point est aussi restreint à trois éléments qui sont $\{\preceq, \succeq, =\}$. L'égalité est introduite dans cet ensemble car elle est nécessaire pour exprimer l'identité de deux points (pouvant être fictifs), ce qui sert pour utiliser des relations quantitatives.

o	$\preceq$	$\succeq$	$\ll$	$\gg$	$\sqsubseteq$	$\sqsupseteq$
$\preceq$	$\preceq$	$\top$	$\preceq$	$\{\preceq,\ll,\sqsubseteq\}$	$\{\preceq,\ll,\sqsubseteq\}$	$\preceq$
$\succeq$	$\top$	$\succeq$	$\{\succeq,\gg,\sqsubseteq\}$	$\succeq$	$\{\succeq,\gg,\sqsubseteq\}$	$\succeq$
$\ll$	$\preceq$	$\{\succeq,\gg,\sqsupseteq\}$	$\{\preceq,\ll\}$	$\{\ll,\gg,\sqsubseteq,\sqsupseteq\}$	$\{\ll,\sqsubseteq\}$	$\{\preceq,\ll,\sqsupseteq\}$
$\gg$	$\{\preceq,\ll,\sqsupseteq\}$	$\succeq$	$\{\ll,\gg,\sqsubseteq,\sqsupseteq\}$	$\{\succeq,\gg\}$	$\{\gg,\sqsubseteq\}$	$\{\succeq,\gg,\sqsupseteq\}$
$\sqsubseteq$	$\preceq$	$\succeq$	$\{\preceq,\ll,\sqsubseteq\}$	$\{\succeq,\gg,\sqsubseteq\}$	$\sqsubseteq$	$\top$
$\sqsupseteq$	$\{\preceq,\ll,\sqsupseteq\}$	$\{\succeq,\gg,\sqsupseteq\}$	$\{\ll,\sqsupseteq\}$	$\{\gg,\sqsupseteq\}$	$\{\ll,\gg,\sqsubseteq,\sqsupseteq\}$	$\sqsupseteq$

Tableau 7.3 - : Composition de relations cartographiques. On remarque également ici que la composition de $\sqsupseteq$ et de $\sqsubseteq$, deux relations sans entité de référence, donne des relations qualitatives orientées. Le sens de cette pseudo-contradiction est le même que celui mentionné auparavant (figure 7.1), d'autant plus que le résultat est $\bowtie$. D'autres compositions sont aussi dans ce cas comme $\sqsubseteq\ o \ll$ qui donne $\{\preceq,\ll,\sqsubseteq\}$; là, le sens est différent car une des relations de la composition ($\ll$) spécifie un ordre. Alors, la relation donnée par la table est valide pour toute entité contenant les deux entités en relation et de même orientation que l'entité de référence de la relation qualitative orientées. Enfin, il faut que les compositions de deux relations orientées aient pour entités de référence des entités de même orientation.

7.1.4 Passage d'une représentation à l'autre

Les règles de passage définies ci-après sont basées sur la transformation d'un intervalle en point, et réciproquement, d'un point de vue à un autre ; cette transformation est contrainte par l'appartenance du point à l'intervalle équivalent : le point associé à un intervalle est contenu dans cet intervalle, et réciproquement, l'intervalle associé à un point le contient. Pour chacune des relations de chacun des types (intervalle-intervalle, point-intervalle, intervalle-point, point-point) liés par cette relation, nous allons détailler la relation impliquée sur les autres types. Les tableaux 7.4, 7.5, 7.6 sont un synoptique de ces transformations.

$I_1 r I_2$	$P_1 r I_2$	$I_1 r P_2$	$P_1 r P_2$
$\preceq$	$\preceq$	$\preceq$	$\preceq$
$\succeq$	$\succeq$	$\succeq$	$\succeq$
$\ll$	$\{\preceq,\sqsubseteq\}$	$\{\preceq,\sqsupseteq\}$	$\top$
$\gg$	$\{\succeq,\sqsupseteq\}$	$\{\succeq,\sqsubseteq\}$	$\top$
$\sqsubseteq$	$\sqsubseteq$	$\top$	$\top$
$\sqsupseteq$	$\top$	$\sqsupseteq$	$\top$

Tableau 7.4 - : Règles de transformation d'une relation intervalle-intervalle vers des relations point-intervalle et point-point.

Toutes ces règles de transformation permettent d'une part d'établir une table de transitivité globale aux points et intervalles, d'autre part de normaliser les relations entre entités lorsque celles-ci ont des dimensions différentes dans les points de vue où elles apparaissent ; ce dernier point est réalisé en traduisant la relation en celle qui implique les

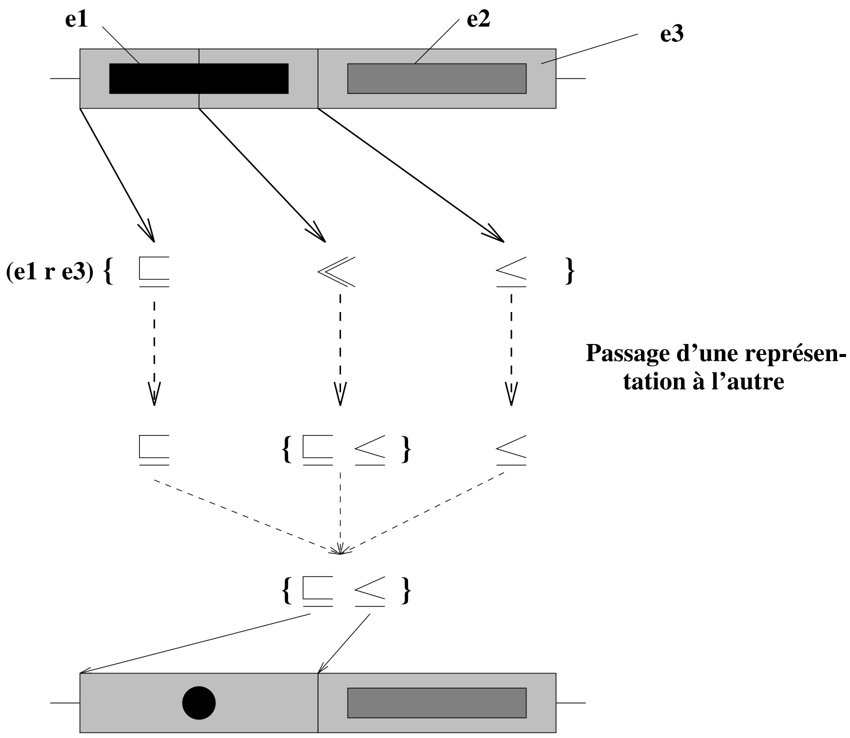


Figure 7.2 - : Utilisation de la table de transitivité pour les intervalles et les points. Suivant les dimensions des entités extrêmes, le résultat de la table de transitivité demande à être modifié; si les entités e_1 et e_3 sont des intervalles, on prend le résultat de la table tel quel (ici, $\{\preceq, \ll, \sqsubseteq\}$); si e_1 est un point et e_3 un intervalle, l'application des règles de passage d'intervalle à point (table 7.4) donne les relations adéquates.

dimensions les plus fines, à savoir les intervalles.

7.2 Relations quantitatives

Les relations quantitatives telles qu'elles ont été définies à la section 6.1.3 s'intègrent naturellement dans le formalisme proposé par Dechter *et al* présenté à la section 6.2.2. En effet, une relation quantitative exprimant dans un point de vue donné une distance et une incertitude contient exactement la même information qu'une double inégalité entre deux positions. Ainsi, $(P_i \leftrightarrow P_j = [d_{ij}, i_{ij}])$ est équivalent à $d_{ij} - i_{ij} \leq P_j - P_i \leq d_{ij} + i_{ij}$. Cependant, il faut faire attention à ce que les relations quantitatives positionnées dans un réseau de contraintes aient la même orientation, puisque l'axe des cartes génomiques possède deux sens. Le formalisme de Dechter *et al.* ne s'applique en effet que dans le cas où les distances sont additives, soit, dans notre cas, pour des relations quantitatives exprimées dans la même direction. Remarquons néanmoins qu'une telle obligation n'implique pas de connaître précisément l'ordre des positions en relation, mais l'orientation relative des extrémités des relations quantitatives.

La section 7.3.3, qui présente plus en détail l'algorithme de construction de cartes, montrera les mécanismes à même de regrouper les relations quantitatives de telle sorte à pouvoir profiter de l'additivité.

$P_1 r I_2$	$I_1 r I_2$	$I_1 r P_2$	$P_1 r P_2$
$\preceq$	$\{\preceq, \ll, \sqsupseteq\}$	$\{\preceq, \sqsupseteq\}$	$\preceq$
$\succeq$	$\{\succeq, \gg, \sqsupseteq\}$	$\{\succeq, \sqsupseteq\}$	$\succeq$
$\sqsubseteq$	$\{\ll, \gg, \sqsupseteq, \sqsubseteq\}$	$\top$	$\top$
$I_1 r P_2$	$I_1 r I_2$	$P_1 r I_2$	$P_1 r P_2$
$\preceq$	$\{\preceq, \ll, \sqsubseteq\}$	$\{\preceq, \sqsubseteq\}$	$\preceq$
$\succeq$	$\{\succeq, \gg, \sqsubseteq\}$	$\{\succeq, \sqsubseteq\}$	$\succeq$
$\sqsupseteq$	$\{\ll, \gg, \sqsupseteq, \sqsubseteq\}$	$\top$	$\top$

Tableau 7.5 - : Règles de transformation d'une relation intervalle-point (respectivement point-intervalle) vers des relations intervalle-intervalle, point-intervalle (respectivement intervalle-point) et point-point. Ici encore, la troisième ligne de chaque table infère des relations orientées alors que la relation de départ n'en est pas une ; on retrouve un des cas déjà rencontré, puisque la relation donnée est équivalente à la relation *non_disjoint_de* ($\bowtie$).

$P_1 r P_2$	$I_1 r I_2$	$P_1 r I_2$	$I_1 r P_2$
$\preceq$	$\{\preceq, \ll, \sqsubseteq, \sqsupseteq\}$	$\{\preceq, \sqsubseteq\}$	$\{\preceq, \sqsupseteq\}$
$\succeq$	$\{\succeq, \gg, \sqsubseteq, \sqsupseteq\}$	$\{\succeq, \sqsubseteq\}$	$\{\succeq, \sqsupseteq\}$

Tableau 7.6 - : Règles de transformation d'une relation point-point vers des relations intervalle-intervalle et point-intervalle.

7.3 Algorithme de construction de cartes

L'algorithme de construction de cartes est en réalité modulable en fonction du choix de la résolution des contraintes, de la même façon que, à partir d'un réseau de contraintes temporelles, on peut juste vérifier la consistance de chemin, déterminer la satisfaisabilité ou calculer le réseau minimal. Avant d'en expliquer le fonctionnement, nous allons récapituler les informations disponibles et sous quelle forme, ainsi que les mécanismes d'inférence qui ont été vus dans les chapitres précédents.

7.3.1 Les connaissances : entités et relations

L'énumération qui suit rassemble les éléments constitutifs du problème de construction de cartes tels qu'ils ont été définis lors de la formalisation.

1. Les entités :

   (a) $\mathcal{E} = \{e_i\}$, un ensemble d'entités typées organisées en hiérarchies de composition (une par point de vue) par l'intermédiaire d'une fonction *desc* ;

   (b) $Pos = \{P_i\} = \{O_i\} \cup \{F_i\}$, un ensemble de positions associées aux entités de $\mathcal{E}$ à travers les fonctions $orig$, fin et $entit$.

2. Les relations :

   (a) Relations qualitatives : $\mathcal{R}_{quali} = \mathcal{R}^+_{quali} \cup \mathcal{R}^-_{quali}$, un ensemble de relations qualitatives de la forme suivante : $\mathcal{R}^+_{quali} = \{(e_i\ t^+_e\ e_j)\}$ et $\mathcal{R}^-_{quali} = \{(e_i\ t^-\ e_j)\} \cup \{ordre(e_k, \ldots, e_{k+n})\}$, avec $e_i, e_j, e_k, \ldots e_{k+n} \in \mathcal{E}, t^+ \in \mathcal{Q}^+_l = \{\preceq, \succeq, \ll, \gg\}, t^- \in \mathcal{Q}^-_l = \{\sqsubseteq, \sqsupseteq, \neq, \bowtie, \rightrightarrows, \rightleftarrows\}$, où $\rightrightarrows$ représente la relation de même orientation et $\rightleftarrows$ la relation d'orientation opposée.

   (b) Relations quantitatives : $\mathcal{R}_{quanti} = (P_i \leftrightarrow_p P_j = [d_{ij}, i_{ij}])$, un ensemble de relations quantitatives où $P_i, P_j \in Pos, p \in \mathcal{P}$.

7.3.2 Les inférences

Dans ce paragraphe sont énumérés tous les mécanismes d'inférence utilisés par les algorithmes. Ceux-ci seront référencés par leur numéro dans l'énumération.

1. Fonction $desc$: la fonction $desc$ permet de définir les fonctions d'appartenances (simple et récursive), qui impliquent les relations qualitatives suivantes :

$$\forall e, e' \in \mathcal{E}, e \in_{rec} e' \Rightarrow (e \sqsubseteq e') \wedge (e' \sqsupseteq e).$$

2. Inverse des relations : toute relation qualitative binaire a un inverse (noté $\breve{\ }$) qui est aussi une relation qualitative :

$$\breve{\preceq} = \succeq, \breve{\ll} = \gg, \breve{\sqsubseteq} = \sqsupseteq, \breve{\neq} = \neq, \breve{\bowtie} = \bowtie, \breve{\rightrightarrows} = \rightrightarrows, \breve{\rightleftarrows} = \rightleftarrows.$$

   Pour les relations de $\mathcal{Q}^+_l$, cela signifie que

$$\forall e, e_1, e_2 \in \mathcal{E}, \forall t^+ \in \mathcal{Q}^+_l, (e_1\ t^+_e\ e_2) \Leftrightarrow (e_2\ \breve{t}^+_e\ e_1).$$

3. Orientation : la relation de même orientation est une relation d'équivalence, et la relation d'orientation opposée possède une propriété de transitivité en liaison avec la relation de même orientation, pour toutes les entités qui sont des intervalles. Les trois inférences utiles sont les suivantes :

$$\begin{aligned}
&\forall e_1, e_2, e_3 \in \mathcal{E},\\
&\quad (e_1 \rightrightarrows e_2) \wedge (e_2 \rightrightarrows e_3) \Rightarrow (e_1 \rightrightarrows e_3);\\
&\quad (e_1 \rightleftarrows e_2) \wedge (e_2 \rightleftarrows e_3) \Rightarrow (e_1 \rightrightarrows e_3);\\
&\quad (e_1 \rightrightarrows e_2) \wedge (e_2 \rightleftarrows e_3) \Rightarrow (e_1 \rightleftarrows e_3).
\end{aligned}$$

4. Génération d'orientations en liaison avec des relations qualitatives orientées :

$$\begin{aligned}&\forall e_1, e_2, e, e' \in \mathcal{E}, \forall t^+ \in \mathcal{Q}_l^+,\\ &\qquad (e_1\, t^+_e\, e_2) \wedge (e_1\, t^+_{e'}\, e_2) \Rightarrow (e \rightrightarrows e');\\ &\qquad (e_1\, t^+_e\, e_2) \wedge (e_1\, \breve{t}^+_{e'}\, e_2) \Rightarrow (e \rightleftarrows e').\end{aligned}$$

5. Équivalence des relations *ordre* : la relation *ordre* gène le bon fonctionnement des algorithmes car elle est la seule qui ne soit pas binaire ; on la remplace alors par un ensemble équivalent de relations *avant* en introduisant une entité fictive :

$$\{ordre(e_1, \ldots, e_n)\} \Leftrightarrow (e_1\; avant_{e_{fict}}\; e_2) \wedge (e_{n-1}\; avant_{e_{fict}}\; e_n).$$

6. Table de transitivité pour les relations qualitatives associées à des relations d'Allen (tableau 7.3) : cette table, si on lui adjoint les règles de passage des tableaux 7.4, 7.5 et 7.6, permet de traiter l'ensemble des relations sur les intervalles et les points. Néanmoins, il convient, pour les relations qualitatives de $\mathcal{Q}_l^+$ de disposer d'une entité constitutive commune pour appliquer la table.

7. Réseau de contraintes : à partir d'un réseau de contraintes qualitatives relatives à une même entité constitutive, les algorithmes de consistance de chemin, de satisfaisabilité ou de recherche du réseau minimal s'appliquent.

8. Inférence d'ordre :

   (a) À partir d'ordres : il s'agit de la propriété montrée à la figure 4.2 et qui s'écrit formellement ainsi :

$$\begin{aligned}&\forall e_1, \ldots, e_n, e'_1, \ldots, e'_m \in \mathcal{E},\\ &\qquad ordre(e_1, \ldots, e_n) \wedge ordre(e'_1, \ldots, e'_m) \wedge e_i = e'_k \wedge e_j = e'_l\; (i < j, k < l)\\ &\qquad \Rightarrow ordre(e_1, \ldots, e_{i-1}, e'_k, \ldots, e'_m) \wedge \ldots\end{aligned}$$

   (b) À partir des quatre équivalences du 4.2.2 [1] :

$$\begin{aligned}&\forall e_1, e_2, e, e' \in \mathcal{E},\\ &\qquad (e_1 \preceq_e e_2) \wedge (e_2 \preceq_{e'} e_3) \wedge (e \rightrightarrows e') \Rightarrow ordre(e_1, e_2, e_3);\\ &\qquad (e_1 \succeq_e e_2) \wedge (e_2 \succeq_{e'} e_3) \wedge (e \rightrightarrows e') \Rightarrow ordre(e_1, e_2, e_3).\end{aligned}$$

[1] On ne conserve que les inférences impliquant des ordres, les autres sont contenues dans la table de transitivité.

9. Ordonnancement des positions :
   (a) Grâce au tableau 4.4 en conjonction avec des relations d'orientation et des relations qualitatives ;
   (b) À partir de relations d'ordre :
   $\forall e_1, \ldots, e_n \in \mathcal{E}, ordre(e_1, \ldots, e_n) \Rightarrow ord(O_1, \ldots, O_n) \wedge ord(F_1, \ldots, F_n)$;
   (c) À partir de relations quantitatives pures : $\forall P_1, P_2, P_3, d_{12} + i_{12} \leq d_{13} - i_{13} \Rightarrow ord(P_1, P_2, P_3)$.
10. Algorithme de Floyd-Warshall : à partir d'un réseau de contraintes quantitatives sur lesquelles l'additivité est valide, cet algorithme en $O(n^3)$ permet de déterminer les intervalles minimaux pour les variables contraintes.
11. Traduction de relations qualitatives en relations quantitatives (Cf. §6.2.3).
12. Traduction de relations quantitatives en relations qualitatives (Cf. §6.2.3).

7.3.3 Application d'algorithmes de raisonnement temporel

Nous avons vu à la section 6.2 des algorithmes permettant de résoudre des problèmes classiques en représentation du temps. Comment peuvent-ils être utilisés ici, vu les caractéristiques de notre problème? Pour cela, il faut transformer le problème initial de manière à satisfaire les contraintes nécessaires à l'utilisation de ces algorithmes. Par exemple, la résolution d'un problème de contraintes quantitatives en utilisant l'algorithme proposé en représentation temporelle nécessite d'avoir des distances exprimées selon la même orientation (pour que les distances soient additives), et de se placer dans un point de vue unique. Si on met de côté pour le moment la question des incohérences, la grande différence va résider dans l'utilisation des relations de même orientation et d'orientation opposée.

Avant de décrire l'algorithme, nous allons en dresser une ébauche basée sur des transformations du problème, pour se ramener aux conditions des algorithmes de raisonnement temporel.

Relations qualitatives : D'abord, à partir de l'ensemble des entités $\mathcal{E}$, on ne considère que celles qui ont des composants, c'est-à-dire dont la fonction *desc* est non vide sur au moins un point de vue. Ceci permet de partitionner $\mathcal{E}$ en sous-ensembles, appelés *îlots*, pour lesquels une partie des entités possède la même orientation et une autre partie (disjointe) une orientation opposée (figure 7.3). Il s'agit bien d'une partition car, si une entité appartient à deux de ces ensembles, ils peuvent être fusionnés en un seul, puisque la relation de même orientation est une relation d'équivalence.

L'intérêt de ce traitement est qu'en projetant ces ensembles sur l'ensemble des relations qualitatives orientées, on obtient des ensembles de relations qu'on va pouvoir manipuler comme cela est fait en représentation du temps, car les orientations de leur entité de référence seront connues les unes par rapport aux autres. En effet, en construisant une entité fictive e_{fict} contenant toutes les entités d'un îlot $\mathcal{I}$ et d'orientation fixée (disons celle des entités du plus grand sous-îlot $\mathcal{I}^+$), on peut se ramener à des relations qualitatives orientées ayant pour seule et unique référence cette entité fictive, en appliquant l'équation 4.12 ;

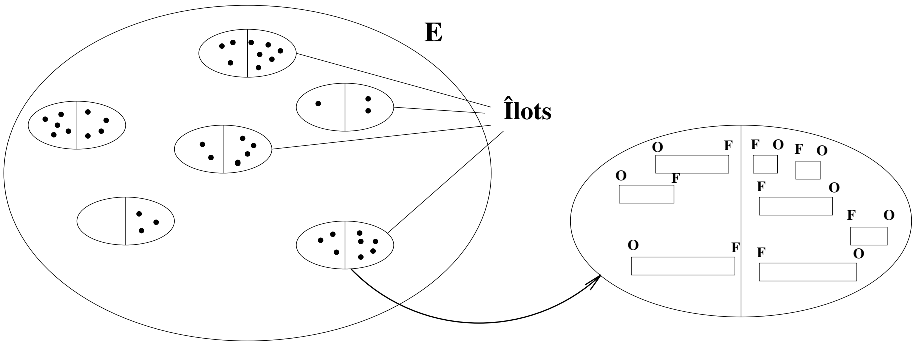


Figure 7.3 - : Partitionnement du sous-ensemble de $\mathcal{E}$ suivant la relation de même orientation. Comme l'information est incomplète, il subsiste des îlots indépendants dont l'orientation par rapport aux autres îlots n'est pas connue. Pour chaque îlot, pour chaque couple d'entités de cet îlot, il existe une relation de même orientation ou d'orientation opposée liant ces deux entités ; réciproquement, pour deux entités de deux îlots différents, il n'existe pas de relation d'orientation entre ces entités (car, s'il en existait une, les deux îlots pourraient fusionner). De plus, pour chaque couple d'entités d'un sous-îlot, la relation les liant est celle de même orientation.

en effet, soit $e \in \mathcal{I}^+$, la relation $e_1 \, t^+_e \, e_2$ implique $e_1 \, t^+_{e_{fict}} \, e_2$ puisque $e \rightrightarrows e_{fict}$; de même, pour $e \in \mathcal{I}^-$, il s'ensuit $e_1 \, \breve{t}^+_{e_{fict}} \, e_2 \Leftrightarrow e_2 \, t^+_{e_{fict}} \, e_1$ car $e \rightleftarrows e_{fict}$. On obtient au bout du compte une partition de R^+_{quali} basée sur la partition de $\mathcal{E}$. Alors, sur chacun de ces ensembles enrichi des relations qualitatives de R^-_{quali}, on peut appliquer des algorithmes de consistance de chemin, de satisfaisabilité ou même d'étiquetage minimum, indépendamment des autres (figure 7.4). Un avantage de ce procédé est qu'il détecte des inconsistances locales aux ensembles de relations, tout en autorisant la poursuite du raisonnement sur les autres ensembles. De plus, tous les développements (dont certains sont encore du domaine de la recherche) de ces algorithmes pourront être pris en compte dans le raisonnement, en particulier l'ajout et le retrait dynamiques de contraintes, le traitement de contraintes flexibles (i.e. non sûres), l'introduction de contraintes préférentielles, etc.

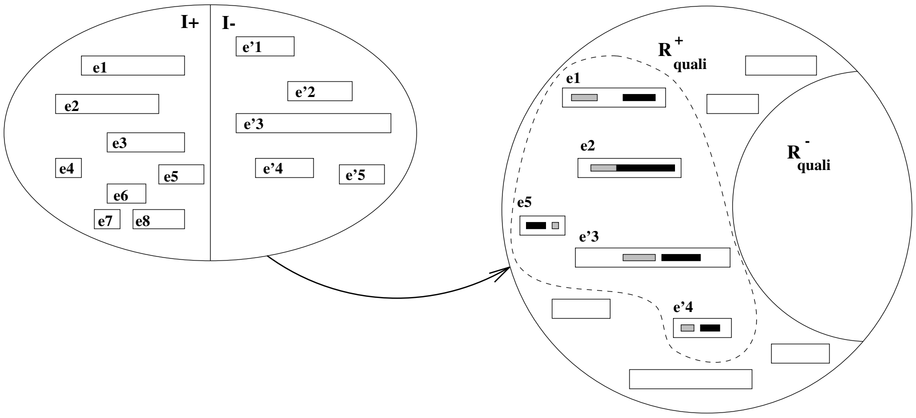


Figure 7.4 - : Projection des îlots d'orientation relative connue sur l'ensemble des relations qualitatives orientées. L'introduction d'une entité fictive englobant toutes les entités de référence met en évidence un ensemble de relations sur lequel appliquer un algorithme de chemin-consistance, de satisfaisabilité ou d'étiquetage minimum.

Les nouvelles relations inférées vont éventuellement permettre de fusionner des îlots grâce à la règle d'inférence 4 étendue à l'ensemble des relations qualitatives (Cf. tableau 7.11), car elle permet d'engendrer de nouvelles relations d'orientation. On peut alors répéter ce processus jusqu'à ce que la stabilité soit atteinte. La description de l'algorithme détaillera ce mécanisme.

Relations quantitatives : En ce qui concerne les relations quantitatives, en plus du problème d'orientation s'ajoute celui lié à la décomposition en points de vue. On ne peut *a priori* que travailler point de vue par point de vue, à l'exception de l'utilisation de la fonction *scale* quand elle existe entre deux unités liées à des points de vue, auquel cas toute relation quantitative nouvelle dans un point de vue est traduite immédiatement dans le point de vue associé. Il est donc indispensable avant toute chose de partionner R_{quanti} par point de vue. Ensuite, de même que pour les relations qualitatives ont été cherchés des îlots de même orientation, de même, le raisonnement sur les relations quantitatives nécessite qu'elles s'expriment selon une direction connue, pour pouvoir appliquer des règles d'additivité.

Pour ce faire, en partant des îlots précédemment mis en évidence, c'est-à-dire en ne considérant que les positions associées aux entités de l'îlot, plusieurs moyens sont à notre disposition pour connaître la disposition de deux positions P_i et P_j liées par la relation quantitative ($P_i \leftrightarrow P_j = [d_{ij}, i_{ij}]$) :

- utiliser le tableau 4.4, l'orientation des entités e_i et e_j associées à P_i et P_j, ainsi que la relation qualitative r qui les lie : en effet, dans le cas le plus compliqué où e_i et e_j sont différentes, l'énumération des éléments de r (qui est *a priori* une disjonction de relations atomiques) montre si P_i et P_j sont toujours disposées de la même façon par rapport à l'entité fictive de référence ;

- utiliser les relations ordre pour en extraire l'ordre des positions ;

- utiliser des informations quantitatives pures pour ordonner les positions ;

- utiliser la traduction des relations qualitatives en relations quantitatives, ce qui revient en fait à ordonner les extrémités puisque l'algorithme n'infère que des relations du style $x - y < 0$ (Cf. §6.2.3).

Une fois l'orientation des relations connue, il est possible d'appliquer le même algorithme que celui utilisé en représentation du temps (figure 7.5). Nous verrons dans la description de l'algorithme général (Cf. §7.3.4) le détail du fonctionnement de ces mécanismes.

Relations qualitatives et relations quantitatives vont pouvoir être aisément intégrées, car l'ordonnancement des positions est issu des entités pour lesquelles l'orientation relative de l'une par rapport à une autre est connue, puisqu'elles appartiennent au même îlot ; par conséquent, les positions utilisées sont les extrémités des entités qui ont été traitées précédemment.

Enfin, grâce aux nouvelles relations de $\mathcal{R}_{carto}$ inférées, il sera possible de déterminer de nouvelles relations d'orientation qui pourront entraîner la fusion d'îlots de $\mathcal{R}_{quali}$.

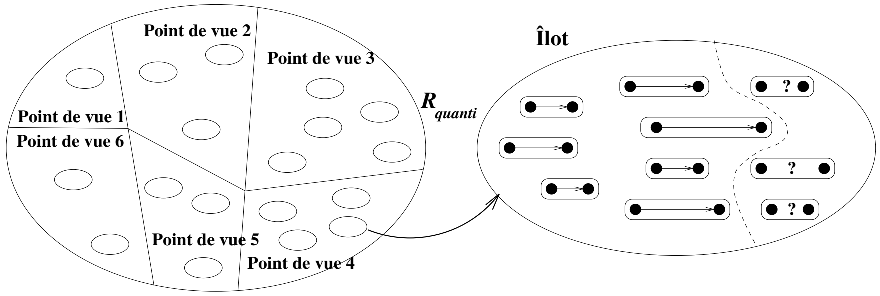


Figure 7.5 - : Détermination d'ensembles de relations quantitatives ≪additives≫. Après un partitionnement en points de vue, puis en îlots de $\mathcal{R}_{quanti}$, on utilise les inférences énoncées précédemment pour ne garder que les relations quantitatives qui satisfont la propriété d'additivité c'est-à-dire celles dont la position relative des points en relation est connue. Alors, chaque ensemble de relations qualitatives définit un réseau de contraintes quantitatives où les distances sont additives et sur lequel on peut appliquer l'algorithme de Floyd-Warshall.

Intégration des formalismes qualitatif et quantitatif : Nous allons détailler les deux transformations qui, à partir des relations qualitatives permettent d'inférer des relations quantitatives (malheureusement, sans certitude quant à la complétude de la transformation), et réciproquement partant des relations quantitatives vers les relations qualitatives (avec cette fois, une optimalité de la transformation, en ce sens que celle-ci détermine toutes les relations qualitatives).

De $\mathcal{R}_{quali}$ **à** $\mathcal{R}_{quanti}$ **:** au sein d'un îlot, comme cela a été dit précédemment, cette transformation va juste permettre d'ordonner deux positions par rapport à l'entité de référence.

De $\mathcal{R}_{quanti}$ **à** $\mathcal{R}_{quali}$ **:** l'algorithme de Kautz et Ladkin peut s'appliquer tel quel au sein d'un îlot pour lequel l'orientation des relations quantitatives en jeu est connue. Pour assurer cela, on exhibera une entité fictive e_{fict} telle que les relations quantitatives sont toutes exprimées dans le sens donné par cette entité, et qui sert de référence pour toute relation qualitative qui en nécessite une.

7.3.4 Description de l'algorithme

L'algorithme suivant a pour objectif de résoudre un problème de construction de cartes génomiques à partir des informations de base du problème, qui ont déjà été énumérées précédemment (Cf. §7.3.1). Étant donné que le problème est NP-complet, on a défini des versions plus lâches de l'algorithme basées sur la consistance de chemin et la satisfaisabilité, pour finir par la détermination d'un réseau minimal. Il reste alors à savoir s'il est complet et à caractériser les résultats obtenus ; en particulier, le passage qualitatif-quantitatif n'est pas systématiquement optimal, et on ne peut donc qu'affirmer des consistances locales aux réseaux considérés au fur et à mesure.

Le corps de l'algorithme est présenté ci-après (algorithme 4) ; la figure 7.6 permet d'en avoir une vision plus globale. Le détail de son fonctionnement apparaît dans les descriptions des procédures qu'il utilise (algorithmes 5, 6, 7, 8, 9).

Algorithme 4 *Construction-cartes*

Entrées :
 un ensemble d'entités $\mathcal{E}$;
 un ensemble de positions Pos issues de $\mathcal{E}$;
 un ensemble de relations qualitatives $\mathcal{R}_{quali}$;
 un ensemble de relations quantitatives $\mathcal{R}_{quanti}$.
Sorties :
 un ensemble de relations qualitatives impliquées par les informations précédentes ;
 un ensemble de relations quantitatives impliquées par les informations précédentes ;
 des ensembles les plus grands d'entités ordonnées.

Initialisations :
 Extraction des entités composites
  Rendre $\mathcal{E}_{comp}$
 Inférence de relations d'appartenance
 Inférence d'orientations
 Remplacement des relations ordre
Partitionnement en îlots de $\mathcal{E}_{comp}$ par l'orientation :
 Rendre $\{I\} = \{I^+\} \cup \{I^-\}$.
Répéter
 Répéter
  Pour *chaque îlot I,* **faire**
   Projection sur les relations qualitatives
    Rendre $\mathcal{R}^+(I) = \mathcal{R}^+(I^+) \cup \mathcal{R}^-(I^-)$.
    Rendre $\mathcal{E}(I)$.
   Résolution du CSP.
  Fin Pour
  Fusion des îlots
 Jusqu'à ce que *plus de fusion.*
 Traitement des relations quantitatives
  Projection sur chacun des points de vue (dont l'unité est additive)
   Pour *chaque relation* $(P_i \leftrightarrow_p P_j = [d_{ij}, i_{ij}])$ *de* $\mathcal{R}_{quanti}$, **faire**
    $\mathcal{R}_{quanti}(p) = \mathcal{R}_{quanti}(p) \cup (P_i \leftrightarrow_p P_j)$
   Fin Pour
  Recherche des orientations des relations quantitatives
   Rendre $\mathcal{R}_{quanti}(p)(I)$.
  Résolution du CSP de relations quantitatives.
  Inférence de relations qualitatives
Jusqu'à *stabilité*
Génération des ordres (éventuellement)

Tableau 7.7 - : Algorithme de construction de cartes. Les fonctions encadrées sont décrites en détail dans les tableaux suivants.

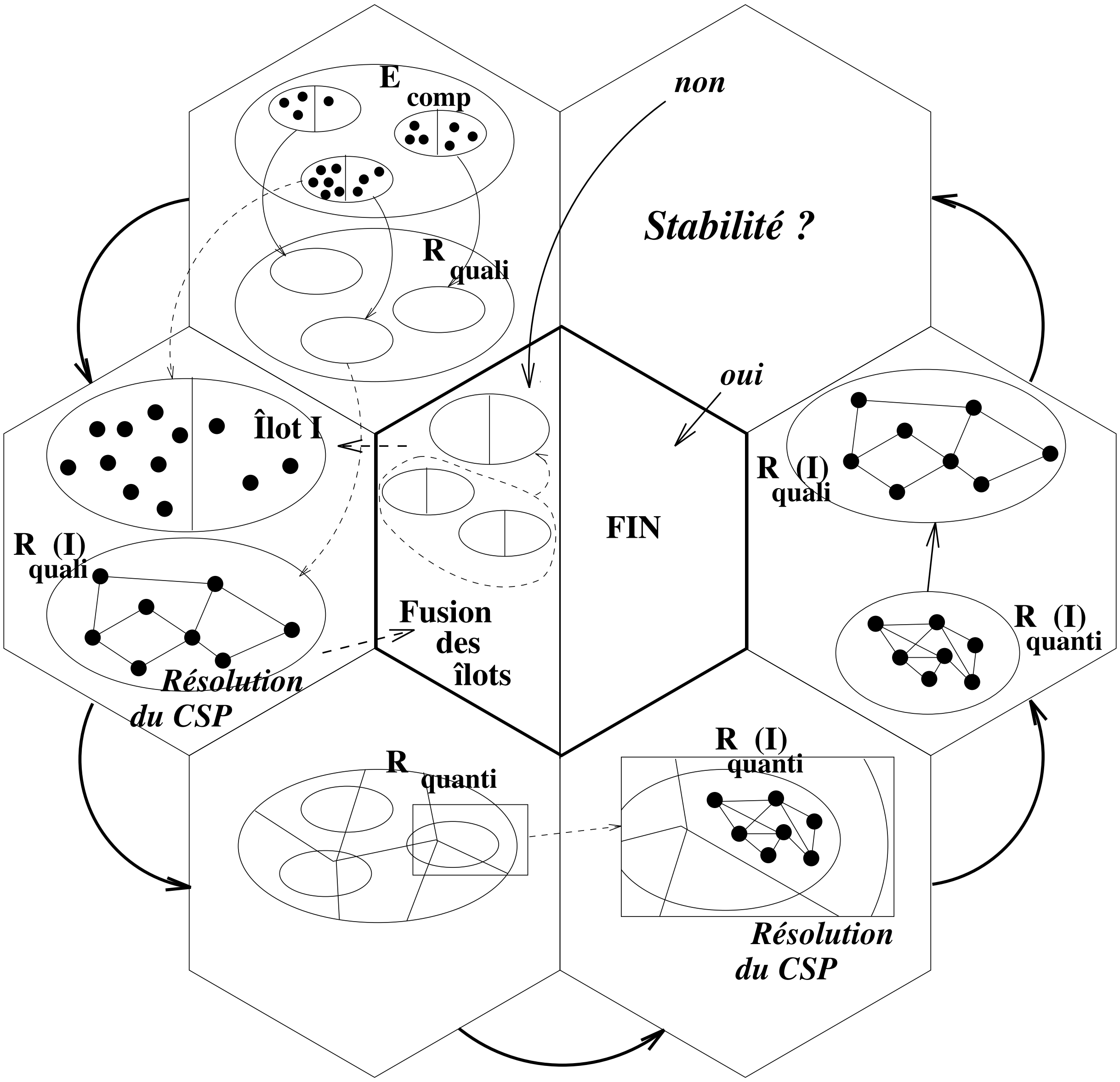


Figure 7.6 - : Schéma de l'algorithme de construction de cartes. Après le partitionnement de $\mathcal{E}_{comp}$ en îlots et leur projection sur les relations qualitatives, chaque îlot applique un algorithme de résolution du CSP associé. Les nouvelles relations inférées permettent éventuellement la fusion d'îlots. Puis, le traitement des relations quantitatives a lieu, encore au sein des îlots. Le partitionnement de celles-ci par point de vue et la recherche de leurs orientations permettent d'appliquer un algorithme de résolution du CSP de contraintes quantitatives. Les nouvelles relations quantitatives servent alors à inférer des relations qualitatives. Si plus aucune relation, tant qualitative que qualitative, n'a été découverte, l'algorithme s'arrête, sinon il boucle sur les diverses résolutions, jusqu'à la stabilité.

Algorithme 5 *Initialisations*

Entrées :

un ensemble d'entités $\mathcal{E}$;

un ensemble de relations qualitatives $\mathcal{R}_{quali}$;

la fonction $desc$.

Sorties :
  un ensemble d'entités composites $\mathcal{E}_{comp}$;
  modification de $\mathcal{R}_{quali}$.

Remplacement des relations ordre :
  Pour *chaque relation* $ordre(e_1, \ldots, e_n)$, **faire**
    Créer une entité fictive e_{fict}
    $\mathcal{R}^-_{quali} := \mathcal{R}^-_{quali} \backslash ordre(e_1, \ldots, e_n)$
    $\mathcal{R}^+_{quali} := \mathcal{R}^+_{quali} \cup (e_1 \; avant_{e_{fict}} \; e_2) \cup \ldots \cup (e_{n-1} \; avant_{e_{fict}} \; e_n)$
  Fin Pour
Extraction des entités composites :
  Pour *tout élément* e *de* $\mathcal{E}$, **faire**
    Si $\exists p \in \mathcal{P}, desc(e,p) \neq \emptyset$
      alors $\mathcal{E}_{comp} := \mathcal{E}_{comp} \cup \{e\}$.
    Fin Si
  Fin Pour
Inférence de relations d'appartenance :
  Pour *tout élément* e *de* $\mathcal{E}_{comp}$, **faire**
    Pour *tout point de vue* p *de* $\mathcal{P}$, **faire**
      Pour *tout élément* e' *de* $desc(e,p)$, **faire**
        $\mathcal{R}^-_{quali} := \mathcal{R}^-_{quali} \cup (e \sqsupseteq e')$.
      Fin Pour
    Fin Pour
  Fin Pour
Inférence d'orientations :
  Pour *tout couple* $(e_1 \, t^+_e \; e_2), (e_1 \, t^+_{e'} \; e_2)$ *de* $\mathcal{R}^+_{quali}$, **faire**
    $\mathcal{R}^-_{quali} := \mathcal{R}^-_{quali} \cup (e \rightrightarrows e')$;
  Fin Pour
  Pour *tout couple* $(e_1 \, t^+_e \; e_2), (e_1 \, \breve{t}^+_{e'} \; e_2)$ *de* $\mathcal{R}^+_{quali}$, **faire**
    $\mathcal{R}^-_{quali} := \mathcal{R}^-_{quali} \cup (e \rightleftarrows e')$.
  Fin Pour

Tableau 7.8 - : Initialisation de l'algorithme de construction de cartes. Elle permet tout particulièrement le partitionnement en îlots, qui définit des résolutions locales à chacun des îlots.

Algorithme 6 *Partitionnement en îlots*

Entrées : un ensemble d'entités composites $\mathcal{E}_{comp}$.
Sorties : une partition de $\mathcal{E}_{comp}$ en sous-ensembles $I = I^+ \cup I^-$ tels que l'orientation des relations entre les entités de I soient connues.

Tant que $\mathcal{E}_{comp} \neq \emptyset$, **faire**
  $e \in \mathcal{E}_{comp}$;
  $I(e) = I^+(e) \cup I^-(e) := \{e\} \cup \emptyset$;

$\mathcal{E}_{comp} := \mathcal{E}_{comp} \backslash \{e\}$;
Tant que $I^+(e) \cup I^-(e) \neq \emptyset$, **faire**
$e_1 \in I^+(e)$;
$I^+(e) := I^+(e) \backslash \{e_1\}$;
Pour *chaque relation* $(e_1 \rightrightarrows e_2)$ *de* $\mathcal{R}^-_{quali}$, **faire**
$I^+(e) := I^+(e) \cup \{e_2\}$;
$\mathcal{E}_{comp} := \mathcal{E}_{comp} \backslash \{e_2\}$;
continuer récursivement sur e_2 ;
Fin Pour
Pour *chaque relation* $(e_1 \rightleftarrows e_2)$ *de* $\mathcal{R}^-_{quali}$, **faire**
$I^-(e) := I^-(e) \cup \{e_2\}$;
$\mathcal{E}_{comp} := \mathcal{E}_{comp} \backslash \{e_2\}$;
continuer récursivement sur e_2 ;
Fin Pour
idem pour $e_1 \in I^-(e)$;
Fin Tant que
Fin Pour
Rendre $\{I\}$.

Tableau 7.9 - : Partitionnement des entités. Cette étape peut mettre en évidence des incohérences si deux entités se retrouvent être à la fois de même orientation et d'orientation opposée. L'algorithme élimine alors de la résolution l'ensemble des entités concernées.

Algorithme 7 *Projection sur les relations qualitatives*

Entrées :
un ensemble d'entités composites d'orientation relative connue $I = I^+ \cup I^-$;
un ensemble de relations qualitatives orientées $\mathcal{R}_{quali}$.
Sorties : un sous-ensemble de $\mathcal{R}_{quali}$ lié a l'îlot I.

$\mathcal{R}^+_{quali}(I) := \mathcal{R}^+_{quali}(I^+) \cup \mathcal{R}^+_{quali}(I^-) := \emptyset$;
$\mathcal{E}(I) = \emptyset$;
Pour *chaque élément* $(e_1\, t_e\, e_2)$ *de* $\mathcal{R}_{quali}$, **faire**
Si e *est dans* I^+, **alors** $\mathcal{R}^+_{quali}(I^+) := \mathcal{R}^+_{quali}(I^+) \cup \{(e_1\, t_{e_{fict}}\, e_2)\}$;
Sinon
Si e *est dans* I^-, **alors** $\mathcal{R}^+_{quali}(I^-) := \mathcal{R}^+_{quali}(I^-) \cup \{(e_2\, t_{e_{fict}}\, e_1)\}$;
Fin Si
$\mathcal{E}(I) = \mathcal{E}(I) \cup \{e_1\} \cup \{e_2\} \cup \{e\}$;
Fin Si
Fin Pour
Rendre $\mathcal{R}^+_{quali}(I) = \mathcal{R}^+_{quali}(I^+) \cup \mathcal{R}^+_{quali}(I^-)$ *et* $\mathcal{E}(I)$.

Tableau 7.10 - : Le partitionnement des entités constitutives induit un partitionnement des relations qualitatives orientées, puisque toute relation de $\mathcal{R}^+_{quali}$ est orientée par une entité de $\mathcal{E}_{comp}$.

La fusion des îlots est un processus plus complexe qu'il n'y parait ; la règle d'inférence 4 n'est pas l'unique moyen à même de la réaliser. En effet, il n'est pas indispensable que les relations soient exactement identiques pour pouvoir inférer que les entités constitutives ont même orientation. De plus, étant donné qu'au fur et à mesure du raisonnement, les relations s'enrichissent et qu'il apparaît des disjonctions de relations cartographiques atomiques, il est indispensable de traiter la fusion dans sa généralité. Le problème s'énonce très simplement de la manière suivante : quand est-il possible de décider de l'orientation relative de deux entités constitutives e et e' sachant les deux relations suivantes $(e_1\, t_e\, e_2)$ et $(e_1\, t'_{e'}\, e_2)$ (t et t' sont des disjonctions de relations de $\mathcal{R}^+_{quali}$) ? La seule façon de traiter tous les cas est d'être systématique ; donnons juste un exemple permettant de mieux comprendre comment la fusion fonctionne :

$$\text{Soient } (e_1\{\preceq, \ll\}_e e_2), (e_1\{\preceq\}_{e'} e_2) \in \mathcal{R}^+_{quali}, \text{ on a } (e_1\{\preceq\}_e e_2) \wedge (e \rightrightarrows e').$$

La relation $\ll_e$ disparaît car elle est incompatible avec la relation liée à e' ; de plus e et e' ont la même orientation[2]. Le tableau 7.11 fait la synthèse des inférences possibles (ce tableau est similaire à celui décrit dans [Lee et al.93] sur la fusion de LOF – Cf. §8.4).

$(e_1\, t'_{e'}\, e_2)$ / $(e_1\, t_e\, e_2)$	$\preceq$	$\ll$	$\succeq$	$\gg$	$\{\preceq, \ll\}$	$\{\succeq, \gg\}$	$\{\preceq, \gg\}$	$\{\succeq, \ll\}$
$\preceq$	$F(e \rightrightarrows e')$	C	$F(e \rightleftarrows e')$	C	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$
$\ll$	C	$F(e \rightrightarrows e')$	C	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$
$\succeq$	$F(e \rightleftarrows e')$	C	$F(e \rightrightarrows e')$	C	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$
$\gg$	C	$F(e \rightleftarrows e')$	C	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$
$\{\preceq, \ll\}$	$F(e \rightrightarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	?	?
$\{\succeq, \gg\}$	$F(e \rightleftarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	?	?
$\{\preceq, \gg\}$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	?	?	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$
$\{\succeq, \ll\}$	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightrightarrows e')$	$F(e \rightleftarrows e')$	?	?	$F(e \rightleftarrows e')$	$F(e \rightrightarrows e')$

Tableau 7.11 - : Fusion des îlots. Cette table indique quand la fusion de deux îlots est possible en fonction des relations entre deux entités identiques e_1 et e_2 relativement à deux entités constitutives différentes e et e'. Quand il y a fusion (lettre F), l'orientation relative des entités e et e' est indiquée ; la lettre C indique un conflit, le point d'interrogation une incertitude.

Si une relation qualitative d'un îlot contient une relation $\sqsubseteq$ ou $\sqsupseteq$, mais que son vis-à-vis dans un autre îlot, n'en a pas, on peut l'enlever de la disjonction et appliquer les résultats donnés par le tableau précédent ; par contre, si chaque relation contient l'une ou l'autre, il n'est pas possible d'appliquer la table puisque ces deux relations atomiques ne déterminent aucune orientation.

La procédure **fusion** s'écrit de la manière suivante (algorithme 8).

[2] Ces conséquences sont valides car les entités e_1 et e_2 ont la même dimension, comme cela a été précisé lors de la formalisation (Cf. §7.1.1).

Algorithme 8 *Fusion des îlots*

Entrées :
un ensemble d'îlots ;
un ensemble de relations qualitatives liées aux îlots.
Sorties : un nouvel ensemble d'îlots et de relations qualitatives associées.

Pour *chaque îlot I et chaque îlot $I' \neq I$*, **faire**
Pour *chaque couple de relations* $(e_1 t_e e_2), (e_1 t'_{e'} e_2) \in \mathcal{R}_{quali}(I) \times \mathcal{R}_{quali}(I')$, **faire**
Si $\sqsubseteq \in t \wedge \sqsubseteq \notin t'$, **alors** $t := t \setminus \sqsubseteq$ **Fin Si**
Si $\sqsupseteq \in t \wedge \sqsupseteq \notin t'$, **alors** $t := t \setminus \sqsupseteq$ **Fin Si**
idem dans l'autre sens
Si $t = \emptyset \vee t' = \emptyset$, **alors** *Conflit : exit* **Fin Si**
Selon *la valeur de la table*, **faire**
C : Conflit :exit
F :
$I_{fus} := I \cup I'$;
$e_{fict}(I_{fus}) := e_{fict}(I)$;
$\mathcal{R}_{quali}(I_{fus}) := \mathcal{R}_{quali}(I)$;
Si $e_{fict}(I) \rightrightarrows e_{fict}(I')$, **alors** $\mathcal{R}_{quali}(I_{fus}) := \mathcal{R}_{quali}(I_{fus}) \cup \mathcal{R}_{quali}(I')$
Sinon
Pour *chaque relation* $(a\, r_{e'}\, b)$ *de* $\mathcal{R}_{quali}(I')$, **faire** $\mathcal{R}_{quali}(I_{fus}) := \mathcal{R}_{quali}(I_{fus}) \cup (a\, r_{e'}^{-1}\, b)$
Fin Pour
Fin Si
Fin Selon
Fin Pour
Fin Pour

Tableau 7.12 - : Fusion d'îlots. Cet algorithme se base sur le tableau précédent ; la fusion entre deux îlots s'opère si deux relations dans chacun des îlots spécifient la même orientation.

Algorithme 9 *Orientation des relations quantitatives*

Entrées :
un ensemble de relations quantitatives $\mathcal{R}_{quanti}$;
un îlot I;
un ensemble de relations qualitatives $\mathcal{R}_{quali}(I)$ liées à I;
un ensemble $\mathcal{E}(I)$ d'entités liées à l'îlot.
Sorties :
un ensemble de relations quantitatives $\mathcal{R}_{quanti}(I)$ sur lesquelles l'additivité est valide;
un ensemble de positions ordonnées $P(I)$ dans l'îlot I.

$\mathcal{R}_{quanti}(I) := \emptyset$;
Pour *chaque relation quantitative* $(P_i \leftrightarrow P_j)$, **faire**
  Si $e_i = entit(P_i), e_j = entit(P_j) \in \mathcal{E}(I)$, **alors**
    Si $e_i = e_j$, **alors**
      Si $e_i \rightrightarrows e_{fict}$, **alors**
        Si $P_i = orig(e_i) \wedge P_j = fin(e_i)$, **alors**
          $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_i \leftrightarrow P_j)$
        Sinon $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_j \leftrightarrow P_i)$
        Fin Si
      Sinon Si $e_i \rightleftarrows e_{fict}$, **alors** l'inverse **Fin Si**
      Fin Si
    Sinon soit $e_i \, t_{e_{fict}} \, e_j \in \mathcal{R}_{quali}(I)$
      Si $e_i \rightrightarrows e_{fict} \wedge e_j \rightrightarrows e_{fict}$, **alors**
        Pour *chaque relation* $r \in t$, **faire**
          Récupérer l'ordre des positions P_i *et* P_j
          Si *l'ordre n'est pas conservé*, **alors** *passer à la relation quantitative suivante*
          Fin Si
        Fin Pour
        Si P_i *est toujours avant* P_j, **alors**
          $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_i \rightarrow P_j)$
          $P(I) := P(I) \cup \{(O_{fict}, P_i, P_j, F_{fict})\}$
        Sinon
          $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_j \rightarrow P_i)$
          $P(I) := P(I) \cup \{(O_{fict}, P_j, P_i, F_{fict})\}$
        Fin Si
      Sinon idem pour les autres cas
      Fin Si
    Fin Si
  Fin Si
Fin Pour
Pour *tout couple de relations* $(P_i \leftrightarrow P_j = [d_{ij}, i_{ij}]), (P_i \leftrightarrow P_k = [d_{ik}, i_{ik}])$, **faire**
  Si $d_{ij} + i_{ij} \leq d_{ik} - i_{ik}$, **alors** $P(I) := P(I) \cup \{(P_i, P_j, P_k)\}$
  Fin Si
Fin Pour
Pour *toute relation quantitative* $(P_i \leftrightarrow P_j)$, **faire**
  Si *il existe un chemin liant* O_{fict}, P_i, P_j *et* F_{fict}, **alors**
    $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_i \leftrightarrow P_j)$;
  Sinon
    Si *il existe un chemin liant* O_{fict}, P_j, P_i *et* F_{fict}, **alors**
      $\mathcal{R}_{quanti}(I) := \mathcal{R}_{quanti}(I) \cup (P_j \leftrightarrow P_i)$;
    Fin Si
  Fin Si
Fin Pour
Rendre $\mathcal{R}_{quanti}(I)$;

Rendre $P(I)$.

Tableau 7.13 - : Orientation de relations quantitatives. Deux mécanismes sont utilisés, le premier est équivalent à une traduction des relations qualitatives, le second se fonde uniquement sur les relations quantitatives.

Avant de conclure cette partie avec l'implémentation de l'algorithme, il est intéressant de faire quelques remarques sur l'algorithme de construction de cartes lui-même.

- La construction de $\mathcal{E}_{comp}$ permet de ne garder que des intervalles, et se débarrasse donc de tous les problèmes qui auraient pu apparaître concernant l'orientation de points.

- Les incohérences sont traitées localement de telle façon qu'il soit possible de continuer le raisonnement sur des parties indépendantes, grâce à la décomposition en îlots ; néanmoins, il est clair que ces incohérences doivent se résoudre par une intervention d'un utilisateur, à même de sélectionner les relations adéquates qui lèvent l'incohérence.

- Cet algorithme n'est pas complet, loin s'en faut ; même si on résout les CSP complètement, certaines procédures sont par essence incomplètes, car il n'est pas certain qu'elles utilisent toutes les inférences possibles. Sa complexité est polynomiale si les résolutions des CSP le sont (c'est-à-dire si on ne calcule pas le réseau minimal).

7.4 Implémentation

Deux aspects interviennent dans l'implémentation de l'algorithme : la résolution des CSP, qualitatif ou quantitatif, et tout le reste, propre au problème des cartes génomiques, c'est-à-dire la phase d'initialisation, le partitionnement en îlots, la procédure de fusion des îlots, le partitionnement en points de vue.

7.4.1 Résolution des CSP

Pour résoudre les CSP et faire l'intégration des contraintes qualitatives et quantitatives, nous avons récupéré le logiciel MATS[3] (pour *Metric/Allen Time System*) développé par Henry Kautz ; MATS implémente la résolution de contraintes qualitatives temporelles basées sur le formalisme d'Allen, celle des réseaux d'inégalités entre extrémités des intervalles et l'intégration des deux, à partir des algorithmes décrits au chapitre 6 et issus de [Kautz et al.91].

L'implémentation du formalisme des cartes génomiques dans le système de représentation de connaissances TROPES (Cf. chapitre 11) nous a obligés à utiliser comme langage

[3]MATS est disponible par ftp anonyme sur le site research.att.com dans le répertoire /dist/ai/ (fichier mats.shar.Z).

de programmation Le-Lisp. MATS ayant été écrit en Common Lisp, il a d'abord fallu le traduire en Le-Lisp. Ceci fait, il a pu être intégré quasiment tel quel dans l'algorithme global de construction de cartes. Une traduction de l'expression des relations en TROPES dans le formalisme de représentation de MATS a été réalisée.

Sans entrer dans les détails de l'implémentation de MATS, précisons simplement que le système fournit des fonctions de :

- réduction du réseau de contraintes qualitatives grâce à l'algorithme de consistance de chemin (cet algorithme ne permet donc pas d'obtenir le réseau minimal) ;
- transcription des contraintes qualitatives au réseau de contraintes quantitatives ;
- réduction du réseau de contraintes quantitatives à l'aide de l'algorithme de Floyd-Warshall ;
- transcription des contraintes quantitatives au réseau de contraintes qualitatives.

Ces opérations se poursuivent jusqu'à ce que plus aucun changement n'ait lieu. Des fonctions permettent alors de récupérer les informations sur chacun des réseaux, de manière globale ou localement pour certains intervalles ou certaines extrémités.

Nous avons introduit les relations cartographiques dans le modèle de représentation de MATS ; néanmoins, dans l'état actuel des choses, le système ne fait que les traduire dans le formalisme d'Allen et résout le CSP dans ce formalisme sans utiliser la table de transitivité propre aux relations cartographiques (tableau 7.3, page 77). Le résultat final, en ce qui concerne les relations qualitatives, n'est donc pas une disjonction de relations cartographiques, mais une disjonction des treize relations d'Allen.

7.4.2 Partie propre aux cartes

Cette partie est encore en cours d'implémentation. Toute l'initialisation ainsi que tous les algorithmes qui s'appuient sur les relations qualitatives ont été implémentés (tableaux 5, 6, 7, 8) ; il reste désormais à traiter les relations quantitatives (tableau 9) et l'intégration des deux.

Le point de départ de l'algorithme est la base de connaissances qui contient les descriptions de toutes les relations sur les éléments de carte. À partir de là, l'application des procédures d'initialisation et de partitionnement en îlots conduit à la création d'instances d'une classe *_îlot_*, contenant toutes les informations propres aux îlots ; les attributs de cette classe sont les suivants :

- *composites* contient les entités composites de l'îlot sous la forme d'un couple de deux ensembles dont les entités partagent la même orientation ;
- *quali* recouvre l'ensemble des relations qualitatives dont la référence est une entité de l'attribut *composites* ;
- *enti-quali* contient les entités mises en jeu dans les relations qualitatives précédentes (dans le but de regrouper les relations quantitatives de même sens) ;

- *quanti* est l'ensemble des relations quantitatives liées à l'îlot ;
- *inconsistant* est un booléen spécifiant si la résolution d'un CSP sur cet îlot a entraîné une inconsistance (ceci permet de les éliminer de la procédure de fusion).

La suite des traitements se partitionne suivant les îlots, jusqu'à la procédure de fusion.

Les tests effectués jusqu'à présent se sont bien sûr limités aux parties implémentées, et ne se basent pas sur des cas réels, mais sur un exemple similaire à celui présenté au chapitre 5.

Deuxième partie

Cartes génomiques et représentation de connaissances

Chapitre 8

Logiciels de représentation et de raisonnement cartographiques

Dans ce chapitre, nous présenterons en détail différents travaux, dont certains ont conduit à la réalisation de logiciels ayant pour but la représentation et la construction de cartes génomiques, que celles-ci soient spécifiques à un type de carte ou se situent dans un cadre plus général. En fait, ces deux aspects sont rarement traités conjointement, les logiciels de construction de cartes (*genome map assembly*, en anglais) s'appuyant peu souvent sur une représentation avancée, sans même aller jusqu'à parler de formalisme ; inversement, les logiciels permettant de représenter les cartes et les entités cartographiques n'incluent pas de raisonnement destiné à leur construction, et considèrent le plus souvent la carte comme un consensus.

Les sections suivantes introduisent les recherches effectuées dans ce domaine ; elles s'appuient sur des techniques informatiques nombreuses, allant de l'algorithmique pure à la représentation de connaissances dans un modèle à objets [Dorkeld94], en passant par l'utilisation de règles [Letovsky et al.92], la formalisation à l'aide de réseaux sémantiques [Graves93], la programmation par objets [Lee et al.93, Honda et al.93] ou l'utilisation de contraintes [Clark et al.94]. Je parlerai en dernier du logiciel HoverMaps[1], qui utilise le système de représentation de connaissances Shirka [Rechenmann et al.91], dont TROPES, le logiciel que nous utiliserons, et qui sera examiné en détail dans le chapitre suivant, se rapproche beaucoup.

8.1 CPROP : règles

Le logiciel CPROP [Letovsky et al.92] a pour vocation d'aider un généticien à inférer de nouvelles connaissances sur l'ordre et les distances entre loci à partir de celles issues d'expériences. Pour cela, à partir d'une représentation des informations qualitatives et quantitatives sur les entités cartographiques, il utilise des règles d'inférence décrites ci-après pour découvrir de nouvelles informations impliquées par les précédentes.

Au niveau de la représentation elle-même, CPROP fait une simplification importante,

[1]HoverMaps s'appelait anciennement MULTIMAP, mais le nombre important d'utilisations de ce nom a conduit à ce changement.

propre à la construction de cartes génétiques, à savoir que les entités positionnées sur la carte sont des marqueurs sans dimension. Aucune gestion d'intervalles n'est faite, mais ceci est justifié dans le cadre des cartes génétiques. Les seules assertions possibles sont soit l'expression d'une distance, éventuellement entachée d'une incertitude, entre deux marqueurs, soit la définition d'un ordre local de deux marqueurs relativement à une référence. Les auteurs introduisent en effet le concept de LOW (pour *Local Ordering Window*) ; ce concept sera repris par Lee *et al.* [Lee et al.93]. Il consiste à regrouper des entités dont l'ordre est connu, au sein du LOW. Chaque LOW définit donc un ordre partiel sur un ensemble de marqueurs, grâce à des assertions du type *A avant B* dans le LOW_i. Toute la connaissance expérimentale est traduite en expressions permettant d'y inclure l'ordre et la distance au sein d'un LOW ; une telle expression est de la forme : [locusA, locusB, LOW, BorneInf, BorneSup], dont le sens est : le locus A est avant le locus B dans le cadre LOW, et la distance séparant ces deux locus est comprise entre BorneInf et BorneSup. Il est intéressant de noter qu'il faut au moins deux assertions impliquant le même LOW pour avoir une information pertinente. Deux assertions [A, B, i,?,?] et [B, C, i,?,?] ont pour signification que B est compris entre A et C dans le LOW i. L'incertitude sur l'orientation des entités est donc contenue dans la définition de plusieurs LOW.

À partir de là sont définies des règles d'inférence qui, en se déclenchant, vont créer de nouvelles assertions. Les auteurs présentent ce mécanisme comme une propagation de contraintes. Ces règles portent à la fois sur les informations d'ordre et de distance ; elles sont décrites à la figure 8.1.

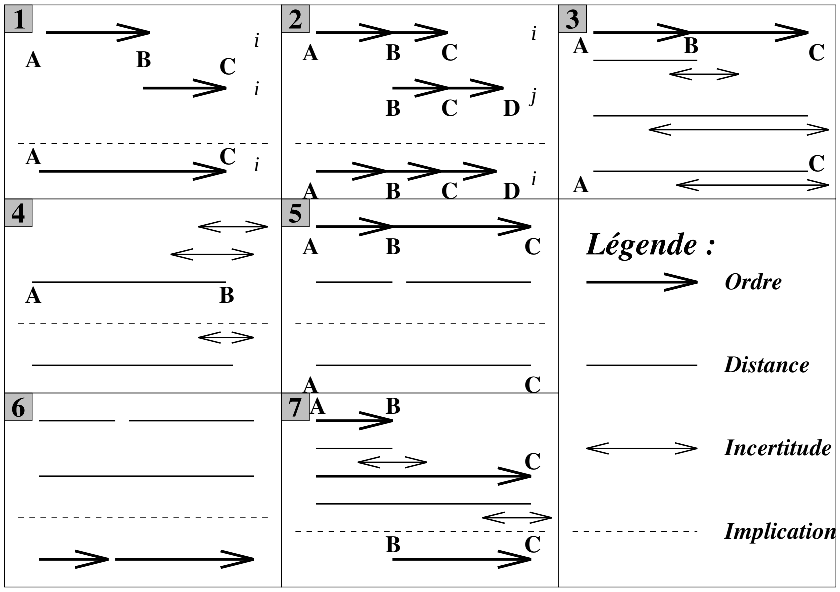


Figure 8.1 - : Les règles d'inférence de CPROP. La première règle est la transitivité ; la seconde permet de fusionner deux LOW (il existe une seconde version de cette règle quand les ordres dans chaque LOW sont inverses l'un de l'autre) ; la règle 3 restreint l'incertitude sur la distance AC pour un groupe ordonné de trois marqueurs A, B et C, car la borne inférieure de cette distance ne peut être avant celle de la distance AB (il en est de même pour la borne supérieure) ; la règle 4 calcule l'intersection des incertitudes sur les distances parmi toutes ; la cinquième règle énonce l'additivité des distances, connaissant l'ordre des marqueurs ; la règle suivante est l'inverse de cette additivité, et infère un ordre à partir d'informations de distance ; enfin, la dernière règle permet elle aussi de déterminer un ordre à partir de distances.

L'algorithme utilisé pour la construction de la carte génétique applique toutes les règles énoncées jusqu'à ce que plus aucune ne soit applicable. Comme le problème général est NP-complet, et que l'algorithme est polynomial, cela signifie que l'algorithme n'est pas complet et que l'ensemble des règles ne recouvre pas l'ensemble des inférences possibles à partir des données. En particulier, la fusion des LOW se limite à examiner des consistances deux à deux alors que d'autres d'ordre plus grand pourraient être examinées.

Une caractéristique du système est la détection d'inconsistances dans le jeu de données initiales ; ainsi, si les données de départ sont inconsistantes, CPROP va relever une contradiction, qui peut être de trois types :

- Contradiction de distance : une distance entre deux marqueurs n'a pas de recouvrement avec la distance actuelle ;
- Contradiction d'ordre : l'ordre de deux marqueurs dans un LOW est en contradiction avec celui existant déjà ;
- Contradiction sur un ordre inféré : toutes les distances entre trois marqueurs sont connues, mais aucun ne peut se trouver au milieu.

Dans les deux premiers cas, deux assertions sont en contradiction, dans le troisième, il s'agit de trois assertions. Quelle que soit la situation, le système se rend compte de la contradiction et donne un arbre de dépendance des assertions en cause, jusqu'à remonter aux assertions des données initiales. C'est à l'utilisateur alors de déterminer quelles sont les assertions à éliminer ou modifier.

Une supposition forte du système est la validité de l'additivité des distances, d'autant plus qu'il s'agit ici de construire des cartes génétiques. Selon les auteurs, cette supposition est acceptable dans la mesure où on peut affaiblir les contraintes métriques. C'est pourquoi dans les tests effectués, les contraintes métriques provenant d'expériences de co-transduction ont été affectées d'une incertitude de 10%. Même ainsi, l'algorithme a relevé des contradictions qui ont nécessité l'affaiblissement d'un certain nombre d'assertions.

Il est intéressant de remarquer qu'un tel fonctionnement peut être intégré sans aucune difficulté dans la formalisation décrite dans la première partie, simplement en affirmant que la carte génétique est additive, et en affectant les incertitudes à une valeur particulière.

8.2 WEAVE : réseaux sémantiques

Mark Graves [Graves93] a développé un système d'aide à la construction de bases de connaissances appelé WEAVE, qu'il a utilisé pour définir des structures de données complexes pour la modélisation des informations de distance et d'ordre dans les cartes génomiques. Sa formalisation a également pour but de réaliser une intégration des différents types de carte, en l'occurrence les cartes génétique, physique et d'hybridation radioactive ; ce dernier type de carte est obtenu en cassant l'ADN en morceaux à l'aide de rayons X. Une étude statistique permet alors d'estimer des distances (exprimées en Rays) entre marqueurs, cette distance ayant la particularité intéressante d'être directement proportionnelle à la distance physique.

La représentation de la connaissance se fait à travers l'utilisation de constructeurs de données, sortes de mini réseaux sémantiques, qui s'agrègent pour exprimer des relations de distance ou d'ordre. Ainsi, le constructeur de distance est un nœud qui pointe vers les informations nécessaires, à savoir deux marqueurs et une estimation de la distance ; le constructeur de l'estimation est de même constitué d'un nœud d'où partent des arcs orientés vers la valeur de l'estimation, son unité, sa provenance, etc. L'expression d'une distance se fait par l'assemblage d'instanciations de ces différents constructeurs (figure 8.2). Les entités sont typées et la définition d'un constructeur de distance est une fonction du produit cartésien des deux types *Marqueur* et du type *Estimation* vers le type *Distance*.

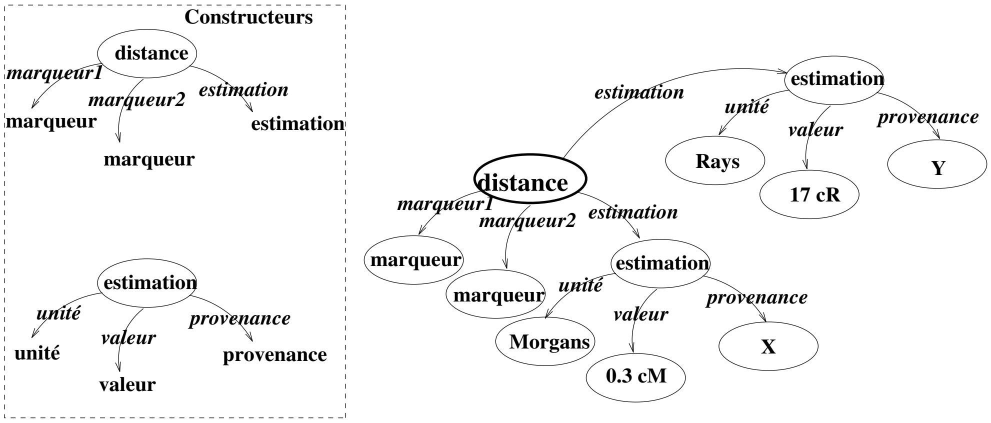


Figure 8.2 - : Les constructeurs de données et leur assemblage pour former une distance (d'après [Graves93]). Le constructeur de distance crée une entité de type *Distance* à partir de deux marqueurs et d'une ou plusieurs estimations de cette distance. L'utilisation conjointe de différents constructeurs permet de représenter une distance, valide sur plusieurs cartes, grâce à la possibilité d'avoir plusieurs estimations.

La représentation de l'ordre entre marqueurs est faite en utilisant le même formalisme à base de constructeurs. L'intégration des différentes cartes est également possible ; de plus, il est nécessaire de pouvoir représenter des incertitudes concernant l'ordre, soit que des marqueurs ne puissent pas être distingués les uns des autres, soit que l'ordre soit partiel, soit qu'il existe plusieurs ordres contradictoires. La figure 8.3 montre la représentation sous forme de réseau sémantique de la relation ordre.

Dans l'état actuel de ces travaux, aucun algorithme particulier n'a été développé au sein de la base de connaissances pour permettre l'inférence de nouvelles relations ou mettre en évidence les contradictions entre plusieurs ordres. Mais la seule représentation des informations de distance et d'ordre est intéressante car il est peu fréquent de trouver des formalismes qui autorisent effectivement l'intégration de connaissances sur les différentes cartes, comme cela est fait ici.

8.3 GeMM : programmation par objets

Le but de l'approche présentée dans [Honda et al.93] est de définir des abstractions des concepts manipulés lors de l'étude du génome du point de vue cartographique. L'utilisation d'un langage de programmation par objets est motivée principalement par les

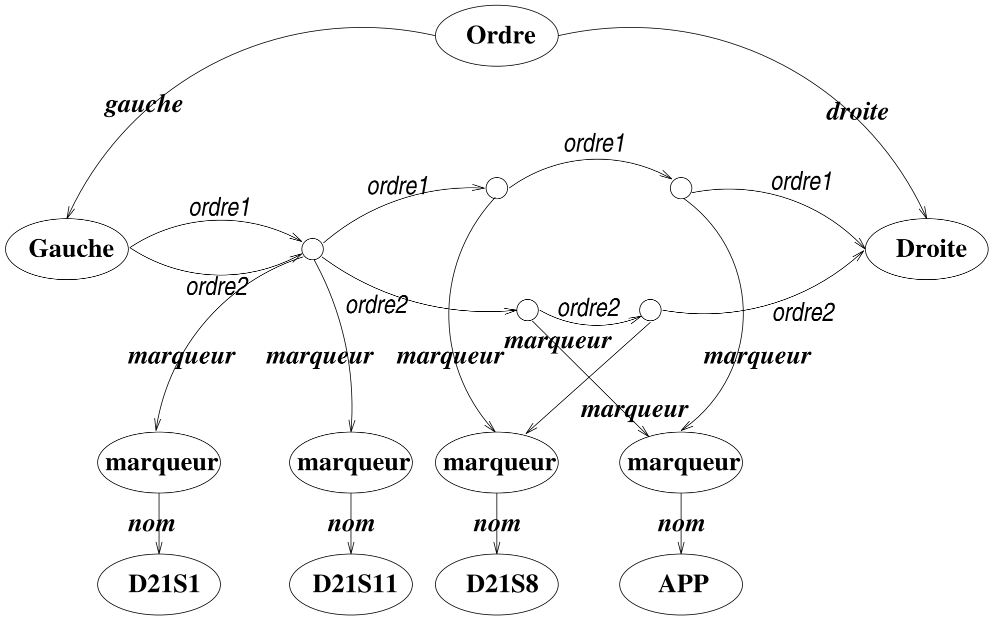


Figure 8.3 - : Représentation de la relation d'ordre. Les marqueurs sont ordonnés de la gauche vers la droite ; deux ordres différents contradictoires sont visibles, et les deux marqueurs D21S1 et D21S11 n'ont pas pu être séparés expérimentalement.

possibilités d'abstraction des données en termes d'objets et l'existence de l'héritage (qui permet essentiellement de réduire la tâche d'expression des connaissances dans le langage de programmation).

L'objectif que se sont fixés les auteurs est la représentation de l'information génomique à tous les niveaux de résolution, c'est-à-dire une représentation de l'ensemble des cartes génomiques et des entités particulières qui les constituent. Ainsi, on retrouve pour la carte cytogénétique la représentation des bandes, celle de tous les marqueurs alléliques (RFLP, etc.) pour la carte génétique, et en ce qui concerne la carte physique la représentation de sites de restriction, des régions séquencées, etc. L'intérêt du développement d'un système permettant de représenter l'ensemble de l'information génomique réside dans son utilisation pour des projets de nature diverse, comme par exemple les cartographies physique et génétique, l'assemblage automatique des séquences d'ADN (projet de séquençage) et l'exploration de l'information codée dans l'ADN.

Cet objectif d'intégration des différentes cartes pose de nombreux problèmes liés à la comparaison de cartes issues d'expériences différentes, ayant des marqueurs différents, des systèmes de coordonnées différents, comme, en particulier, l'impossibilité de convertir les distances génétiques en distances physiques.

Le modèle à base d'objets va donc représenter les cartes et les entités de ces cartes, en essayant de satisfaire les requis suivants :

- représentation d'un ensemble de points de repère différents ;
- représentation de limites incertaines de ces entités, de relations d'ordre, de distance et de recouvrement ;
- représentation des mêmes entités sur des cartes différentes ;

- intégration d'information cartographique de basse et haute résolution ;
- représentation de cartes ayant des systèmes de coordonnées différents ;
- représentation de sous-régions de cartes.

L'entité de base du modèle est la carte, vue comme un ensemble d'items auquel est adjointe l'information nécessaire à l'ordonnancement de ces items. Chaque item peut être aussi simple qu'une annotation ou aussi complexe qu'une autre carte, autorisant de la sorte la définition d'une hiérarchie de cartes imbriquées les unes dans les autres. Une carte a donc une nature identique à celle de n'importe quel autre item. La hiérarchie de classes de la figure 8.4 montre comment sont représentées les entités nécessaires à la satisfaction des contraintes requises énumérées ci-haut.

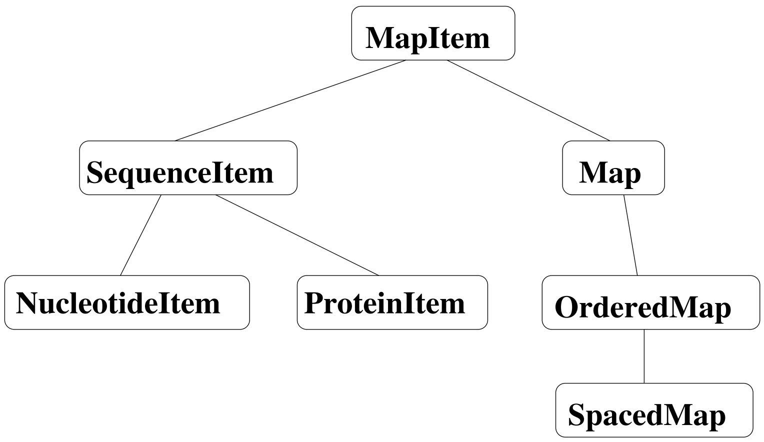


Figure 8.4 - : Hiérarchie des classes du modèle GeMM. MapItem regroupe toutes les entités susceptibles d'appartenir à une carte, comme par exemple une séquence de la classe SequenceItem. Une autre sous-classe de MapItem est la classe Map, qui référence un ensemble d'éléments de MapItem. L'information sur l'ordre de ces items est contenue dans la description de sous-classes de Map ; ainsi, la classe OrderedMap contient la connaissance sur l'ordre relatif des items sur la carte, de même que son orientation par rapport à la carte ; plus précise encore est la classe SpacedMap, dont une instance possède en plus toutes les informations de distances séparant les items ordonnés (par défaut, il y a additivité, mais ce comportement peut être redéfini dans d'autres sous-classes, comme GeneticMap).

L'implémentation actuelle n'implique le traitement que de cartes physiques, et le développement de méthodes destinées à la construction de contigs ; l'intégration des différents types de carte n'a donc pas été faite. De plus, aucun algorithme général n'est proposé pour résoudre les questions d'ordonnancement des entités des cartes, à partir des relations exprimables (ordre, distance, recouvrement).

8.4 Hiérarchie de relations

Même si le modèle développé par Lee *et al.* se limite à la représentation et au raisonnement de la carte physique dans le but de raccorder des contigs, leur démarche est intéressante à plus d'un titre, car ils s'attaquent aux mêmes problèmes que ceux exposés dans la partie sur la modélisation des cartes génomiques (Cf. première partie). En particulier, leur travail est un des rares qui incluent à la fois une représentation des entités du problème des cartes génomiques et un raisonnement sur les entités des cartes grâce à la description d'un ensemble de relations entre elles. Ici aussi est utilisé un langage de programmation par objets, dans le but de représenter une hiérarchie des entités en question, mais surtout une hiérarchie des relations qui seront amenées à être traitées par la suite.

Les entités représentées sont celles qui apparaissent lors de l'ordonnancement de contigs, c'est-à-dire des fragments d'ADN. Cette classe d'objets se raffine en fragments clonés, gènes, cosmides, etc. D'autres classes d'objets sont définies comme les sondes, pour lesquelles existe également une hiérarchie. Toutes ces entités sont considérées comme des intervalles, et les relations qu'il est possible d'exprimer entre elles sont limitées et organisées en une hiérarchie en haut de laquelle on trouve les relations les plus générales, jusqu'à aboutir, au niveau des feuilles, aux relations atomiques. Le but du raisonnement sur ces relations, qui sera présenté par la suite, est de spécialiser le plus possible les relations exprimées, grâce aux informations initiales et aux inférences. Les relations exprimables sont directement issues des expériences biologiques effectuées pour la construction de contigs, à savoir des techniques d'hybridation de sondes sur les fragments d'ADN. C'est pourquoi à toute relation est associée une justification donnée par un certain nombre de sondes qui ont permis d'établir cette relation. Par exemple, la relation de non-disjonction *non-disjoint-1(A, B)* est justifiée par la présence d'une même sonde localisée à la fois sur les entités A et B.

Les auteurs ont également été amenés à introduire le concept d'orientation, puisque, comme cela a déjà été dit, les relations qu'on veut exprimer ne sont valides que par rapport à une orientation particulière. Ils ont donc repris le concept de *Local Ordering Window* des auteurs de CPROP [Letovsky et al.92], l'appelant *Local Orientation Frame* ou *LOF*. Un LOF est un ensemble de relations qui ont la même orientation, indépendamment de celle (globale) du chromosome. Ainsi, pour toute relation dont on ne connaît pas l'orientation dans un LOF_i particulier, on crée la relation correspondant à l'orientation inverse, en intervertissant les entités en relation et en prenant l'«inverse» (dans un sens particulier, pour cette transformation) de la relation, pour la placer dans le LOF_{-i} d'orientation inverse (figure 8.5). Les auteurs présentent des règles de transcription des relations d'un LOF au LOF inverse.

Ce traitement de l'orientation fait que la relation la plus générale est la relation *not-after* puisqu'elle est doublée de la relation inverse. Il suffit donc de montrer la moitié de la hiérarchie des relations puisque l'autre est obtenue en appliquant les règles d'inversion dont on a parlé juste avant (figure 8.6). Les relations qui impliquent une égalité sur la position des extrémités des intervalles ont été enlevées car elles ne sont pas pertinentes dans ce contexte.

Quels sont alors les moyens de raisonner sur un ensemble de relations ? Un des buts est la fusion de LOF, de façon à disposer des plus grands ensembles d'entités complètement ordonnées. Si la fusion de deux LOF est simple dans le cas d'entités ponctuelles pour lesquelles il suffit de vérifier que la même relation entre ces deux entités se retrouve d'un LOF à l'autre (comme cela est fait par la règle 2 de CPROP, figure 8.1), le problème se complique quand on passe aux intervalles. Il est en effet possible de fusionner deux LOF à partir d'une relation dans chacun des LOF impliquant deux mêmes entités sans

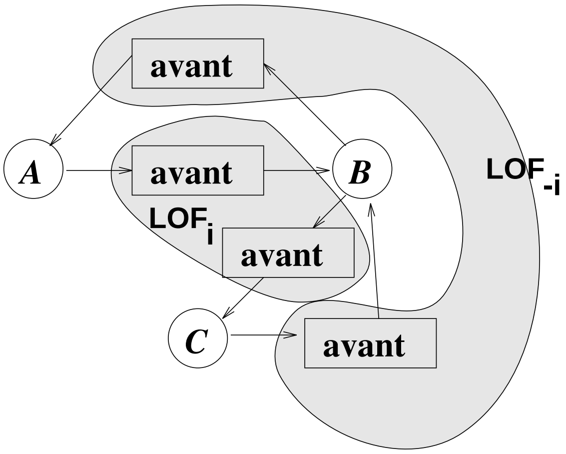


Figure 8.5 - : Description des LOF. La représentation de *(A avant B avant C)* ou *(C avant B avant A)* se fait en créant deux LOF d'orientation opposée, qui regroupent les différentes relations. Une relation d'ordre comme *avant* appartient donc à un LOF particulier et coexiste avec son image miroir dans un LOF d'orientation opposée.

que les relations soient identiques. De la même façon, on peut mettre en évidence des conflits entre deux relations n'appartenant pas au même LOF s'il n'existe aucune sous-classe commune aux deux relations (figure 8.7). Les auteurs décrivent une table donnant pour tout couple de relations, une dans chaque LOF, les conséquences qui en résultent, que ce soit une fusion des deux LOF, un conflit ou encore une spécialisation d'une des deux relations.

Un autre mécanisme de raisonnement est l'utilisation d'une table de transitivité qui permet d'inférer une nouvelle relation à partir de deux partageant une entité. La table de transitivité qui est donnée est valide à l'intérieur d'un même LOF. Chaque nouvelle relation inférée grâce à la table de transitivité bénéficie d'une justification à partir des sondes justifiant les deux relations utilisées.

Pour le moment, seul un algorithme simplifié a été développé, qui calcule le recouvrement de YAC pour créer un contig. Cet algorithme n'utilise qu'un certain nombre des relations définies précédemment et ne fait pas usage de la fusion des LOF. À partir des données d'hybridation de sondes avec les YAC, l'algorithme construit le contig en accord avec ces données et élimine les YAC inutiles car redondants. L'algorithme définitif calculera la fermeture transitive des relations au sein d'un LOF et fusionnera les LOF de même orientation, mais il reste à voir sa complexité et sa complétude vis-à-vis du problème de construction de cartes. De plus, les auteurs envisagent d'étendre le modèle pour qu'il puisse traiter des informations imprécises et contradictoires, un point soulevé par beaucoup mais peu traité de manière satisfaisante en raison de sa grande complexité.

D'autres recherches ont été effectuées qui se comparent à celle de Lee *et al.*, en particulier des travaux utilisant un formalisme basé sur la logique temporelle [Cui94, Hearne et al.94]. Il s'agit d'une adaptation de l'ensemble des relations d'Allen donnant lieu à une hiérarchie de spécialisation des relations entre intervalles. Si le formalisme est assez détaillé, les aspects algorithmiques n'ont malheureusement pas été traités ; or, c'est là que se situent les problèmes les plus difficiles.

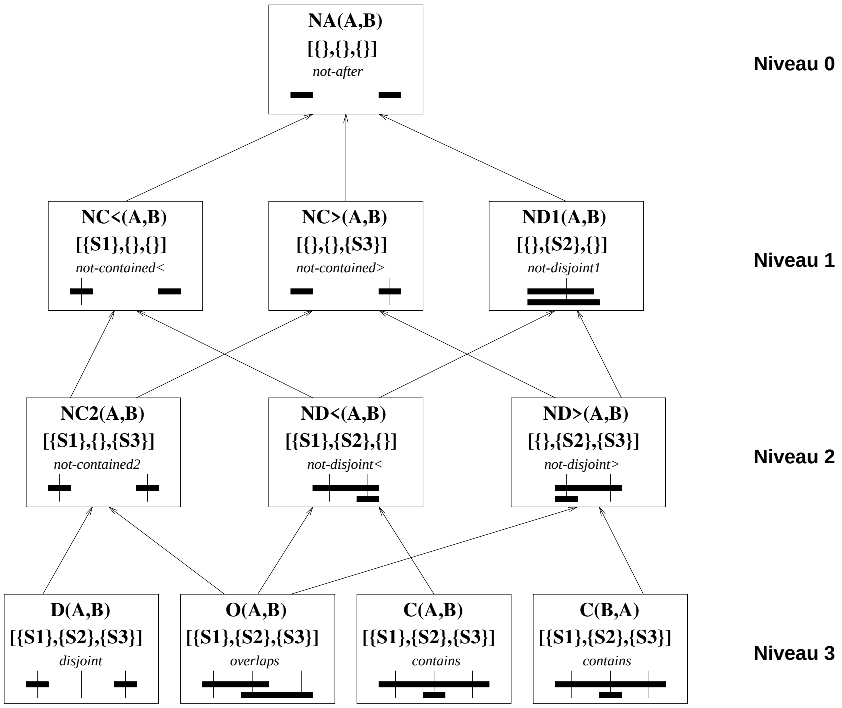


Figure 8.6 - : Hiérarchie des relations entre entités avec leur justification par sonde(s). La racine de l'arbre de spécialisation est la relation la plus générale, et par conséquent, aucune sonde ne la justifie. Au fur et à mesure que des sondes interviennent dans la justification d'une relation, celle-ci devient de plus en plus précise ; les feuilles de la hiérarchie sont des relations atomiques justifiées par trois sondes.

8.5 HoverMaps : représentation de connaissances

Le logiciel HoverMaps [Dorkeld94] est destiné à la représentation de données cartographiques concernant les mammifères. Cette représentation a été effectuée en utilisant un système de gestion de bases de connaissances à objets appelé Shirka [Rechenmann et al.91], dont nous parlerons dans les paragraphes qui suivent. Nous ne discuterons pas dans cette partie des raisons pour lesquelles est choisi un système de représentation de connaissances plutôt qu'un autre outil informatique pour la modélisation des cartes génomiques, nous y viendrons plus en détail au chapitre 9.

Shirka est un langage permettant de décrire des connaissances sous forme d'objets appelés *schémas*, dont la structure commune est la suivante : un schéma est caractérisé par son nom et possède un ensemble d'attributs typés. Les attributs sont précisés par un certain nombre de facettes, dont le but est d'en restreindre le type, par exemple en en limitant le domaine ou en spécifiant une contrainte à satisfaire. L'ensemble des schémas se sépare en deux sous-ensembles, les classes et les instances ; les classes sont des schémas de description génériques représentant un ensemble d'entités (ou *extension* de la classe), qui sont les instances du schéma de classe, et qui doivent correspondre à la description de la classe. La figure 8.8 montre un schéma de classe et un schéma d'instance.

Les classes sont organisées en une hiérarchie telle qu'une classe donnée domine d'autres classes plus spécifiques auxquelles elle transmet la connaissance sur les attributs ; chaque classe est ainsi une spécialisation de classe(s) de plus haut niveau (figure 8.9), et, par

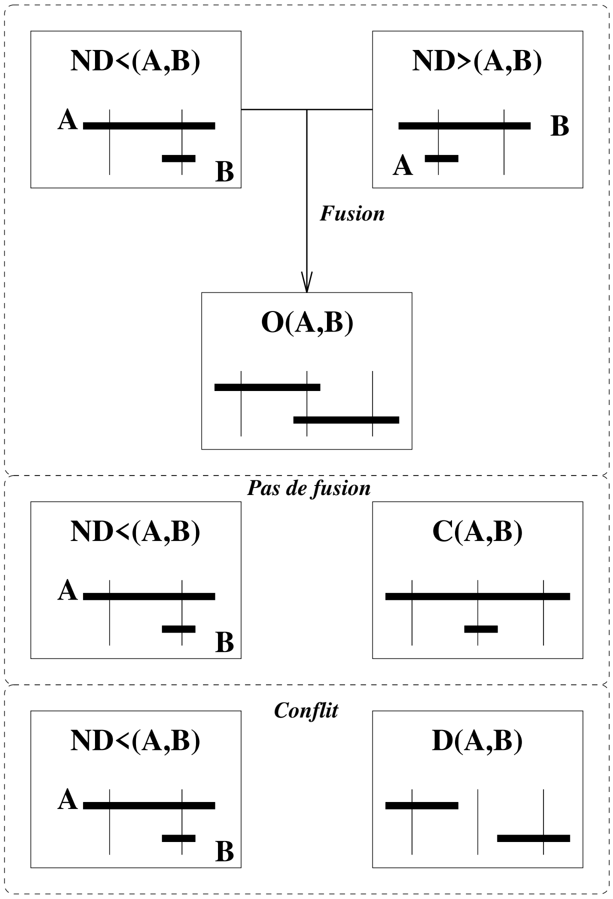


Figure 8.7 - : Exemples d'inférence à base de deux relations dans deux LOF différents. Le premier exemple implique une fusion des deux LOF et une restriction des deux relations en une seule dans le LOF obtenu ; le second ne permet pas de déduire la fusion des LOF, mais spécifie davantage la première relation $ND_<(A,B)$ en $C(A,B)$; enfin, le troisième met en évidence un conflit.

conséquent, au niveau des instances, la propriété suivante se trouve vérifiée : toute instance d'une classe C est également instance de C', sur-classe de C.

Les mécanismes d'inférence sont utilisés pour découvrir de nouvelles connaissances sur les entités de la base de connaissance (sachant que certaines instances sont incomplètes, puisque les attributs ne sont pas nécessairement tous valués). Le premier mécanisme a

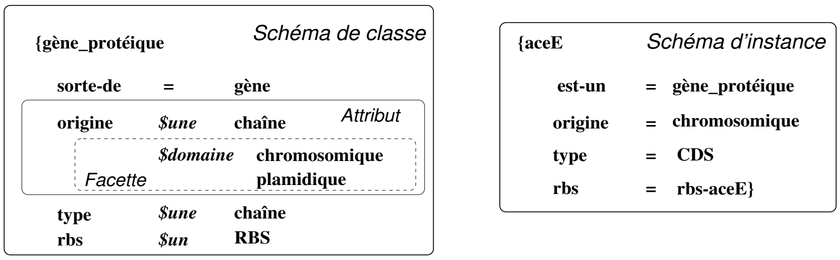


Figure 8.8 - : Schéma de classe et schéma d'instance dans Shirka. Chacun des schémas est caractérisé par un ensemble d'attributs ; dans le schéma de classe, les attributs sont précisés par des facettes de typage, tandis que le schéma d'instance leur donne des valeurs.

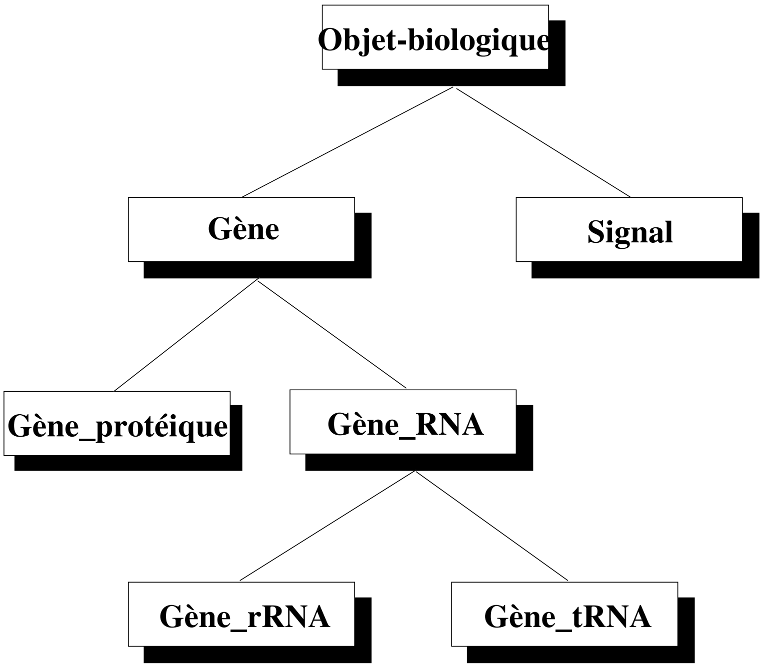


Figure 8.9 - : Hiérarchie de classes. Toute classe est la spécialisation d'une ou plusieurs classes plus générales, dont elle tire une partie de sa description, la complétant en rajoutant des attributs ou en raffinant ceux existants.

pur but de déterminer la valeur d'un attribut, si celui-ci n'en a pas déjà ; pour cela, Shirka permet de rattacher à un attribut une méthode permettant de calculer sa valeur ou un filtre qui sélectionne les instances adéquates de la base. Le second est la *classification*, qui augmente la connaissance disponible sur une instance en essayant de la descendre dans la hiérarchie de classes, vers des classes de plus en plus spécifiques.

La description des entités cartographiques qui est faite dans HoverMaps consiste, à l'instar de celle de GeMM [Honda et al.93], à définir des objets les modélisant. De plus, même si Shirka ne permet pas de modéliser des liens de composition de façon naturelle, ce lien a été construit artificiellement de manière à les distinguer des attributs classiques qui ont pour sémantique celle de propriétés ou caractéristiques de l'objet. Ainsi, un chromosome est *décrit* par des attributs indiquant l'organisme dont il est issu, ses longueurs génétique et physique, mais il *inclut* des bandes et des gènes (figure 8.10).

Le concept de carte est mis en évidence dans le système et possède une grande importance puisque la carte est l'entité générique qui regroupe des objets et définit un repère local, dont l'orientation peut ne pas être connue vis-à-vis de celle du chromosome. À partir de là peuvent être exprimées des informations sur l'ordre ou sur les distances entre entités, prises deux à deux ou définies par rapport à une origine.

Comme HoverMaps est destiné à être utilisé pour la comparaison d'informations cartographiques de différents mammifères, les cartes sont séparées par espèce ; il y a donc une hiérarchie de cartes de l'Homme, une pour les cartes de la souris, une pour celles du rat. Dans chacune de ces hiérarchies sont définies des classes plus spécifiques suivant le type de carte ou les informations qui les concernent (figure 8.11). Néanmoins, toute carte contient des informations sur le chromosome auquel elle appartient, une orientation

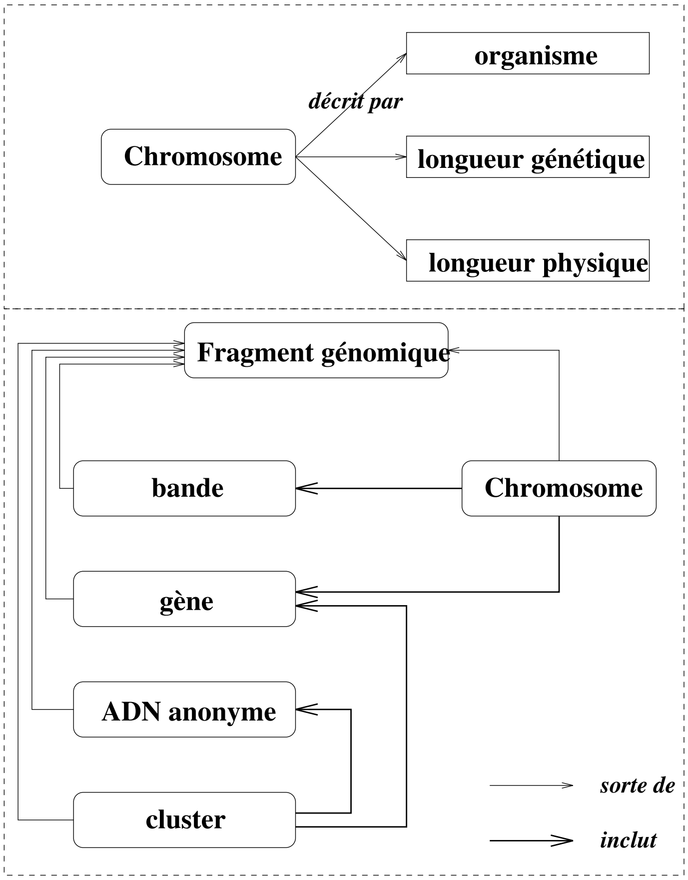


Figure 8.10 - : Expression de différents types de liens dans HoverMaps (d'après [Dorkeld94]). On distingue artificiellement le lien d'attribution et le lien d'inclusion ; un chromosome est décrit par les attributs *organisme*, *longueur génétique* et *longueur physique*, mais possède des composants qui sont les bandes, les gènes, etc.

éventuelle (si elle est connue) et ses éléments.

Chaque élément de carte est aussi une instance de la base de connaissances ; la classe à laquelle il appartient est déterminée par l'espèce, le type de carte auquel appartient cet élément et le type de l'élément. De cette manière (un peu lourde) sont représentées les structures des cartes indiquant par exemple que la carte physique est constituée de marqueur, d'exons, de sous-cartes (figure 8.12).

Tout élément de carte peut contenir des informations concernant sa position sur la carte qu'il référence. Par exemple, la classe d'éléments de carte cytogénétique possède un attribut *pos-cyto* qui positionne cet élément dans une bande cytogénétique particulière (ou un groupe de bandes).

Au niveau algorithmique, voire simplement à celui de la définition des relations qu'il est possible d'exprimer entre éléments de carte, un vide subsiste. Mais la représentation elle-même est la plus complète rencontrée jusqu'à présent même si on remarque des redondances et des difficultés de représentation liées à la pauvreté des liens entre objets de Shirka, limités à l'attribution.

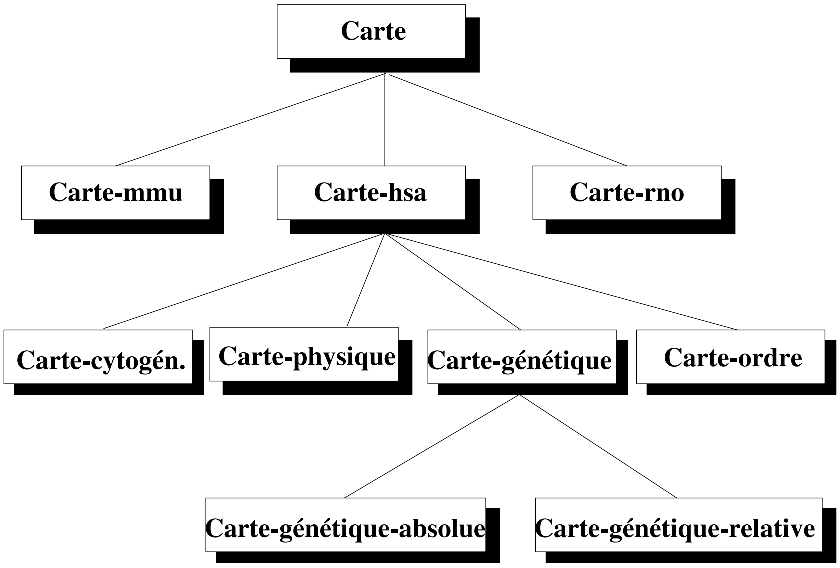


Figure 8.11 - : Hiérarchie des cartes. Chaque espèce définit une sous-hiérarchie similaire, mettant en évidence les différents types de carte (cytogénétique, génétique, physique) et les informations sur ces cartes : cartes d'ordre regroupant des éléments dont l'ordre relatif est connu, cartes absolues contenant des informations de distance par rapport à une origine, cartes relatives contenant des informations de distance inter-éléments. Seule la hiérarchie pour l'Homme (*carte-hsa* pour *Homo Sapiens*) est montrée ici.

8.6 Discussion

Le tableau récapitulatif suivant (tableau 8.1) résume de manière synthétique – et simplifiée – les différentes caractéristiques des systèmes présentés précédemment. Il est intéressant de les comparer avec la formalisation et l'algorithmique que nous avons développées.

Référence	CPROP [Letovsky et al.92]	WEAVE [Graves93]	GeMM [Honda et al.93]	[Lee et al.93]	HoverMaps [Dorkeld94]
Représentation					
cartes	génétique	toutes	toutes	physique	toutes
entités	points	toutes	toutes	intervalles	toutes
relations	ordre, distance	ordre, distance	ordre, distance, recouvrement	sous-ens d'Allen	ordre, distance
incertitude	Oui	Oui	Oui	Non	Non
orientation	LOW	Non	Sous-carte	LOF	Sous-carte
Raisonnement					
moyen	règles	$\emptyset$	$\emptyset$	règles	$\emptyset$
algorithme	$O(n^5)$	$\emptyset$	$\emptyset$	simplifié	$\emptyset$

Tableau 8.1 - : Récapitulatif des caractéristiques des différents systèmes de représentation et de raisonnement cartographiques.

Avant de parler des systèmes de représentation de connaissances, et plus particuliè-

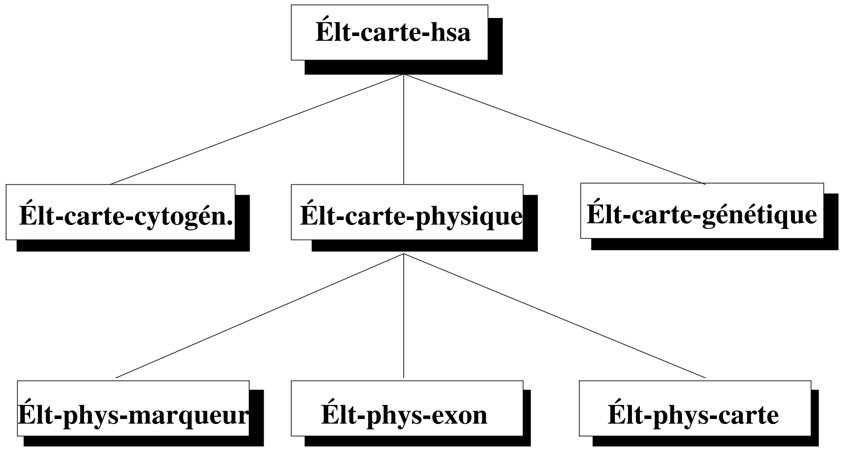


Figure 8.12 - : Description de la hiérarchie des éléments de carte. Celle-ci spécifie la composition des différents types de carte. Par exemple, la carte physique chez l'Homme contient des marqueurs, des exons, des sous-cartes, etc. Cette représentation a malgré tout l'inconvénient d'être hautement redondante, puisque la plupart des informations pourraient être partagées par l'ensemble des espèces.

rement de celui que nous utiliserons, TROPES, essayons de justifier les choix qui ont été faits auparavant par rapport aux systèmes existants.

Au niveau de la représentation, si on veut être générique, il est indispensable de pouvoir traiter l'ensemble des cartes génomiques. Le logiciel CPROP et celui de Lee *et al.* traitent respectivement les cartes génétiques et physiques, et donc ne permettent pas cette généricité, même si le second logiciel parait être plus facilement adaptable aux autres cartes, car la représentation qui est faite est plus souple. L'ensemble des relations exprimables est lui aussi important, car là encore, se restreindre peut nuire à la généralité. La plupart des systèmes se limitent aux relations d'ordre et de distance, car ce sont celles qui se retrouvent (et qui sont souhaitées) à la fin du raisonnement et que les biologistes préfèrent ; une carte consensus est effectivement donnée par l'ordre des entités de la carte et leur distance à une origine ou entre elles. Néanmoins, d'autres relations apparaissent lors d'expériences biologiques, et pouvoir les exprimer telles quelles est un avantage indéniable. Le problème se pose alors de sélectionner les relations pertinentes, vis-à-vis d'un problème particulier ou en restant dans la généralité. L'approche de Lee *et al.* est particulièrement intéressante, car, tout en limitant les relations à une dizaine, elle conserve suffisamment de généralité pour introduire sans trop de travail d'autres relations, simplement en étendant la hiérarchie. De même, notre formalisation autorise l'extension des relations jusqu'à incorporer toutes celles d'Allen si cela s'avère nécessaire, en changeant juste la définition de $\mathcal{R}_{carto}$. Cette facilité détermine la généricité de l'algorithme, dont les mécanismes n'ont pas à être modifiés pour continuer le raisonnement sur l'ensemble étendu des relations.

En effet, au niveau du raisonnement, il suffit pour chaque nouvelle relation de rajouter les inférences qui lui sont liées (que ce soit par des règles, une table de transitivité ou un autre mécanisme). Le choix de se raccorder aux relations d'Allen et à un formalisme de contraintes vient de l'énorme avantage qu'il apporte et que n'apportent pas les algorithmes spécifiques tels que celui de CPROP ou Lee, à savoir la possibilité de bénéficier de tous les progrès dans ce domaine et dont, manifestement, les logiciels de cartographie ont besoin. Ces progrès consistent particulièrement en la définition de contraintes flexibles, c'est-à-dire à la fois non sûres et pouvant être relaxées dynamiquement en cas d'inconsistance, point qui a été mis en exergue par tous les concepteurs des systèmes, mais qui n'est jamais traité de façon satisfaisante. L'algorithme présenté à la section 7.3 n'est pas plus satisfaisant que les autres à ce niveau-là, puisque sa gestion de l'inconsistance est très pauvre, mais il sera à même de profiter quasi-immédiatement des avancées dans ce domaine, sur lequel de nombreuses recherches sont effectuées et qui est en pleine évolution.

Chapitre 9

Les systèmes de représentation de connaissances

Même si la logique a apporté la première le moyen de représenter la connaissance et de raisonner dessus, d'autres formalismes ont vu le jour, dans un but de structuration de la connaissance et de flexibilité du raisonnement. Ainsi, les réseaux sémantiques ont fait leur apparition, sous forme de graphes dont les nœuds sont les entités à représenter et les arcs les relations entre ces entités ; nous en avons vu un exemple quand nous avons présenté le système WEAVE (section 8.2). Certaines relations ont une sémantique particulière, en particulier le lien *est-un* qui lie un nœud représentant un individu à un nœud représentant un concept générique (comme dans la phrase «l'hémoglobine est une protéine»), et le lien *sorte-de* qui lie un concept à un concept plus général (comme dans «un ARN de transfert est une sorte d'ARN»).

L'introduction des *frames* a fourni une structuration de la représentation en définissant un cadre sémantique [Minsky75], permettant de représenter une situation ou une connaissance typique. Un frame possède un certain nombre d'*attributs* dont la valeur décrit des caractéristiques du frame. Les frames sont reliés les uns aux autres par des relations similaires à celles des réseaux sémantiques. Ces idées ont été reprises dans de nombreux systèmes, qui n'ont conservé que les deux liens mentionnés plus haut, les autres liens étant modélisés par les attributs, donnant lieu à deux approches, à base de prototypes ou fondée sur la distinction entre frame générique et frame individuel. L'approche prototypique considère chaque frame comme un modèle du concept représenté. Un tel frame peut néanmoins engendrer des copies de lui-même, modifiées pour représenter des entités particulières. Les sous-frames obtenus héritent les caractéristiques qu'ils ne redéfinissent pas. Un inconvénient majeur de ce mécanisme [Brachman83] est que les attributs ne permettent jamais de décider de l'appartenance d'un frame à un concept, puisque ceux-ci peuvent constamment être remis en cause. L'approche classe-instance, que nous allons développer ici, est souvent utilisée dans les langages construits sur la notion d'objet.

9.1 L'approche classe-instance

L'approche classe-instance fait une distinction entre les entités (frames) qui sont des descriptions génériques de catégorie ou de famille et celles qui représentent des individus particuliers, et qui sont liés par le lien *est-un* aux entités de la première sorte. Cette séparation apparaît d'abord dans les langages de programmation par objets (dont elle est issue), mais aussi dans les langages terminologiques et les langages de représentation de connaissances par objets.

9.1.1 Les langages de programmation par objets

Il est important de faire la distinction entre ces langages et les langages de représentation par objets (section 9.2). Une classe d'un tel langage est une structure de données abstraite regroupant en son sein une description et un comportement communs à un ensemble d'objets, instances de cette classe. Une instance reflète donc la description de sa classe et se comporte comme une boîte noire dont le comportement est dicté par l'activation de méthodes définies au niveau de la classe. C'est l'envoi d'un message à l'instance qui déclenche la méthode correspondante.

Les classes bénéficient d'une facilité d'écriture grâce à l'existence d'une hiérarchie de spécialisation. Une sous-classe hérite de sa (ses) sur-classe(s) la description et les méthodes qui y sont définies.

Ces particularités ne font néanmoins pas de ces langages des langages de représentation, car ils ne fournissent qu'un cadre procédural et ne font pas la distinction entre la représentation de la connaissance et son utilisation [Patel-Schneider91]. De plus, leurs capacités d'expression sont limitées et certains concepts ne sont pas descriptibles. Enfin, une instance est liée de manière définitive à une classe, qui n'a donc qu'un rôle de modèle de données, sans pouvoir être utilisée comme un mécanisme d'inférence.

9.1.2 Les langages terminologiques

Le précurseur des langages terminologiques, KL-ONE [Brachman79], repose sur la notion de *concept* : un concept est décrit par des termes qui sont des formules logiques impliquant d'autres concepts et des *rôles* liés par des constructeurs, qui permettent par exemple d'effectuer des opérations ensemblistes sur les concepts, de restreindre le nombre de valeurs d'un rôle, etc. (tableau 9.1).

Un concept est soit *générique* s'il décrit un ensemble d'individus, soit *spécifique* s'il n'en désigne qu'un. Un concept générique est dit *primitif* quand sa description ne donne que des conditions nécessaires d'appartenance ; ce sont en général des concepts dont la définition est imprécise ou difficile à donner (comme par exemple le concept de *Personne*). Un concept *défini* exprime lui des conditions nécessaires et suffisantes d'appartenance.

Les concepts sont organisés en une hiérarchie par la relation d'ordre appelée *subsomption* et dont les interprétations sont les suivantes [Woods91] : un concept A subsume un concept B si :

- l'ensemble des individus dénotés par A contient l'ensemble des individus dénotés par B (interprétation extensionnelle) ;

- le concept A est plus général que le concept B vis-à-vis d'un critère formel appliqué aux descriptions des concepts (interprétation intentionnelle ou structurelle) ;

```
(def-prim-concept personne)
(def-prim-concept mâle)
(def-concept homme
             (and personne mâle))
(def-prim-rôle enfant)
(def-concept père
             (and
                  (some
                        (restr enfant personne))))
(def-concept père-heureux
             (and père
                  (at-least 3 enfant)))
```

Tableau 9.1 - : Expression de concepts dans les langages terminologiques. *Personne* et *mâle* sont des concepts primitifs, tandis que *homme*, *père* et *père-heureux* sont des concepts définis dont la description constitue un ensemble de conditions nécessaires et suffisantes. *Enfant* est un rôle primitif utilisé dans la définition de *père*.

- il existe des règles déductives permettant d'inférer que le concept A subsume le concept B (interprétation logique).

Woods définit deux autres interprétations liées à la définition explicite de liens de subsomptions entre les concepts.

La hiérarchie de subsomption est déterminée automatiquement par un classifieur à partir de la description en termes des concepts . De nombreux travaux ont porté sur la complexité et la complétude des algorithmes classificatoires, les extrêmes allant d'une part d'un algorithme complet et de complexité polynomiale obtenu en réduisant l'expression du langage à d'autre part des algorithmes soit incomplets, soit exponentiels, mais avec un langage plus riche [Brachman et al.91].

Certains langages terminologiques ont ajouté à la description des concepts sous forme de termes (appelé TBox) la possibilité de définir explicitement des instances à l'aide d'assertions (appelées ABox).

9.2 Les langages de représentation de connaissances par objets

Cette section va montrer les caractéristiques des langages de représentation par objets (LRPO), sans trop entrer dans les détails puisque le système utilisé, TROPES, sera présenté en profondeur dans le chapitre suivant. De plus, certaines de ces caractéristiques ont été abordées lors de la succincte partie sur le système Shirka (Cf. §8.5). Il existe également des similitudes avec les langages terminologiques.

9.2.1 Classes et instances

Les LRPO font la séparation entre les deux types de frame, même si la structure du frame (schéma dans Shirka) est toujours la même et formée des trois niveaux frame-attribut-facette (figure 8.8). La classe possède une description (son *intension*) sous forme d'un ensemble d'attributs précisés par des facettes ; l'ensemble des instances de cette classe est appelé son *extension*. Une instance est liée à sa classe (unique le plus souvent) par le lien *est-un*, et donne des valeurs aux attributs en accord avec leur spécification dans la classe ; néanmoins, des instances incomplètes peuvent être créées si des attributs n'ont pas reçu de valeurs.

9.2.2 Attributs et facettes

Les attributs d'une classe sont les constituants de sa description, et représentent en général une caractéristique (ou une propriété) des objets représentés par la classe. Chaque attribut peut recevoir une seule ou plusieurs valeurs d'un type défini ; ceci est spécifié par une facette de typage comme *un* ou *ensemble-de* dont la valeur peut être un type simple (entier, chaîne, etc.) ou un type correspondant à une classe de la base (l'attribut *rbs* de la classe *gène-protéique* prend ses valeurs dans la classe *RBS* (figure 8.8)). Ce dernier cas permet de lier des objets les uns avec les autres, mais comporte un défaut important en ce que ce lien est souvent utilisé avec une sémantique différente de celle pour laquelle il a été défini, dans le but de tirer avantage de l'existence du lien et de son utilisation *ad hoc*. C'est pourquoi des systèmes fournissent des mécanismes pour exprimer d'autres relations ayant une sémantique particulière, comme par exemple la relation de composition. Cet aspect sera abordé plus en détail ultérieurement (section 9.3).

Les facettes de typage sont précisées par des facettes permettant de réduire le type en spécifiant un domaine ou un intervalle de valeurs à l'attribut, en restreignant le nombre de valeurs possibles pour les attributs multi-valués ou en ajoutant un prédicat que la valeur doit satisfaire. Il existe également des facettes d'inférence liant un attribut à une méthode (c'est-à-dire du code exécutable) ou à un filtre sur des instances de la base.

9.2.3 Hiérarchie de classes et héritage

Les classes d'une base de connaissances sont organisées en une hiérarchie dite de spécialisation grâce au lien *sorte-de*, qui induit un ordre partiel sur les classes. À l'instar de la relation de subsomption des langages terminologiques, la sémantique extensionnelle de ce lien énonce que l'ensemble des instances d'une classe est contenu dans l'ensemble des instances de sa (ses) sur-classe(s) (suivant que le système autorise l'héritage multiple ou non). De même, intentionnellement, la description d'une sous-classe comporte soit un raffinement des attributs existants, soit l'ajout de nouveaux attributs. Par contre, cette hiérarchie n'est (en général) pas construite automatiquement, mais est fournie par le développeur de la base de connaissances.

Un mécanisme d'héritage est associé à cette relation, tel que la description d'une classe, à travers ses attributs, est héritée par ses sous-classes. Des problèmes de conflit peuvent se poser si on autorise l'héritage multiple, quand par exemple une classe hérite d'un attribut de même nom provenant de plusieurs de ses sur-classes [Masini et al.91].

9.2.4 Les mécanismes d'inférence

L'instanciation est le premier de ces mécanismes, car la création d'une instance lui confère tout de suite une structure et un comportement particuliers donnés par la classe. L'instanciation n'est entérinée que si les valeurs proposées pour ses attributs sont en accord avec la description de la classe à laquelle on veut lier l'instance.

D'autres mécanismes ont pour but d'inférer la valeur d'un attribut, en utilisant les facettes dont on a parlé : méthode ou filtrage. L'exécution de la méthode renvoie un résultat qui doit satisfaire les contraintes posées sur l'attribut en question ; le filtrage parcourt des objets de la base et les apparie avec le filtre, ne gardant que les objets qui satisfont la description du filtre.

Un dernier mécanisme est la *classification* ; son but est de déterminer, pour une instance donnée, la description la plus précise en accord avec elle, c'est-à-dire en accord avec les valeurs de ses attributs. Le processus consiste donc à essayer de descendre l'instance dans la hiérarchie des classes tout en vérifiant que les valeurs de ses attributs satisfont les contraintes de la classe testée. Chaque classe reçoit de la sorte une étiquette selon le résultat du test ; si chaque attribut de la classe testée a une valeur dans l'instance qui satisfait les contraintes de la classe, celle-ci est étiquetée *sûre*, si un attribut de la classe testée a une valeur dans l'instance en contradiction avec les contraintes de la classe, celle-ci est étiquetée *impossible*, sinon elle est étiquetée *possible* (dans le cas où des attributs n'ont pas de valeur). Ce processus est répété récursivement sur les sous-classes sûres et possibles (la figure 10.2, page 127 explique la classification multi-points de vue, qui est plus générale que la classification classique).

Il est intéressant de se poser la question à laquelle ont répondu les langages terminologiques sur le statut de la classe vis-à-vis de sa description : cette dernière est-elle un ensemble de conditions nécessaires ou nécessaires et suffisantes d'appartenance ? Les différents systèmes restent assez vagues à ce sujet, mais on considère que l'instanciation utilise la description de la classe comme des conditions nécessaires (il faut que l'instance satisfasse cette description pour y être rattachée, mais c'est l'utilisateur qui décide si cela suffit), mais que par la suite, dans le processus de classification, ce sont des conditions nécessaires et suffisantes (ce qui permet effectivement d'étiqueter une classe comme sûre quand toutes les contraintes sont satisfaites).

9.3 Liens et relations

Les réseaux sémantiques ont la particularité d'exprimer n'importe quelle relation, dont le sens est contenu dans le nom et à laquelle n'est en général pas associé de comportement spécifique. Les langages de frames ont restreint cette généricité en ne conservant que certains liens privilégiés, en l'occurrence *est-un* (lien d'instanciation) et *sorte-de* (lien de spécialisation). Un autre lien est celui d'*attribution* qui lie deux objets par l'intermédiaire des attributs. Si la sémantique des deux premiers liens est bien définie, celle de l'attribution l'est moins, même si, en général, ce lien représente une propriété d'un objet, dont la valeur est un autre objet. Cette surcharge du lien d'attribution pour des utilisations pour lesquelles il n'était pas prévu a amené certains systèmes à fournir soit d'autres liens (comme la relation de composition), soit même la possibilité d'en définir dynamiquement, et d'en préciser le comportement. La relation de composition a été particulièrement étudiée, et on a pu remarquer son importance dans la modélisation des cartes génomiques. De plus, les relations qualitatives et quantitatives définies lors de cette modélisation nécessiteront également une représentation.

9.3.1 La relation de composition

L'étude de la relation de composition a montré à quel point ses significations pouvaient être nombreuses. Winston *et al.* [Winston et al.87] l'ont classée dans une taxonomie basée sur trois aspects :

- la fonctionnalité : les parties possèdent un rôle fonctionnel par rapport au tout ;
- l'homogénéité : les parties sont similaires entre elles et au tout ;
- la séparabilité : les parties peuvent être séparées du tout auquel elles appartiennent.

Ils ont ainsi mis en évidence six relations différentes (tableau 9.2). Markowitz *et al.* [Markowitz et al.92] n'en distinguent que quatre, dont trois transitives équivalentes à celles de Winston *et al.*, marquées d'un rond (∘) dans le tableau.

		Propriétés des éléments		
Relation	Exemple	Fonctionnalité	Homogénéité	Séparabilité
composant/tout ∘	roue/voiture	+	-	+
membre/collection ∘	arbre/forêt	-	-	+
portion/masse ∘	grain/sel	-	+	+
matière/objet	fer/clef	-	-	-
trait/activité	payer/faire des courses	+	-	-
lieu/région	oasis/désert	-	+	-

Tableau 9.2 - : Les six types de relation de composition. La transitivité est valide à l'intérieur d'un même type de composition, mais ne l'est plus quand on mélange ces types.

L'obligation de choisir une relation parmi ces six vient du fait que la propriété de transitivité n'est valide qu'au sein d'une même relation. En effet, l'application de la transitivité à deux expressions de relations de composition différentes donne des résultats faux en général. Par exemple, si on combine «la main de Pierre est une partie de Pierre» (composant) et «Pierre est une partie de l'équipe» (membre), on obtient «la main de Pierre est une partie de l'équipe», ce qui est faux quelle que soit la sémantique de la relation choisie.

Le type de relation choisi dépend de la sémantique voulue ; de façon générale, c'est la relation composant-tout qui est prise car elle s'applique bien aux entités physiques du monde réel, et à le bon goût d'autoriser des décompositions récursives qui sont parfois bien utiles. Dans le cadre de la modélisation de cartes génomiques, c'est manifestement celle-ci qui convient. D'autres applications seraient sans doute amenées à sélectionner une relation *ad hoc* différente.

Un autre aspect lié à la relation de composition – et qui apparaît également quand on veut spécifier une nouvelle relation – est le comportement des composants vis-à-vis de l'objet composite. Quatre types de comportement ont été définis ; ils sont fonction de propriétés de dépendance et de partage [Kim et al.89]. Un composant est dépendant de son composé s'il ne peut pas exister sans lui ; ceci implique que la destruction d'un composant entraîne récursivement celle de ses composants. Un composant peut être partagé si plusieurs composés peuvent se l'approprier comme composant. Sur ce point aussi, les systèmes qui modélisent la relation diffèrent. Une dépendance existentielle et exclusive du composant vis-à-vis de l'objet composite implique que l'existence même du composant est soumise à celle de l'objet composite et que tout partage est interdit ; l'objet composite encapsule alors ses parties, qui ne sont visibles qu'à travers lui [Blake et al.87]. Un argument avancé pour justifier ces choix est que les objets physiques ne peuvent appartenir à plus d'un composé. Néanmoins, l'exemple des cartes génomiques réfute cette justification puisque les entités cartographiques sont bien partagées par différentes cartes, même à l'intérieur d'un point de vue ; ceci vient du fait que les modélisations des objets peuvent être telles qu'il est intéressant de les considérer selon différents critères ; ces critères ne sont d'ailleurs pas nécessairement choisis par la modélisation mais peuvent être imposés par le monde réel (à l'instar des expériences biologiques qui mettent en évidence des cartes qui se recouvrent et qui partagent donc des entités).

C'est encore une fois l'application qui impose le choix des comportements ; dans le cas des cartes génomiques, les composants peuvent exister librement, indépendamment de la présence d'un composite ; de plus, leur partage est bien sûr autorisé.

9.3.2 Les relations

Il existe deux façons de représenter une relation générale entre deux ou plus de deux objets. La première consiste à mettre le lien dans les objets eux-mêmes pour pouvoir y accéder directement ; cette manière de faire souscrit à la «philosophie» objet qui encapsule toute la connaissance d'un objet dans cet objet. Par contre, l'inconvénient de cette démarche est l'obligation (pour des raisons de cohérence et d'efficacité) de définir le lien inverse dans l'objet correspondant (pour les relations binaires). Ceci induit une redondance de l'information qui se retrouve dans plusieurs entités. L'autre moyen est de réifier la relation en en faisant un objet à part entière, contenant la description de la relation et les objets qu'elle lie.

Dans le cas de la biologie moléculaire, c'est cette seconde approche qui sera utilisée car une grande partie du raisonnement va être effectué sur ces relations, plus que sur les objets qu'elles lient. Il est donc plus adéquat de disposer d'une classe qui regroupera toutes les relations pour pouvoir les utiliser immédiatement, sans avoir à les chercher dans les objets.

Chapitre 10

Le modèle Tropes

Tropes est un système de représentation de connaissances par objets et utilisant l'approche classe-instance, avec la particularité d'intégrer deux entités descriptives supplémentaires qui sont le concept et le point de vue. Le modèle minimal a été proposé par Olga Mariño [Mariño93] ; peu de modifications y ont été apportées, mais des extensions ont été développées ou sont en cours de développement ; ces extensions incluent en particulier la définition d'un module de gestion des types, celle d'un modèle de tâches et l'implantation d'un module de traitement des contraintes et des relations. Ces deux derniers aspects sont intimement liés l'un à l'autre, et sont des ajouts importants au modèle, en ce qui concerne les aspects de cohérence et d'inférence.

10.1 Les entités du modèle

Une base de connaissances Tropes est partitionnée en *concepts* disjoints qui peuvent être observés selon un certain nombre de *points de vue*. Chaque concept représente une famille d'individus qui en sont les instances. Ces instances sont décrites par un ensemble d'*attributs-concept*, dont un sous-ensemble non vide forme la clef du concept, c'est-à-dire les valeurs indispensables pour distinguer une instance du concept d'une autre ; ces valeurs sont les garants de l'intégrité de toute instance. Toute création d'une instance d'un concept nécessite donc de donner une valeur aux attributs-concept qui en sont la clef. Cette clef peut consister en un seul attribut qui spécifie le nom de l'instance (comme par exemple dans le concept *gène*, dont la clef est l'attribut-concept *nom*), ou peut être plus complexe pour les concepts qui en ont besoin (ainsi, le concept *personne* a pour clef les attributs-concept *nom*, *prénom* et *date-de-naissance*).

Les points de vue sous lesquels un concept est visible établissent une catégorisation des attributs-concept ; si les attributs qui forment la clef sont visibles selon tous les points de vue, les autres ne sont pertinents que suivant un ou plusieurs points de vue spécifiés dans la base de connaissances.

Les attributs-concept décrivent des caractéristiques immuables des instances du concept ; ces caractéristiques sont son *type*, sa *nature*, qui précise si un attribut a une sémantique de propriété, de composition ou, plus généralement, de relation, son *constructeur* spécifiant s'il s'agit d'un attribut mono ou multi-valué (ensemble ou liste), et enfin les *tâches* qui serviront éventuellement à en déterminer la valeur. Ces informations pourront être affinées lors de descriptions ultérieures de l'attribut dans les classes.

De la même façon que la base de connaissances est partitionnée en concepts, elle l'est aussi par les points de vue, même si pour des raisons d'implémentation, tout point de vue est une entité du concept. C'est dans chaque point de vue d'un concept qu'existe une hiérarchie de spécialisation de classes ; chacune de ces hiérarchies est en fait un arbre, qui interdit donc le multi-héritage et les problèmes de conflit afférents. La justification de cette démarche est que, de manière générale, une classe a plusieurs sur-classes lorsque justement on se la représente selon des perspectives différentes, c'est-à-dire lorsque les points de vue manquent. La racine de chaque hiérarchie dans chacun des points de vue est une classe dont l'extension représente l'ensemble des instances du concept ; les classes racine possèdent le même nom puisque leurs extensions sont confondues. Une instance de concept est rattachée à une et une seule classe de chaque point de vue, par défaut la classe racine.

La possibilité de spécifier l'égalité d'extensions de deux classes d'un concept dans deux points de vue différents, qui se rencontre pour les classes racine, se généralise grâce aux *passerelles*. Une passerelle *bidirectionnelle* entre deux classes C_1 et C_2 correspond à ce cas précis où toute instance de C_1 est instance de C_2 et réciproquement. Plus généralement, une passerelle *unidirectionnelle* relie un certain nombre de classes dites *source*, $C_1, \ldots, C_n$ à une classe C, chaque classe appartenant nécessairement à un point de vue différent, si l'intersection des extensions de toutes les classes $C_1, \ldots, C_n$ est contenue dans l'extension de C. Exprimée logiquement, cette propriété se lit : si I est une instance de C_1 dans le premier point de vue, ..., de C_n dans le énième point de vue, alors I est une instance de C.

La figure 10.1 montre les entités du modèle rencontrées : les concept, point de vue, classe, instance, passerelle.

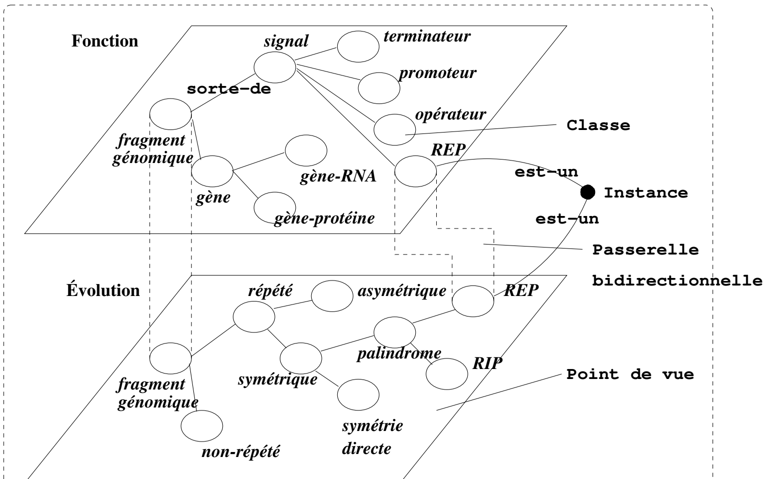


Figure 10.1 - : Description d'un concept en TROPES. Le concept de *fragment génomique* est représenté selon deux points de vue, l'un contenant la connaissance fonctionnelle et l'autre la connaissance évolutive. Les deux hiérarchies de classes sont des arbres reliés par l'intermédiaire de passerelles unidirectionnelles ou bidirectionnelles. Les classes racine des points de vue sont automatiquement reliées par une passerelle bidirectionnelle, de même que les classes de même nom (comme REP).

La hiérarchie de classes à l'intérieur des points de vue est construite de la même façon que dans les LRPO classiques (Cf. section 9.2) ; une classe possède des *attributs-classe*, précisés par des facettes, elle hérite la connaissance de sa sur-classe. Une particularité liée aux passerelles est que deux classes reliées par une passerelle bidirectionnelle et qui possèdent un attribut-classe de même nom ont la même description pour cet attribut, à savoir l'union des descriptions (i.e. des facettes).

Deux hypothèses importantes sont faites sur la hiérarchie, une hypothèse d'*exclusivité* et une hypothèse de *non-exhaustivité*. La première impose que deux classes sœurs soient disjointes, une instance appartenant *exclusivement* à une classe. Ceci aura des conséquences au niveau de la classification (Cf. section 10.3.2). Il est important de noter que ce requis de la base de connaissances ne peut être vérifié à partir des seules descriptions des classes (même si des incohérences peuvent être mises en évidence), et que c'est son concepteur qui en a la charge et la responsabilité. La seconde hypothèse est que, par contre, les sous-classes d'une classe ne forment pas obligatoirement une partition de cette classe. Leurs descriptions ne représentent pas exhaustivement celle de leur sur-classe.

10.2 Le système de types

L'adjonction d'un système de types à un modèle de connaissances à objets comme Tropes [Capponi94] présente de nombreux avantages liés à l'extensibilité, la connexion avec des modules procéduraux, ainsi que l'efficacité et la sécurité de mécanismes d'inférence tels que la classification.

Le système de types développé, appelé MÉTÉO (pour *Module Extensible de Types Élaborés pour les Objets*), comprend deux niveaux de typage.

- D'une part, des classes de types (appelés C-types) représentent des structures de données particulières (spécification d'ensembles de valeurs et d'opérations applicables à ces valeurs). De nouveaux C-types, tels que *matrice* ou *séquence génomique*, peuvent être intégrés par l'utilisateur ; ils seront considérés de la même façon que les C-types primitifs comme *entier*. Un C-type permet la description d'opérations de manipulation de ses valeurs, ainsi que la spécification d'opérations de manipulation de ses parties : c'est au niveau du C-type qu'est décrite, par exemple, la relation de sous-typage qui tient compte des propriétés de ses valeurs. De même, c'est au niveau du C-type que peut être décrite la mesure de similarité entre deux valeurs.

- D'autre part, à chaque C-type est associé un ensemble de Δ-types qui représentent les parties de l'ensemble caractérisé par le C-type. Les Δ-types sont organisés en treillis par la relation de sous-typage traduisant l'inclusion ensembliste. Par exemple, dans le cas du C-type *entier*, les Δ-types sont des listes d'intervalles fermés de valeurs entières ; la relation de sous-typage dérive de la relation d'inclusion entre listes d'intervalles.

Toute entité abstraite du modèle de représentation est typée :

- les classes se voient associer un C-type si la spécification d'opérations particulières sur des éléments de cette classe est nécessaire, ou un Δ-type issu du C-type *record* sinon. Les attributs sont typés par des Δ-types ;

- les instances elles-mêmes sont typées de façon à prendre en compte statiquement les contraintes qui portent sur elles. L'opération de typage d'une entité, d'une part prend en compte sa définition statique, d'autre part intègre dynamiquement les résultats donnés par le système de gestion des contraintes.

Le système de types considère de façon générique les opérations des C-types et les utilise pour l'interface entre le modèle de connaissances et la base de types. Il est constitué principalement d'une opération de typage (comprenant une phase de normalisation), d'une opération de test de sous-typage, d'opérations génériques telles que l'union, la disjonction ou l'intersection de types. Ces opérations génériques utilisent les opérations particulières de chaque C-type. MÉTÉO réalise en outre la gestion dynamique des treillis de Δ-types. En conséquence, toute opération d'ajout, de suppression ou de modification d'un Δ-type est dynamiquement prise en compte par le système de types.

Les primitives offertes par le système de types sont alors directement utilisables par les mécanismes d'instanciation et d'inférence du modèle de connaissances tels que le filtrage, la classification de classe ou d'instance, la gestion de contraintes ou encore l'activation d'attachements procéduraux.

La propriété d'extensibilité permet l'intégration de nouvelles structures de données de façon à personnaliser la base de types pour une application particulière. Enfin, le modèle de représentation des connaissances manipule indifféremment des structures de données extraites, ou non, des entités de représentation, et ceci du fait de l'homogénéité des C-types qui sont une implémentation de ces abstractions.

10.3 Les mécanismes d'exploitation de Tropes

Les mécanismes d'exploitation de TROPES utilisent les connaissances contenue dans une base de connaissances de manière à en déduire de nouvelles. L'instanciation et la classification sont les mécanismes fondamentaux disponibles ; les tâches et les contraintes sont des mécanismes plutôt extérieurs, même s'ils ont été intégrés au modèle.

10.3.1 L'instanciation

Elle se fait en premier lieu au niveau du concept, en donnant des valeurs aux attributs (au moins à tous les attributs-clefs), puis au niveau des classes en en précisant une dans chaque point de vue ; la classe-racine est prise par défaut si aucune classe n'est spécifiée dans un point de vue. L'instance n'est créée que si toutes les contraintes de tous les attributs auxquels a été assignée une valeur sont satisfaites.

10.3.2 La classification

La classification dans TROPES a pour but de déterminer dans chacun des points de vue la classe la plus spécialisée à laquelle l'instance peut être rattachée ; elle représente néanmoins plus qu'une simple concaténation du mécanisme décrit sur les LRPO (Cf. section 9.2.4), car elle profite de l'existence des passerelles pour améliorer l'étiquetage des classes avec les marques «sûre», «possible» et «impossible». De plus, elle utilise également l'hypothèse d'exclusivité en éliminant lors de la recherche toutes les classes sœurs d'une classe étiquetée «sûre».

L'utilisation des passerelles repose sur leur sémantique ; en effet, si une passerelle existe entre les n classes $C_1, \ldots, C_n$ et la classe C, et que les classes $C_1, \ldots, C_n$ sont sûres, alors la classe C peut être étiquetée sûre également. Réciproquement, s'il existe une passerelle partant de la classe C pour aller à la classe C' et que la classe C' a été marquée impossible, alors la classe C est aussi étiquetée impossible (une version encore plus générale de cette propriété comporte plusieurs sources dont toutes sauf une sont sûres et une destination étiquetée impossible, alors la dernière source est impossible) (figure 10.2).

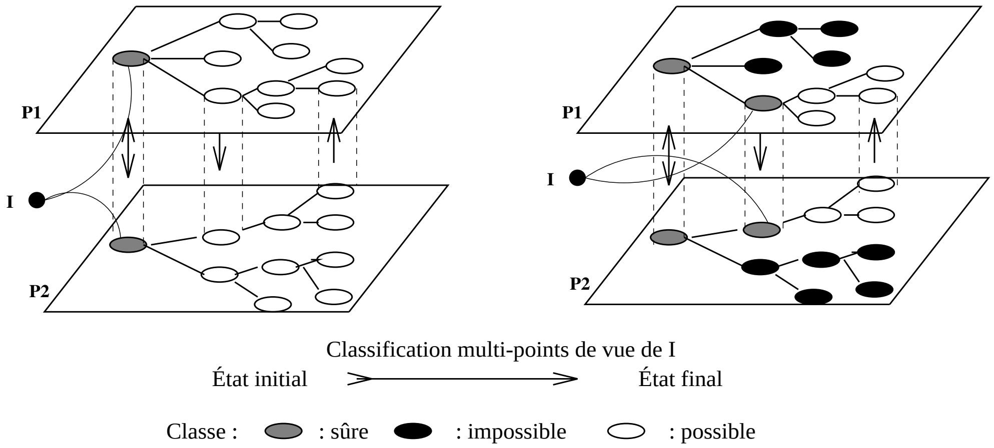


Figure 10.2 - : La classification multi-points de vue dans TROPES [Mariño et al.90]. L'état initial représente une instance I rattachée aux classes-racine de chaque point de vue. Après l'opération de classification, les classes ont été étiquetées selon la satisfaction des contraintes de leurs attributs par les valeurs des attributs de l'instance, et en usant des passerelles entre les points de vue.

La classification des objets composites, qui fait intervenir les attributs dont la nature est *composant*, se divise en deux mécanismes. Lors de la classification minimale, un composant n'est classé dans son concept que si cela est nécessaire pour classer son composite ; la classification maximale, plus coûteuse, cherche également à positionner les composants le plus bas possible dans leurs concepts respectifs [Mariño91].

Les mécanismes d'inférence qui suivent (tâches et contraintes) sont des extensions au noyau.

10.3.3 Le modèle de tâches

TROPES se démarque des LRPO au niveau du calcul des valeurs des attributs en ce sens qu'il n'utilise pas l'attachement procédural classique qui lie un attribut dans une classe à une méthode par une facette spécifique ; dans TROPES, cette information est reportée au niveau du concept et est placée dans la définition des attributs-concept. Ce mécanisme a été choisi pour se débarrasser du problème de l'utilisation de l'attachement procédural pour déterminer l'appartenance d'une instance à une classe ; en effet, activer une méthode d'une classe à laquelle on veut tester l'appartenance de l'instance, c'est déjà préjuger de cette appartenance [Rechenmann92]. Le détachement procédural de TROPES permet de sélectionner la tâche ou la méthode à activer pour calculer la valeur d'un attribut en fonction de cette valeur en classant l'instance dans une hiérarchie de spécialisation de tâches.

Le modèle de tâches proposé [Gensel et al.92] est basé sur la définition d'un concept *tâche* qui organise l'ensemble des tâches destinées au calcul de valeur d'attributs en hiérarchies. Les attributs qui décrivent une instance du concept *tâche* sont :

- son but qui décrit en une chaîne de caractères l'objectif que la tâche permet d'atteindre ;
- ses entrées représentées par une ensemble de valeurs ;
- son traitement : cet attribut décrit la stratégie de résolution à appliquer pour résoudre la tâche en donnant les sous-tâches sous forme de composants ; la résolution de la tâche passe par celle de chacune de ces sous-tâches de décomposition ;
- ses sorties représentées comme les entrées par un ensemble de valeurs et dont le type doit être satisfait par le résultat de la tâche.

Dans la hiérarchie de tâches, on distingue quatre types de tâches suivant le niveau de précision des divers composants. Une tâche est dite *abstraite avec but abstrait* lorsque sa classe ne fournit qu'une description générale. Une tâche est dite *abstraite avec but concret* lorsque sa classe indique explicitement le but atteint mais ne définit pas le traitement précisément. Une tâche est dite *concrète non terminale* lorsque sa classe décrit une décomposition en sous-tâches particulières. Une tâche est dite *concrète terminale* lorsque sa classe indique une procédure (figure 10.3).

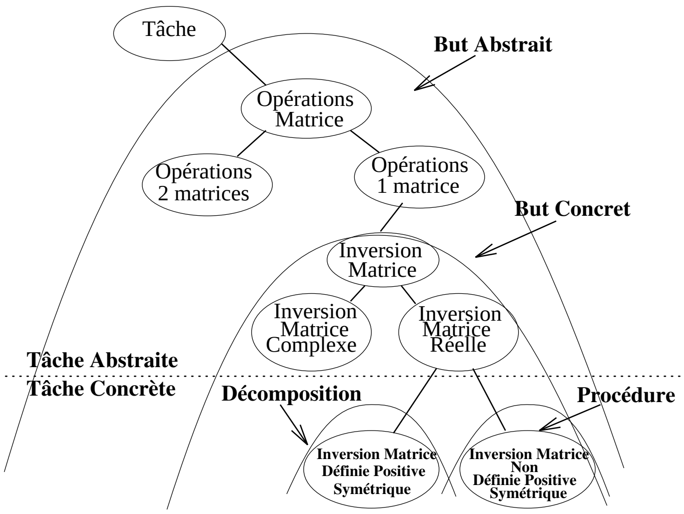


Figure 10.3 - : Le concept «Tâche». Les opérations sur les matrices deviennent de plus en plus précises au fur et à mesure qu'on descend dans la hiérarchie, passant de tâches générales utiles pour la structuration de la connaissance (*Opérations Matrice, Opérations 2 Matrices et Opérations 1 Matrice*) pour arriver aux tâches concrètes d'inversion auxquelles une décomposition ou un code exécutable (sous forme de procédure) est disponible.

L'exécution d'une tâche [Gensel et al.93] débute par l'instanciation du concept *tâche*. Cette instance est ensuite classée dans la hiérarchie de tâches jusqu'à trouver la classe la plus spécifique à laquelle elle appartient. Si cette classe est une tâche concrète non terminale, le processus se poursuit par l'instanciation successive et récursive des tâches de décomposition ; la réussite de la tâche est reportée à celles de ces sous-tâches. Si la classe est concrète terminale, la procédure adéquate est lancée ; sa réussite ou son échec détermine celle de la tâche.

10.3.4 Les contraintes

La déclarativité et les capacités d'inférence d'un modèle tel que TROPES peuvent être étendues par la définition et la maintenance de relations mathématiques entre les attributs. Les contraintes peuvent être définies aux différents niveaux que sont les concepts, les classes et les instances. D'autre part, ces réseaux de contraintes sont dynamiques : des contraintes peuvent être ajoutées ou supprimées à tout moment.

Un Module d'Intégration de Contraintes et de Relations aux Objets (MICRO) a été conçu. Couplé à TROPES, il y est chargé de la maintenance des réseaux de contraintes. MICRO propose une panoplie de contraintes pré-définies classiques (numériques et booléennes), mais aussi spécifiques aux attributs multivalués de TROPES (ensemblistes, sur des listes), ou plus originales (conditionnelles, d'évolution). MICRO utilise une représentation par objets des contraintes et gère les liens entre les attributs contraints des instances de TROPES et les attributs des instances de contraintes associées. Les techniques de propagation reposent sur des méthodes génériques de maintien de la consistance associées aux classes de contraintes. La maintenance des contraintes numériques exploite les principes de l'arithmétique des intervalles. La résolution des contraintes est basée sur un algorithme de «forward-checking» paramétrable pour les domaines finis et sur une méthode de fractionnement dynamique pour les domaines infinis ou continus.

Les contraintes s'avèrent un puissant outil de maintien de la cohérence pour le modèle. La maintenance des contraintes assure la consistance du domaine d'un attribut contraint et, par là même, est amenée à modifier son type. Le type effectif d'un attribut contraint (dans le concept, la classe, ou l'instance) est alors calculé par MICRO après la phase de propagation, puis transmis au module de gestion des types. Les contraintes étendent donc la définition des types des attributs.

Enfin, les contraintes ont été utilisées pour l'extension du modèle lui-même. Ainsi, le partage de propriétés entre un objet composite et ses composants, le passage de paramètres entre une tâche et ses sous-tâches, la sémantique des liens de modélisation de relation et les requêtes effectuées à base de filtres, sont décrits et contrôlés par des contraintes.

10.4 Réflexivité de Tropes

Une particularité utile du système TROPES est sa réflexivité, c'est-à-dire le fait que les entités même du modèle sont définies dans le formalisme de TROPES, sous forme de concepts, points de vue, classes, etc. L'intérêt de cette auto-définition est de permettre l'extensibilité du modèle, sans avoir à casser pour reconstruire ensuite. Nous en ferons la preuve en implémentant le formalisme des cartes génomiques en étendant le méta-modèle, c'est-à-dire la couche supplémentaire qui fait que toutes les entités du modèle sont décrites en son sein.

Pratiquement, ceci signifie que tout concept d'une base de connaissances est aussi une instance d'un méta-concept, qui est appelé *Concept* ; de même, tout point de vue, toute classe, tout attribut, etc., est instance du concept correspondant au méta-niveau. La figure 10.4 montre les méta-concepts les plus importants, et qui sont très fortement interconnectés, *Concept*, *Point-de-vue* et *Classe*.

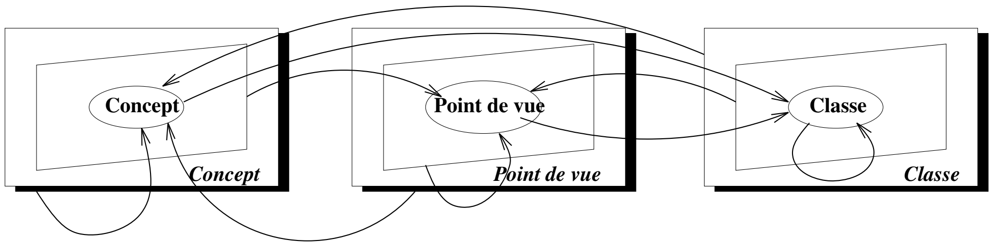


Figure 10.4 - : Le méta-modèle de TROPES. Le concept *Concept* regroupe toutes les instances de concept, dont lui-même, les autres concepts du méta-modèle comme *Point-de-vue* et *Classe*, et tous les concepts définis dans une base de connaissances, comme *Fragment-génomique* ; ce concept ne comporte qu'un seul point de vue et qu'une seule classe, instances respectivement des deux autres méta-concepts montrés ici, qui sont *Point-de-vue* et *Classe*. Les flèches représentent le lien d'instanciation.

Il est déjà possible de préjuger de l'extension à effectuer dans le concept *Concept* pour regrouper les concepts modélisant les entités cartographiques qui seront amenés à être comparés entre eux. Au lieu que ces concepts soient des instances du concept *Concept*, il suffira de les faire instances d'une sous-classe de la classe *Concept*, créée à cet effet (Cf. §11.2).

Chapitre 11

Implémenter les cartes génomiques en Tropes

La formalisation des cartes génomiques s'implémente d'autant plus facilement à l'aide de TROPES que certaines de ses caractéristiques s'adaptent naturellement aux cartes (en particulier, l'existence d'une représentation multi-points de vue et la relation de composition). Dans un premier temps, nous allons montrer comment implémenter, sous forme de concepts, les entités décrites lors de la formalisation, puis la méta-connaissance liée aux cartes et qui concerne les entités et les relations entre elles (fonctions *dim*, etc.), et enfin, nous nous attaquerons à la représentation des relations et du raisonnement associé.

11.1 Implémentation de la connaissance biologique

Ce qui nous intéresse ici est la représentation dans le modèle TROPES de la typologie des cartes génomiques qui a été faite à la section 3.2, sans entrer dans les détails d'une modélisation complète ; en particulier, la description des concepts se limitera à exprimer les différents types de carte et les fonctions de construction. Celle des attributs, méthodes, etc., des concepts biologiques ne sera pas abordée.

11.1.1 Types de carte et points de vue

Les différents types de carte ont un vis-à-vis naturel dans le modèle, qui sont les points de vue. En effet, chaque type de carte crée une séparation des entités, de la même manière qu'un point de vue partitionne une base de connaissances. Ainsi, tous les concepts de biologie moléculaire qui sont nécessaires à la modélisation de cartes posséderont trois points de vue, cytogénétique, génétique et physique suivant leur pertinence dans le type de carte correspondant. Les concepts *Carte* et *Gène* qui apparaissent sur n'importe quelle carte apparaîtront dans le modèle sous les trois points de vue ; par contre, le concept *Bande* qui n'est pertinent que sur la carte cytogénétique n'apparaîtra que dans le point de vue cytogénétique, de même que *Intron* et *Exon* ne seront visibles que sous le point de vue physique.

11.1.2 Constitution des entités et relation de composition

La relation de composition, telle qu'elle a été conçue dans le modèle TROPES, permet non seulement d'implémenter la fonction *const* qui donne les constituants de chaque entité, mais également la fonction *desc* qui s'applique aux cartes elles-mêmes. En effet, comme cette relation est fondée sur l'utilisation d'attributs, elle se soumet à la partition par points de vue ; ceci signifie qu'un objet composite peut disposer de décompositions multiples, autant qu'il y a de points de vue, ce qui correspond exactement aux spécifications de la fonction *const* qui associe à un couple (type, point de vue) les types constitutifs du type initial dans ce point de vue. De plus, toute instanciation d'un concept doit se faire en accord avec les contraintes sur ses attributs – composites ou non, en particulier leur type ; instancier un objet composite nécessite donc que ses attributs composites prennent leur valeur dans le type spécifié.

La figure 11.1 montre comment sont utilisés les points de vue et la relation de composition pour représenter dans le modèle certains concepts biologiques.

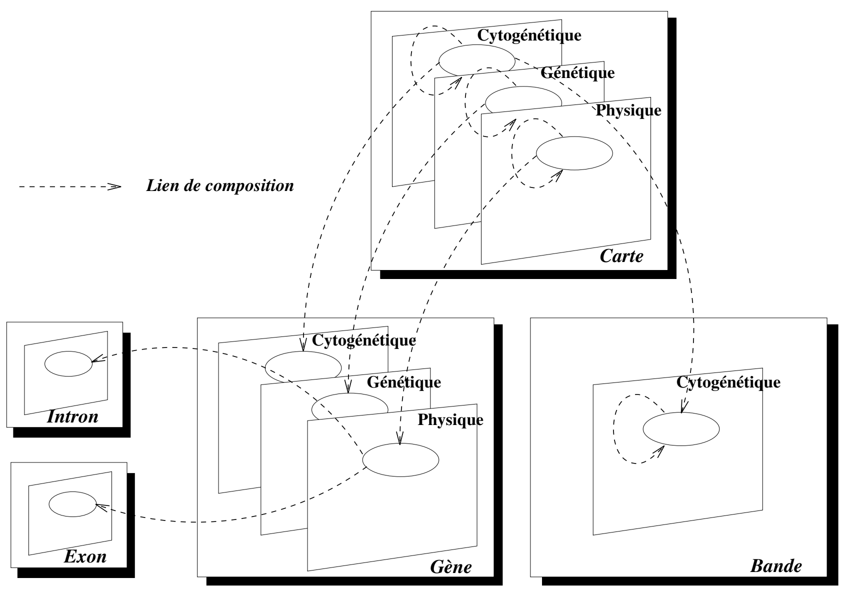


Figure 11.1 - : Représentation de concepts biologiques dans TROPES. Le concept de carte apparaît sur les trois points de vue, puisqu'il est utilisé dans les trois types. Il est lié, à travers le lien de composition, aux concepts *Gène* et *Bande* : au premier selon les trois points de vue, au second selon le seul point de vue cytogénétique. Pour ce faire, il suffit de créer dans le concept *Carte* un attribut composite *gènes* prenant ses valeurs dans le concept *Gène* et visible dans tous les points de vue, et un autre attribut composite *bandes* qui prend ses valeurs dans le concept adéquat et visible uniquement dans le point de vue cytogénétique. De même, un gène est défini dans tous les points de vue et se décompose physiquement en introns et exons.

11.2 Description de la méta-connaissance

Il nous faut désormais représenter la connaissance sur les entités, c'est-à-dire regrouper les concepts biologiques comparables, sur lesquels le raisonnement aura lieu, exprimer quelles sont les dimensions de ces concepts dans leurs points de vue respectifs, quelles sont les unités des concepts et si celles-ci sont additives.

Pour cela, le méta-modèle de TROPES a été étendu, en définissant des sous-classes des méta-concepts *Concept* et *Point-de-vue* et en leur ajoutant si nécessaire de nouveaux attributs. Regrouper les concepts comparables selon l'optique des cartes génomiques est indispensable, car une base de connaissances, même restreinte à l'étude des cartes, se compose obligatoirement d'une multitude de concepts, dont certains n'ont rien à voir avec la construction des cartes. Il en est ainsi de concepts représentant des commentaires, des références aux auteurs ou à des articles, etc. Pour les distinguer de ces autres concepts, ont été créées des sous-classes du concept *Concept*, telles que tous les concepts utiles dans la modélisation des cartes et comparables pour la construction sont instances d'une de ces classes (appelée *Construction-cartes*). Cette spécialisation pourrait également servir à réunir des concepts de représentation temporelle (figure 11.2). De plus, un nouvel attribut-concept a été créé, pour lier cette classe aux relations qu'il est possible de définir entre ces concepts.

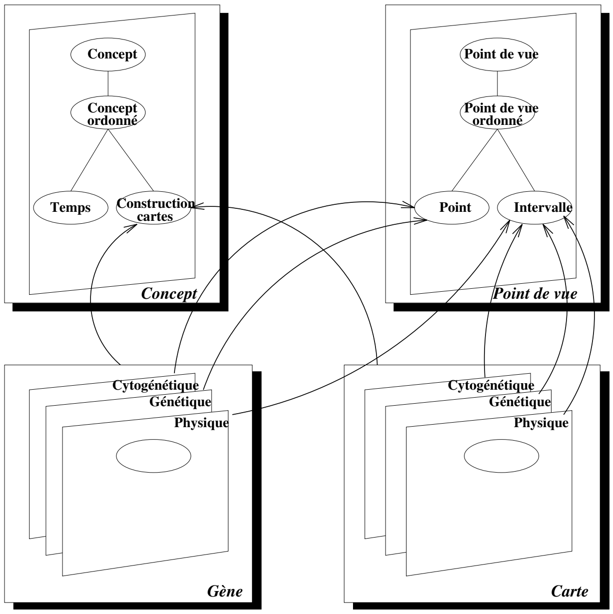


Figure 11.2 - : Extension du méta-modèle. Au lieu d'être des instances de la classe-racine du concept *Concept*, les concepts cartographiques comparables entre eux sont regroupés par leur appartenance à la classe *Construction-cartes*. De même, chaque point de vue de ces concepts est instance du méta-concept *Point-de-vue*, mais est lié à une sous-classe de la racine indiquant si cette entité est un point ou un intervalle dans ce point de vue. Ainsi, le point de vue génétique du concept *Gène* est instance de la classe *Point* tandis que le point de vue physique est rattaché à la classe *Intervalle*.

Le concept *Point de vue* a aussi été étendu de façon à représenter les dimensions des entités. Comme celles-ci dépendent précisément du type de carte où une entité est définie, il a suffi de créer deux sous-classes de la classe-racine, modélisant respectivement les points de vue dans lesquels l'entité est un point, et ceux dans lesquels c'est un intervalle (figure 11.2).

Un méta-attribut du concept *Point de vue* fait le lien avec l'unité du point de vue. Ce méta-attribut prend ses valeurs dans le concept *Unit*, qui ne comprend que deux attributs, le premier indiquant le nom de l'unité, le second si cette unité est additive (figure 11.3). Le modèle TROPES impose une redondance d'information car, même si concepts et points de vue sont indépendants, ces derniers sont rattachés à un concept. Ainsi, il faut répéter l'information associant un point de vue à une unité pour chacun des concepts de la base de connaissances, même si les points de vue en question ont le même nom.

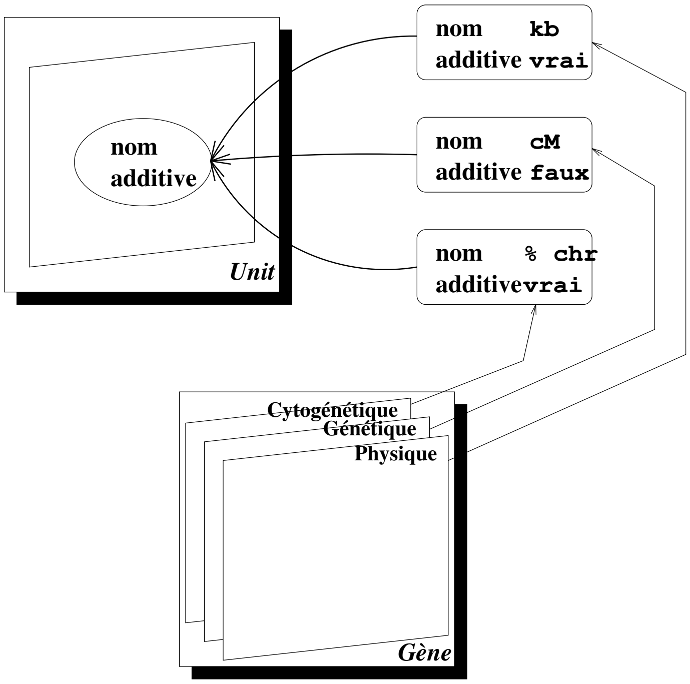


Figure 11.3 - : Le concept *Unit*. Une unité est caractérisée par son nom ; elle comporte un attribut supplémentaire indiquant si elle est additive ou non. Dans le cas des cartes génomiques, trois instances de ce concept ont été définies, une pour chacun des trois types de carte. Tout point de vue comporte un méta-attribut pointant sur l'instance d'unité adéquate ; par exemple, le point de vue génétique du concept *Gène* pointe sur l'unité cM (il en est de même des autres points de vue génétiques des autres concepts car un point de vue dans le modèle TROPES est rattaché à un concept).

De même, nous définissons un concept *Scale*, dont la clef est un couple d'unités, et qui contient un attribut *échelle* donnant la valeur du facteur d'échelle entre les deux unités. Ce concept n'est pas utilisé puisque, comme cela a été vu, il n'existe pas de facteur d'échelle entre les différentes cartes génomiques. Néanmoins, il pourrait être utilisé pour inférer des relations approximatives basées sur une valeur constante du rapport entre unités.

11.3 Représentation des relations

Les relations, qu'elles soient qualitatives ou quantitatives, sont représentées dans un concept spécifique appelé *Relation*. Ce concept contient un certain nombre d'attributs représentant les éléments de la formalisation. La clef du concept est constituée des attributs *noms* et *entités*, qui prennent leur valeur respectivement dans un ensemble de chaînes de caractères et un ensemble d'instances de la base de connaissances. Le choix d'un ensemble de noms permet de représenter les disjonctions de relations atomiques. L'attribut *entités* contient, dans le cas des relations qualitatives orientées, l'entité de référence et les entités mises en relation. Il est nécessaire de placer la référence dans les attributs-clefs car, si elle avait été ajoutée en tant qu'attribut à la classe des relations qualitatives orientées, il n'aurait pas été possible de créer deux relations différant seulement par la valeur de cet attribut, ce qui non seulement doit être autorisé, mais permet en outre d'inférer des relations d'orientation (cas de la fusion où les relations sont identiques).

La figure 11.4 montre l'organisation du concept *Relation* ; au fur et à mesure qu'on descend dans la hiérarchie des classes, les valeurs possibles pour l'attribut *noms* sont de plus en plus restreintes, pour se limiter à une seule chaîne de caractères au niveau des classes terminales. La classe des relations quantitatives est adjointe de deux nouveaux attributs spécifiant les distances possibles entre les extrémités des entités en relation.

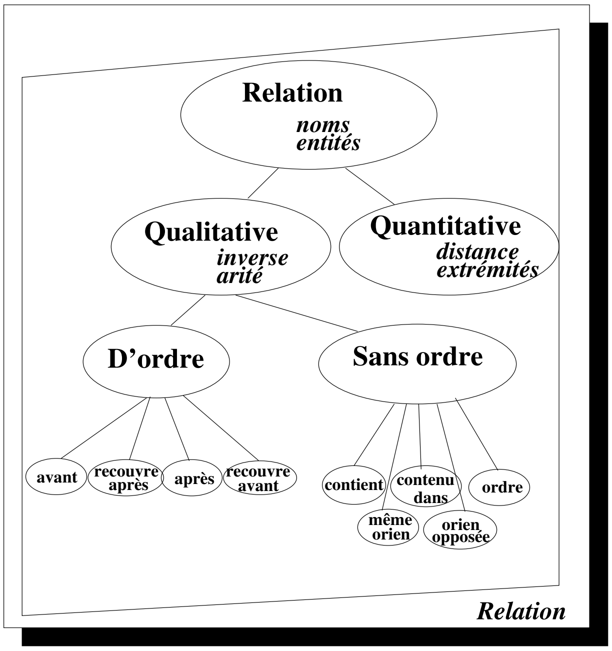


Figure 11.4 - : Le concept *Relation*. Il se scinde en deux parties, une classe de relations qualitatives et l'autre de relations quantitatives, qui partagent les attributs *noms* et *entités*. D'autres attributs sont définis pour représenter les informations nécessaires à l'expression des relations qualitatives (*inverse* et *arité*) et quantitatives (*distance* et *extrémités*). L'attribut *noms* est progressivement restreint au fur et à mesure que les classes se spécialisent.

Quelques exemples d'instances du concept *Relation* sont donnés dans le tableau suivant (tableau 11.1).

11.4 Correspondance entre la formalisation et l'implémentation

Maintenant que l'implémentation dans Tropes a été décrite, dans quelle mesure est-elle en accord avec la formalisation ? Au niveau de la typologie (Cf.§3.2),

- les types introduits sont représentés par des concepts de la base de connaissances ; il en est ainsi des types *Carte*, *Gène*, etc. ; l'ensemble des unités est aussi représenté

Classe	*noms*	*entités*	*distance*	*extrémités*
Qualitative	(avant, contient)	(Carte1, TSHB, NotI)	–	–
Orientées	(après, recouvre_avant)	(Carte1, PGM1, D1S14)	–	–
Avant	(avant)	(D1S21, intron1, exon1)	–	–
Quantitative	(distance)	(D1S21, D1S17)	((9.1, .0))	(orig, orig)

Tableau 11.1 - : Instances de relations. Même si *a priori* les relations initiales sont atomiques, il faut pouvoir représenter des disjonctions de relations, car celles-ci risquent d'apparaître après les traitements de l'algorithme de construction de cartes.

grâce à l'introduction d'un concept ;

- l'ensemble des points de vue est directement lié au modèle TROPES ;
- les fonctions sur les ensembles ont été représentées de manière différente : la fonction *const* est liée intrinsèquement à la modélisation de la relation de composition ; la fonction *dim* s'implémente à l'aide de l'extension du méta-modèle ; enfin, les fonctions *unit*, *add* et *scale* sont liées à la fois à la définition de nouveaux concepts et à l'utilisation de méta-attributs ;
- les propriétés des fonctions sont assurées pour les deux premières par la modélisation du concept *Carte* ; les suivantes ne peuvent être contrôlées que par l'ajout de code, soit au modèle TROPES, soit au sein de l'algorithme de construction de cartes.

Au niveau des relations, les ensembles et les fonctions sont implémentés dans les concepts *Relation* et *Distance*. Par contre, toutes leurs propriétés sont contenues dans l'algorithme de construction de cartes, que ce soit au sein des résolutions des CSP ou non.

Le tableau suivant (tableau 11.2) récapitule les éléments de la formalisation et l'implémentation correspondante en TROPES.

11.5 Contraintes et algorithmique des cartes génomiques

Il est raisonnable de se demander pourquoi nous n'avons pas utilisé le système de contraintes de TROPES pour résoudre le problème de construction de cartes, puisque les contraintes temporelles qui sont à la base de l'algorithme ne sont en fin de compte que des contraintes particulières. Il y a plusieurs raisons à cela.

- La première raison est que le système de gestion de contraintes de TROPES est encore en cours de développement ; les évolutions du langage utilisé au cours des deux dernières années ont nécessité des réécritures qui n'ont pas été très faciles. Actuellement, TROPES, qui fonctionne avec le système de types et le module de gestion des contraintes sur la version 2.0 de Talk, est traduit en Talk 3.0 ; les nombreuses différences entre ces deux versions rendent la tâche d'autant moins aisée.

Formalisation	*Modèle* TROPES
Type	Concept
Point de vue	Point de vue
Concepts comparables	Sous-classes de *Concept*
Fonction *const*	Relation de composition
Fonction *desc*	Relation de composition
Fonction *dim*	Sous-classes de *Point-de-vue*
Fonction *unit*	Concept *Unit* et attribut de *Point-de-vue*
Fonction *add*	Attribut de *Unit*
Relation	Concept *Relation*
Inverses	Attribut de *Relation*
Arité	Attribut de *Relation*
Propriété 1	Modélisation du concept *Carte*
Propriété 2	Modélisation des points de vue de *Carte*

Tableau 11.2 - : Récapitulatif de la correspondance entre la formalisation et TROPES. À chaque élément de la formalisation est associée une implémentation particulière basée sur le modèle lui-même, sur une extension du modèle ou sur l'écriture de la base de connaissances.

- Une autre raison, plus fondamentale, est que MICRO a pour objectif de traiter les contraintes numériques (et booléennes). Or, les contraintes que nous voulons exprimer sont symboliques : elles s'expriment, dans le cas des contraintes temporelles entre intervalles, comme une disjonction de symboles associés aux relations atomiques d'Allen. Même s'il est possible de représenter par des nombres tant les disjonctions de relations que la table de transitivité[1], la représentation serait plus lourde et moins lisible que celle mettant en jeu des instances du concept *Relation*.

- Enfin, une dernière restriction à l'utilisation de MICRO est son incapacité à traiter les incohérences. L'apparition d'une incohérence entraîne en effet une remise en cause de la dernière instanciation qui a provoqué cette incohérence. L'utilisation de MATS permet au contraire de disposer de réseaux de contraintes indépendants les uns des autres, et de ne pas dépendre des mécanismes propres à MICRO qui ne conviennent pas.

11.6 Implémentation des cartes génomiques

Tout ce qui a été présenté dans ce chapitre a été implémenté. Cette implémentation recouvre divers aspects.

- Implémentation de TROPES : celle-ci a d'abord été précédée d'une phase importante de spécifications, en particulier au niveau du noyau, à laquelle ont participé nombre

[1] La représentation interne de ces disjonctions dans MATS est un nombre, qui est son code binaire : par exemple, 43 s'écrit en binaire 101011, et représente donc la disjonction des première, seconde, quatrième et sixième relations. Les opérations de l'algèbre d'intervalles peuvent alors se faire par l'application d'opérateurs logiques sur les codes binaires des disjonctions.

de doctorants de l'équipe ; elle a été suivie de l'implémentation effective en Le-Lisp v16, puis en Talk 2.0 pour aboutir prochainement à une version en Talk 3.0.

- Extension du méta-modèle : elle a nécessité la création des sous-classes des concepts *Concept* et *Point-de-vue*, et la liaison effective des entités biologiques avec cette extension.

- Définition des concepts *Relation*, *Unit*, *Distance* : celle-ci s'est faite dans le langage de description de base de connaissances de TROPES, qui spécifie les éléments des concepts.

- Base de connaissances cartographique : cette base a également été écrite en TROPES et contient la description de la constitution des concepts biologiques dont nous avons eu besoin.

Troisième partie

Les interfaces cartographiques

Chapitre 12

Logiciels de cartographie

Les cartes génomiques sont aussi des objets graphiques. Une carte, en particulier si elle est le résultat d'un consensus, se visualise graphiquement, sous la forme d'un axe, d'entités positionnées sur cet axe et d'informations de distance entre ces entités. Une telle représentation contient de la connaissance sur l'ordre des entités et les distances qui les séparent, c'est pourquoi ces deux informations sont celles qui sont modélisées et recherchées en priorité par les systèmes détaillés précédemment (chapitre 8). L'importance de l'aspect graphique accordée par les biologistes fait que, souvent, c'est l'interface homme-machine qui détermine le succès d'un logiciel, toutes choses égales par ailleurs. La plupart des systèmes existants consistent en une couche graphique *ad hoc* rajoutée au-dessus d'une base de données ; peu importe que les structures de données soient des objets ou des relations, le point important à souligner est l'absence de modèle sous-jacent. On aboutit ainsi en général à des systèmes spécifiques de certaines espèces, dont l'extension à d'autres, même si elle est possible, est lourde et pénible, car ces systèmes manquent de la généricité qu'aurait apportée l'élaboration d'une formalisation, même minime.

Les logiciels présentés ici se distinguent les uns des autres suivant ces différents points : l'existence d'un modèle de représentation, la généricité, et également l'existence de moyens intégrés de raisonnement.

Nous ne présenterons ci-après que des logiciels dont nous avons pu nous servir, et cette contrainte restreint ceux-ci à des systèmes non-commerciaux disponibles sur station de travail sous Unix. Nul doute qu'il existe des logiciels, commerciaux ou non, satisfaisant les mêmes besoins sur d'autres machines telles que Macintosh ou PC, mais nous n'avons pas été à même de les étudier.

12.1 ACeDB

ACeDB [Durbin et al.92, Sulston et al.92] est un logiciel de représentation cartographique qui a fait son chemin ; il est largement employé par les biologistes moléculaires, et tend à devenir un standard de fait, malgré ses limitations. Initialement conçu pour stocker des informations de cartographie et de séquençage et les afficher concernant le nématode *Cœnorhabditis Elegans*, ACeDB a été plusieurs fois adapté à d'autres organismes, en particulier à l'Homme pour l'étude du chromosome 21. Son succès a été – et reste encore – fortement lié à son interface, très appréciée des biologistes. Celle-ci possède l'avantage de visualiser les cartes d'une manière qui plaît car c'est celle avec laquelle les biologistes ont l'habitude de travailler. Elle consiste principalement en une fenêtre composée de l'axe cartographique (vertical pour les cartes génétiques, horizontal pour les cartes physiques) muni d'une échelle, sur lequel sont positionnées de façon plus ou moins précise les entités cartographiques ; de plus, un marqueur indique la position de la partie de carte visible sur le chromosome entier (figure 12.1).

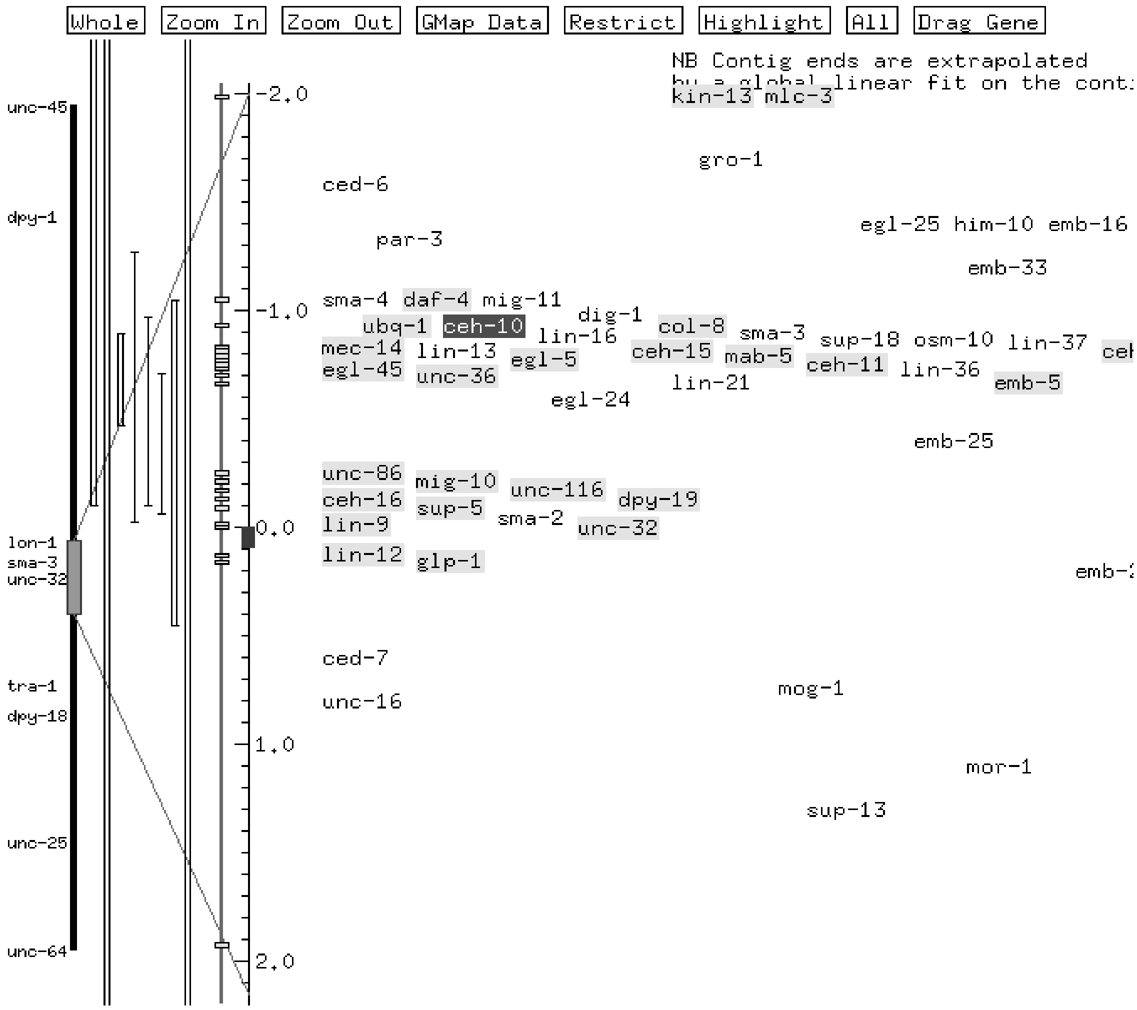


Figure 12.1 - : Interface graphique d'ACeDB. Cette copie d'écran montre la carte génétique, ainsi que les entités qui y apparaissent ; l'action d'un clic de la souris sur une entité sélectionnée (comme ceh-10) ouvre une fenêtre contenant des informations textuelles, et permet d'accéder à des informations contingentes grâce à un hypertexte.

ACeDB dispose de fonctionnalités diverses, dont certaines sont classiques et fournies par la quasi-totalité des systèmes, comme la possibilité de zoomer sur la carte, de visualiser des informations textuelles sur les entités en double-cliquant dessus, d'interroger la base de données pour récupérer des entités qui seront mises en évidence graphiquement à l'écran, etc. Mettons tout de même l'accent sur le langage de requêtes intégré à ACeDB, plus riche que ceux rencontrés en général dans les autres systèmes.

Même si les données sont représentées dans des classes d'objets, celles-ci ne servent qu'en tant que structure de données ; aucune hiérarchie n'existe, il n'y a donc pas d'héritage. Une classe est un moule pour une entité, que ce soit une entité de la représentation (gène, chromosome, séquence, etc.) ou non (référence bibliographique, commentaire, auteur, laboratoire, etc.).

12.2 SIGMA

SIGMA [Sigma92] est l'acronyme de *System for Integrated Genome Map Assembly* ; c'est un outil destiné à la création, l'édition et la maintenance de cartes génomiques intégrées c'est-à-dire incluant n'importe quel type de carte et représentant des informations en provenance de toute expérience. L'ensemble des entités est ainsi visualisé dans une seule fenêtre, qui intègre toutes les cartes, quelle que soit leur résolution (figure 12.2). Ceci ne permet bien sûr pas de disposer de représentations multiples en fonction de la carte.

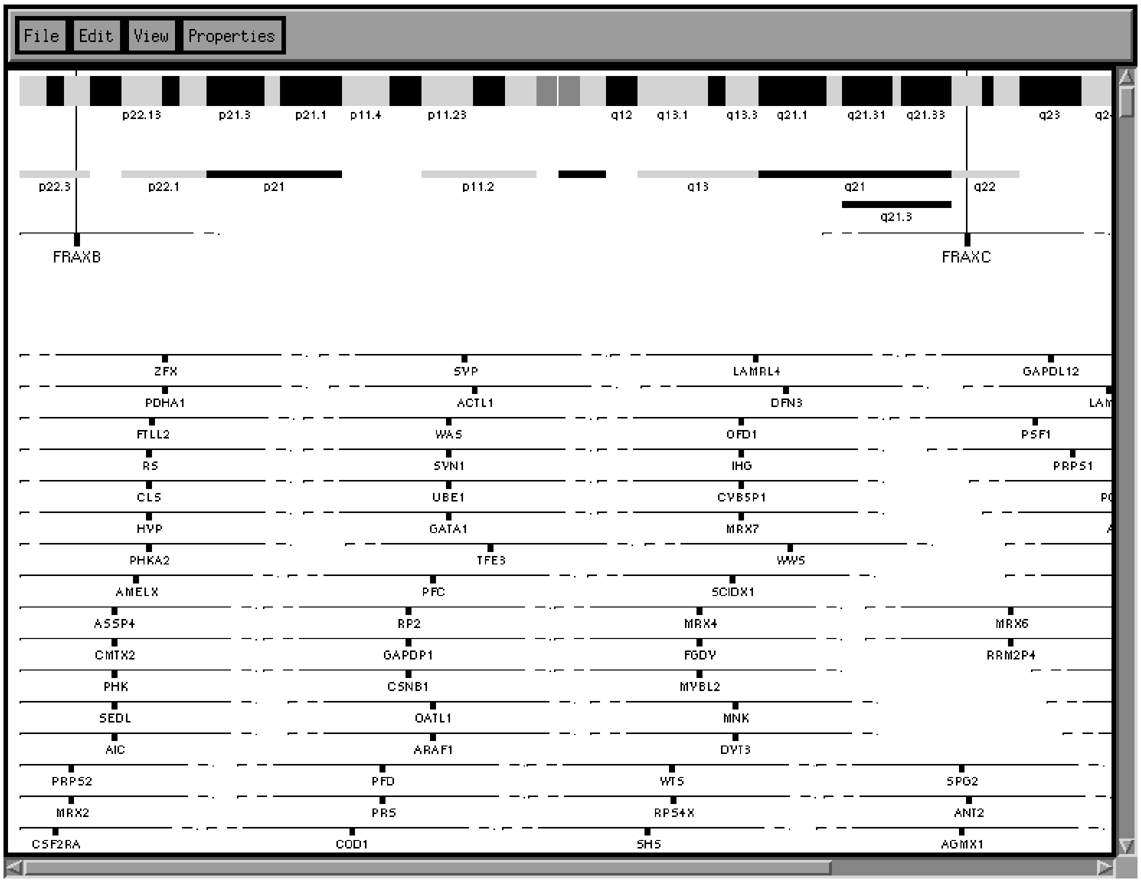


Figure 12.2 - : Interface graphique de SIGMA. Celle-ci se compose d'une seule fenêtre qui intègre l'ensemble des cartes ; la représentation graphique permet de visualiser des incertitudes sur les positions des entités.

Même si l'ensemble des informations relatives à une carte se retrouve dans un unique fichier, c'est-à-dire sous une forme peu structurée, SIGMA met en évidence une formalisation minime de ce qui doit être représenté. Ainsi, ce fichier contient une table des types d'entité, chacun d'entre eux étant lié à une représentation graphique particulière qui précise la couleur, les dimensions, l'étiquetage, etc., des différents types. De plus, la carte globale peut elle aussi être modifiée selon les desiderata de l'usager ; si, par défaut, elle contient tous les types d'élément, ceux-ci peuvent être restreints pour n'en conserver qu'un certain nombre. L'ensemble de ces informations graphiques forme le style de la carte ; il est possible de sauver ces styles et de les réutiliser.

SIGMA apporte également la possibilité de représenter des relations entre éléments de carte. Il ne se limite donc pas seulement à afficher une carte dont la position des entités les unes par rapport aux autres est connue intégralement. On peut en effet exprimer des relations qualitatives, au nombre de six, qui sont *à-droite-de*, *à-gauche-de*, *recouvre-à-droite*, *recouvre-à-gauche*, *contient*, *contenu-dans* et *espace (gap)* pour indiquer une disjonction. Des relations quantitatives peuvent aussi être exprimées, que ce soient des informations sur la longueur d'une entité ou une distance entre deux entités.

À partir de toutes ces informations, incluant les positions globales des entités par rapport au chromosome, les relations qualitatives et quantitatives (appelées *objectifs de la carte*), le système détermine les localisations supposées de toutes les entités qui maximisent le nombre des objectifs satisfaits, à l'aide d'un algorithme de recherche opérationnelle – sur lequel aucun détail n'est donné.

Un autre point original du système est la gestion qui est faite des problèmes de conversion entre les différentes unités. La non-linéarité des conversions d'une carte à l'autre est traitée grâce à l'introduction d'un mécanisme de conversion de régions ; la carte globale est morcelée en régions dans lesquelles il existe un facteur de conversion unique. De cette manière, les informations de distance peuvent être utilisées lors de la construction automatique de la carte pour passer d'un type de carte à un autre.

Enfin, SIGMA dispose d'un moyen pour représenter des variations individuelles c'est-à-dire faire coexister en plusieurs exemplaires des copies d'une entité sur laquelle on ne possède pas beaucoup d'information.

SIGMA fournit des fonctionnalités plus ou moins classiques de manipulation graphique : le zoom, la sélection d'objet, l'affichage des propriétés d'un objet (position, relations avec les autres objets, etc.), des commentaires concernant les références liées à une entité, le déplacement avec la souris d'un objet et la recherche d'éléments à travers des requêtes.

Malheureusement, de nombreuses fonctionnalités ne sont pas encore implémentées, et cela rend difficile l'évaluation des capacités réelles du système. En l'occurrence, les explications concernant la gestion des conversions semblent un peu rapides ; on ne voit pas trop comment cela pourrait fonctionner avec plus de deux cartes, puisque la solution proposée impose alors de trouver une région où tous les facteurs de conversion des cartes deux à deux sont constants. D'autres points qui demanderaient des éclaircissements sont le choix d'une entité globale par rapport à laquelle sont exprimées toutes les positions, en liaison avec le manque de gestion des orientations, les détails de l'algorithme de construction de carte à partir des objectifs et ses caractéristiques.

12.3 GnomeView

GnomeView [Douthart et al.94, Gnomeview93] est encore une interface cartographique de visualisation et manipulation axée plus particulièrement sur les requêtes. En effet, l'affichage des cartes se fait seulement à travers une requête demandant une recherche d'entités dans deux banques de données qui sont la *Genome Data Bank* (qui contient des informations chromosomiques) et *GenBank* (qui est une banque de séquences). Le lancement d'une requête se fait en remplissant les champs d'une fenêtre de dialogue (figure 12.3).

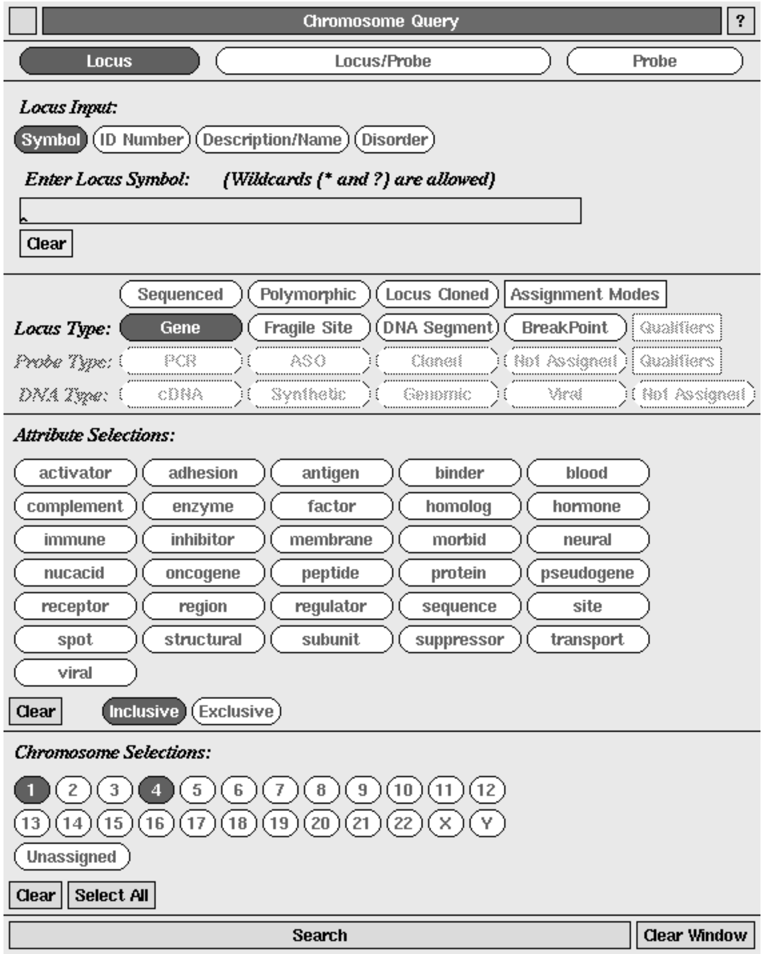


Figure 12.3 - : Fenêtre de dialogue des requêtes sous GnomeView.

C'est le résultat de la requête qui apparaît sous la forme d'une carte cytogénétique des chromosomes humains et d'une carte dite de densité qui montre par un niveau de gris la proportion des entités qui satisfont la requête dans chacune des bandes (figure 12.4). L'action d'un clic de la souris sur les bandes ou les entités récupérées est d'obtenir des informations textuelles, qui sont intégrées à un hypertexte. Il s'agit donc principalement d'une interface de récupération d'information de banques de données biologiques. La carte n'est qu'un support à ces informations.

Si on interroge les banques sur les séquences, on peut visualiser, en plus de la position de la séquence sur la carte cytogénétique, la séquence elle-même issue de GenBank. L'ac-

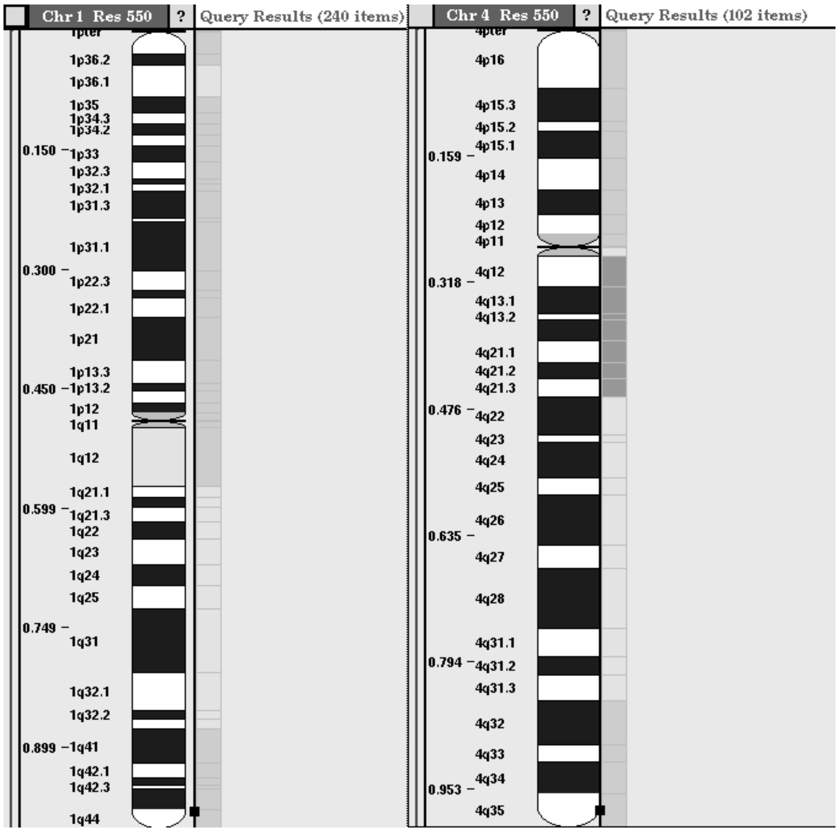


Figure 12.4 - : Résultat de la requête chromosomique. Celui-ci se présente dans une fenêtre affichant la carte du (ou des) chromosome(s) sur le(s)quel(s) la requête a été effectuée, avec en vis-à-vis une carte de densité qui met en évidence les entités trouvées par la requête. Celles-ci sont sensibles à un clic de la souris, qui fournit des informations textuelles.

tion de zoomer sur cette séquence apporte de plus en plus d'information au fur et à mesure que la résolution augmente, jusqu'à aboutir en fin de compte à la séquence (figure 12.5).

12.4 HoverMaps

Le système HoverMaps [Dorkeld94], déjà décrit lors de la présentation des outils de modélisation cartographique, a l'avantage de fournir une interface cartographique basée sur son modèle de représentation de connaissances, à la différence des interfaces précédentes pour lesquelles la connaissance n'est pas structurée, étant contenue dans de simples fichiers. L'implémentation de cette interface peut donc s'appuyer sur les objets cartographiques définis dans le modèle sous-jacent.

En effet, la possibilité de lier un concept de la base de connaissances à un objet graphique et une représentation particulière facilite d'une part le développement de cette interface, tout en augmentant la généricité, d'autre part, permet de lier plus facilement les actions sur l'interface aux objets de la base de connaissances qu'elle visualise. Par exemple, la modification interactive d'une dimension peut être immédiatement répercutée sur la base de connaissances, entraînant de la sorte automatiquement une vérification de la validité de la modification. Enfin, le fait de n'avoir qu'à lire la description d'un objet

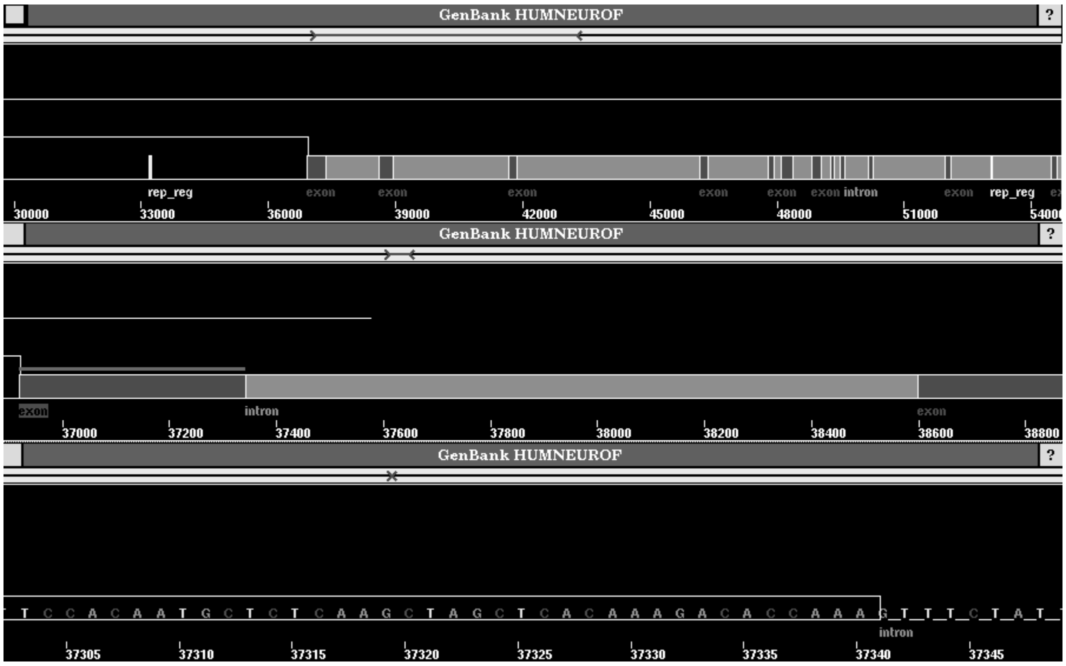


Figure 12.5 - : Visualisation d'une séquence. Au fur et à mesure que la résolution augmente en zoomant sur la séquence, la fenêtre de visualisation montre de nouvelles informations, jusqu'à aboutir à la séquence elle-même et aux nucléotides.

pour l'afficher, au lieu de parcourir tout un fichier, améliore les performances de l'outil graphique.

En dehors des outils graphiques propres à Shirka et qui permettent de visualiser la hiérarchie des classes, d'éditer les instances, etc., à travers une interface appelée IVAN [Grivaud et al.92], HoverMaps fournit des composants graphiques propres à la cartographie. Ceux-ci permettent de visualiser des cartes cytogénétiques, génétiques et physiques avec les entités présentes, de récupérer des informations textuelles et également de décrire graphiquement le résultat de calculs sur les entités de la base de connaissances, comme par exemple le calcul du taux de composition en bases G et C dans les bandes de la carte cytogénétique ou les correspondances génétiques entre les différentes espèces ou *synthénie*.

La carte cytogénétique, dans l'interface graphique, est constituée des objets graphiques représentant les bandes ; elle est liée à sa représentation dans la base de connaissances, de même que les bandes sont liées à leur description dans cette même base. Les actions effectuées sur cette carte sont alors la récupération d'une valeur d'attribut et son traitement spécifique. Par exemple, un clic de la souris dans une bande entraîne l'affichage d'un éditeur qui contient toutes les entités appartenant à cette bande. Un autre clic sur une de ces entités ouvre une fenêtre de texte sur les caractéristiques de cette entité (figure 12.6).

La carte génétique est construite de la même manière, à partir des entités de la base de connaissances. Des informations complémentaires peuvent apparaître sur sa représentation graphique, en l'occurrence les liaisons synthéniques entre l'Homme et la souris, c'est-à-dire un morceau de chromosome chez l'Homme qui correspond au segment de la carte chez la souris quant à l'ordonnancement de gènes homologues (figure 12.7).

12.5 Discussion

Il existe de nombreux autres logiciels de visualisation d'informations cartographiques, utilisables dans des conditions plus ou moins spécifiques :

- QuickMap [Quickmap94] a été développé au Généthon et est particulièrement destiné à la construction de contigs à partir de YAC ;
- EMG (Encyclopedia of the Mouse Genome) [Emg94] présente des cartes cytogénétiques, génétiques et physiques de la souris, mais pourrait vraisemblablement être étendu à d'autres espèces ;
- XGrail [Xgrail94] est un logiciel de visualisation de séquences et dispose de fonctionnalités propres à l'étude de séquences, comme la recherche des gènes ou plus généralement de régions codantes ;
- ChromoScope [Zhang et al.94] est propre à *Escherichia Coli* (ce qui nécessite de pouvoir afficher une carte circulaire puisque telle est la forme du chromosome) ;
- Genographics [Genographics93] est un logiciel très fruste, mais possède la particularité de montrer les liens entre entités identiques d'une carte à l'autre.

Si on excepte quelques particularités de ces logiciels (comme par exemple, l'affichage dépendant de la résolution dans GnomeView), ils procurent tous le même ensemble de fonctionnalités suivant :

- Affichage d'un axe gradué et d'entités ;
- Zoom ;
- Sélection d'entités ;
- Information textuelle (plus ou moins riche).

Il existe néanmoins de subtiles différences quant à la manière de répondre à ces fonctions, parfois liées à la vocation de la carte comme réceptacle de la connaissance biologique accumulée ou comme outil de construction de la carte. GnomeView affiche les cartes en fonction des réponses à une requête, ACeDB possède des cartes séparées, SIGMA intègre tous les types de carte dans une seule représentation. Le chapitre suivant montrera ce que doit être à nos yeux une interface cartographique, dans le but de rester aussi générique que possible.

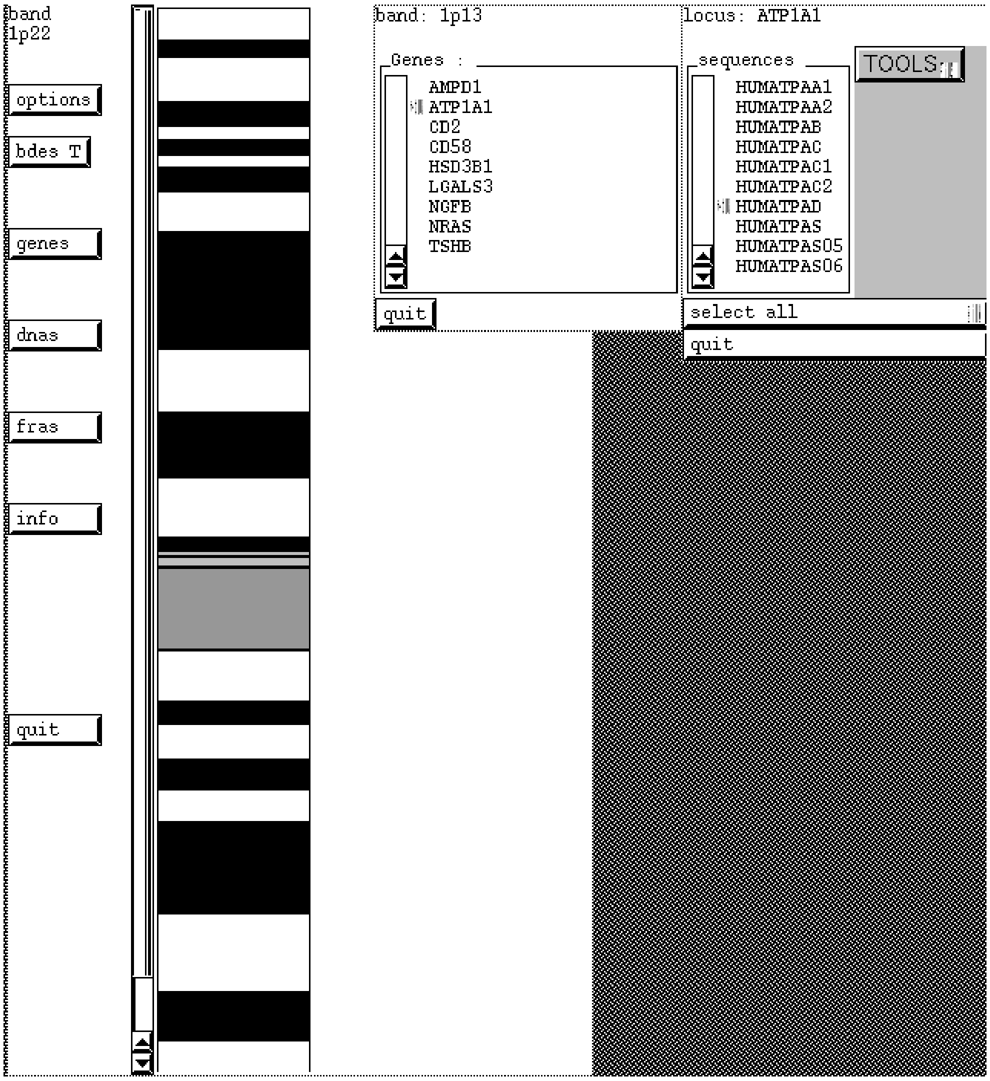


Figure 12.6 - : Visualisation de la carte cytogénétique. Celle-ci est liée à l'objet de la base de connaissances correspondant, de même que ses bandes. Un clic de la souris sur l'une d'entre elles récupère la valeur d'un attribut donnant les instances de la base contenues dans cette bande et les affiche dans un éditeur de texte. Un clic sur une entité fait apparaître des informations textuelles issues d'une banque de données.

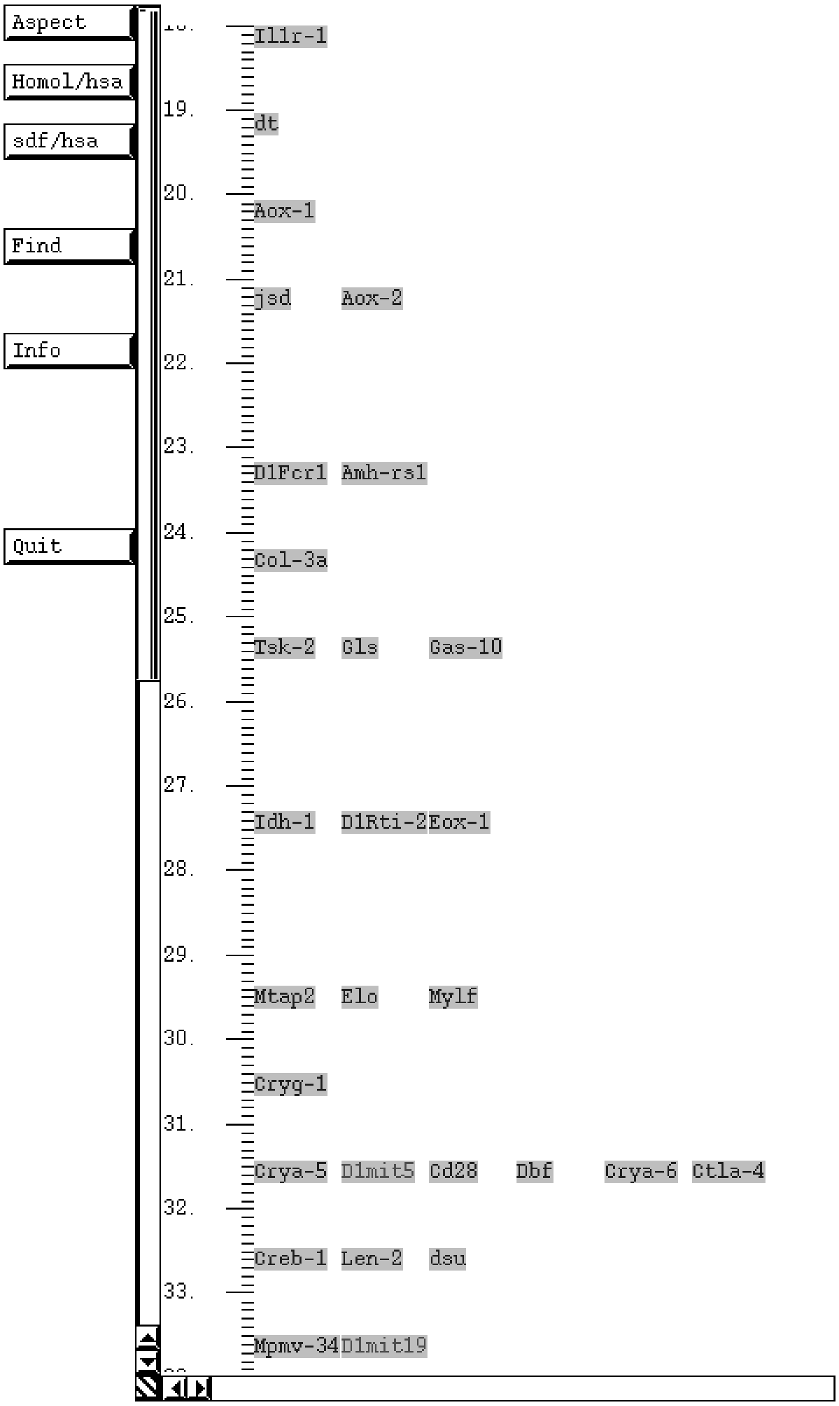


Figure 12.7 - : Visualisation de la carte génétique dans HoverMaps. On y voit classiquement un axe gradué sur lequel sont disposés les marqueurs, gènes, etc., qui y apparaissent. De plus, s'y trouve également la représentation des propriétés de synthénie entre l'Homme et la souris.

Chapitre 13

Un générateur d'interfaces cartographiques

Tous les logiciels présentés dans le chapitre précédent sont ce qu'il est convenu d'appeler des logiciels d'interface cartographique. En ce sens, ils visualisent ce que leur développeur a choisi qu'ils montrent, exécutent des tâches décidées par lui dans un cadre plus ou moins spécifique, SIGMA recevant la palme de la généricité en raison de son indépendance vis-à-vis de l'espèce, des types de carte représentés, etc.

Notre but ici est de franchir un nouveau cap dans la généricité en fournissant non plus une «simple» interface cartographique, qui ne serait qu'une copie plus ou moins améliorée des logiciels existants, mais un générateur d'interfaces cartographiques, c'est-à-dire un outil générique permettant de construire de telles interfaces. Ainsi, de la même manière qu'un générateur d'interfaces graphiques fournit des modules pour créer des courbes, pour afficher des graphes, etc., cet outil sera un module comportant des structures de données, des fonctions de création, d'accès et de manipulation, etc., destiné à faciliter à un utilisateur le développement d'une interface cartographique personnalisée. En l'occurrence, une telle boîte à outils doit être à même d'engendrer l'ensemble des interfaces cartographiques. Cette généricité va bien entendu être fondée sur le modèle qui a été détaillé dans le chapitre 3.

13.1 Données initiales

La construction du générateur d'interfaces cartographiques nécessite un certain nombre d'ingrédients de façon à créer une image représentant les données cartographiques. Ils sont énumérés ici :

- un langage hôte : dans notre cas, ce sera le langage Lisp ;
- un outil de génération d'interfaces graphiques, c'est-à-dire une interface fonctionnelle offrant le moyen de créer des interfaces graphiques (fenêtres, «scrollers», dessins, etc.) : ici, une sur-couche de Lisp permettant de définir des classes d'objets graphiques ;

- un langage de représentation des données, fondé sur des structures complexes : en l'occurrence, nous utiliserons TROPES, sachant qu'un autre langage de plus bas niveau (comme un langage de programmation par objets) pourrait convenir, l'important étant de disposer de fonctions d'accès aux données.

Tous les noms des variables ou des fonctions qui seront décrites par la suite suivront une convention de nommage séparant les entités de la base de connaissances de celles liées à l'interface ; les premières auront un nom commençant par *bdc*, les secondes par *gic*.

13.2 Interface cartographique et modélisation

La représentation graphique des cartes est une version «compilée» des relations entre ses éléments, qui montre des informations sur leur ordre et les distances qui les séparent. Ce sont ces deux seules informations qui sont immédiatement disponibles visuellement sur la carte. C'est pourquoi elles ont une importance particulière en cartographie génomique. Les autres relations n'y apparaissent pas, même si bien sûr elles subsistent dans les données et peuvent être utilisées par la suite.

L'affichage d'une carte nécessite donc uniquement la donnée de la position et de l'incertitude de chaque entité par rapport à un point de repère, choisi en général comme l'origine de la carte.

Néanmoins, cet affichage s'enrichit d'un certain nombre de caractéristiques liées au type de carte, à l'unité de l'axe, à l'additivité des distances sur cet axe, etc. ; tous les éléments décrits lors de la modélisation interviennent pour créer une représentation graphique correcte de la carte. Par exemple, si les distances sont additives, il n'est pas besoin de pouvoir afficher les distances entre entités non consécutives, ce qu'il faut pouvoir faire dans le cas des cartes génétiques.

C'est pourquoi, puisque l'interface requiert ces informations lors de l'affichage des cartes, il est indispensable, dans une phase d'initialisation, de spécifier les caractéristiques des cartes. Cette initialisation va donner des valeurs à des variables chargées de représenter les différents points de vue, les types des entités, les fonctions *const* et *dim*, etc. Elle peut se faire automatiquement si ces données sont stockées d'une manière ou d'une autre dans des structures adéquates. Si, par exemple, le mode de représentation des connaissances est TROPES, le parcours de la base de connaissances permet de récupérer toutes ces informations, en puisant aux bons endroits les valeurs des variables du modèle.

Les variables suivantes du générateur d'interfaces cartographiques sont donc définies pour contenir ces informations :

- *gic-l-pdvs* a pour valeur une liste de points de vue ;
- *gic-l-types* a pour valeur une liste des types des entités ;
- *gic-l-units* a pour valeur une liste d'unités ;
- *gic-const* a pour valeur un tableau indexé par les types et les points de vue donnant une liste de types constitutifs ;

- *gic-dim* a pour valeur un tableau indexé par les types et les points de vue donnant la dimension des entités ;
- d'autres qui seront précisées lors des spécifications fonctionnelles du générateur d'interfaces cartographiques (Cf. chapitre 14).

13.3 Interface cartographique et représentation des données

Deux interrogations existent concernant les liens entre l'interface cartographique et les données qu'elle visualise. La première concerne la distinction à faire entre ce qui est du domaine de cette interface cartographique et ce qui relève de l'interface du système à base de connaissances ; la seconde traite des interactions qui existent entre ces deux éléments.

13.3.1 Visualisation des données

A priori, tout ce qui concerne la visualisation d'instances relève de la base de connaissances et d'un éditeur spécifique du modèle de connaissances. De même, toute information textuelle liée à une instance utilise un moyen de visualisation propre qui n'appartient pas à l'interface cartographique, même si cet éditeur d'instances et cette documentation sont accessibles à partir de l'interface cartographique. Celle-ci a pour seul rôle l'affichage des cartes et la gestion du comportement des objets affichés selon les spécifications d'un utilisateur. Ce comportement inclut la recherche d'informations textuelles dans la base de connaissances et son affichage grâce à une interface de la base.

13.3.2 Interactions base de connaissances – interface cartographique

La base de connaissances et l'interface cartographique interagissent naturellement, puisque, d'une part, les cartes sont le reflet de ce qui est présent dans la base de connaissances, et d'autre part, des actions sur cette interface peuvent impliquer des modifications de la description des entités dans la base.

Pour cela, il est indispensable que chaque entité de la base de connaissances ait une entité correspondante sous la forme d'un objet graphique susceptible de s'afficher à l'écran selon sa description (Cf. section 13.4) et qu'il existe un lien direct entre ces deux objets. Ce lien existe aussi bien sur les descriptions des entités (les classes) que sur les entités elles-mêmes (les instances).

Au niveau des classes (ou des concepts en Tropes), chaque description d'un type d'entité *bdc-type* dans la base de connaissances entraîne la création d'une classe d'objet graphique *gic-type* et autant de sous-classes *gic-type-pdv1, gic-type-pdv2, ..., gic-type-pdvn* que de points de vue dans lequel ce type d'entité est défini, puisque les représentations diffèrent d'un point de vue à l'autre (figure 13.1). Les attributs purement graphiques de ces sous-classes d'objets dépendront de la description de ce type d'entité dans ce point de vue (un attribut *longueur* n'étant par exemple pertinent que si la dimension du type dans le point de vue est un intervalle) (Cf. section 13.4.1). Deux attributs fondamentaux des classes du générateur d'interfaces cartographiques sont la liste des constituants et la liste de leurs positions.

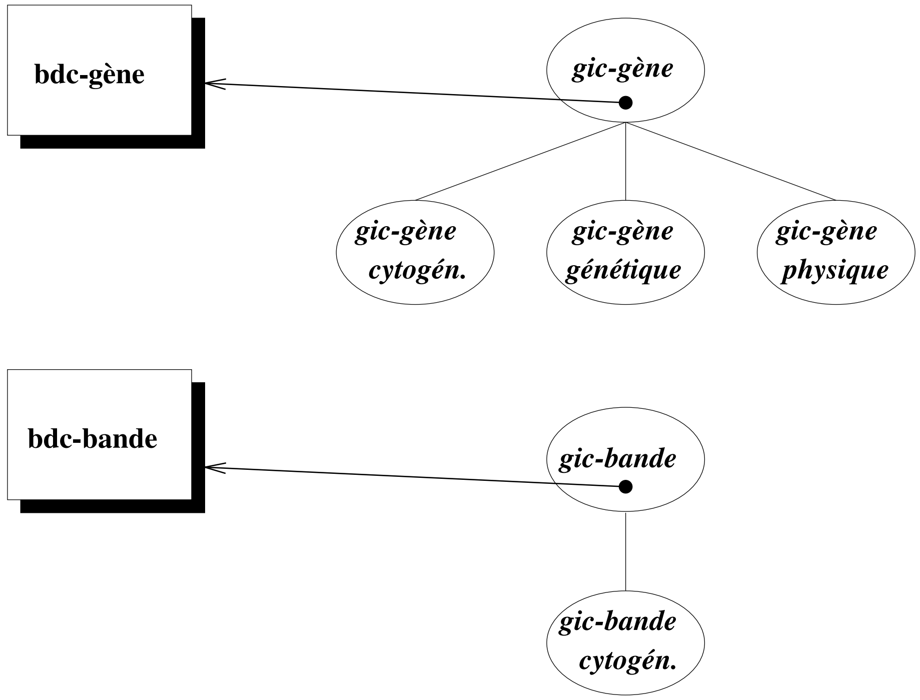


Figure 13.1 - : Liens entre classes de la base de connaissances et classes du générateur d'interfaces cartographiques. Chaque classe de la base de connaissances représentant un type correspond à une classe de l'interface et à autant de sous-classes de celle-ci qu'il y a de points de vue dans lesquels le type est visible. Ainsi, la classe *bdc-gène* a une hiérarchie de quatre classes associée, car le type gène apparaît dans les trois points de vue, tandis que la classe *bdc-bande* n'en a que deux.

Au niveau des instances, pour chaque instance *bdc-inst* de la base de connaissances est créée une instance de chacune des classes *gic-type-pdvi* (figure 13.2). Si ces instances sont des objets composites, elles pointent vers des ensembles d'instances de la base de connaissances, *a priori* différents puisque les décompositions peuvent différer d'un point de vue à un autre.

Les valeurs des attributs des classes du générateur d'interfaces cartographiques sont remplies de la manière suivante : chacun des trois attributs de ces classes est muni d'un accesseur, c'est-à-dire d'une fonction qui calcule sa valeur.

- L'attribut qui pointe vers la base de connaissances – et représenté avec un rond noir dans les figures précédentes – est rempli lors de l'instanciation d'une classe du générateur d'interfaces cartographiques, par la valeur de l'instance de la base de connaissances dont on souhaite visualiser la carte.

- L'attribut *l-composants* est muni d'un accesseur qui détermine les composants de l'instance de la base de connaissances dans le point de vue considéré qui ont une position donnée au sein de la carte (ce qui nécessite de chercher les relations d'ordre dans la base de connaissances), crée alors les instances du générateur d'interfaces

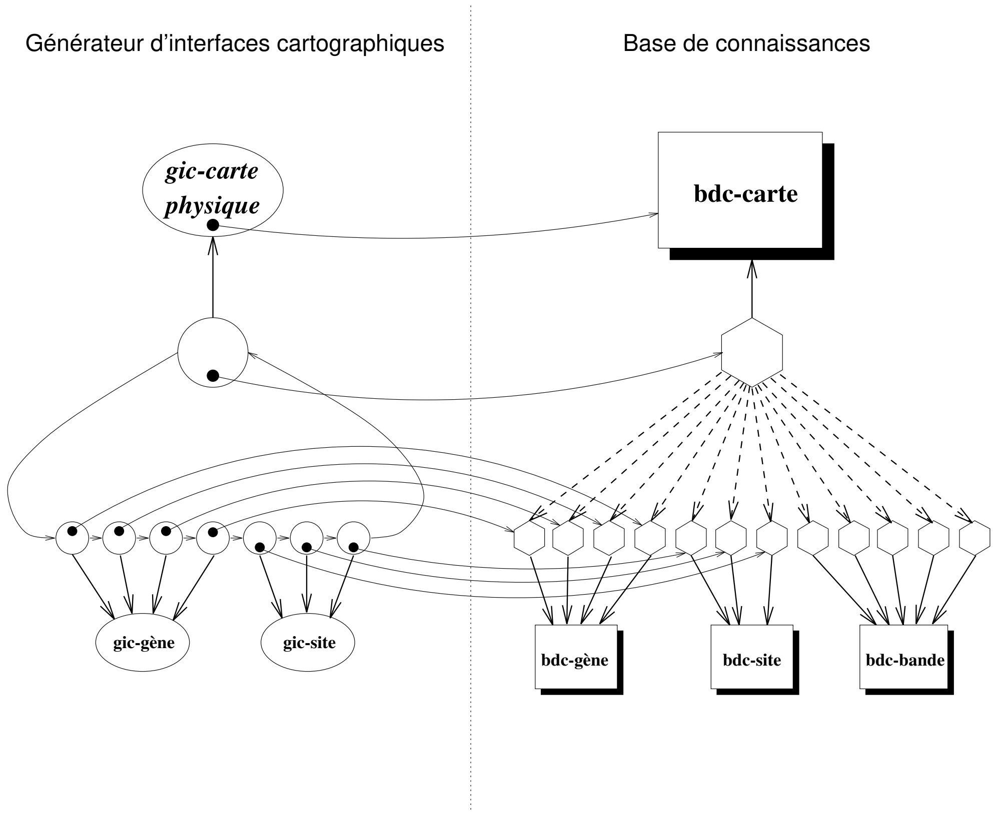


Figure 13.2 - : Instances de l'interface graphique. La demande de construction de la représentation graphique d'une carte, i.e. d'une instance composite *bdc-inst* d'une classe *bdc-type* de la base de connaissances, passe par la création d'une instance *gic-inst* de la classe d'objets graphiques associée à la classe *bdc-type*, puis par celles des instances *gic-insti* liées aux composants de *bdc-inst* dans le point de vue considéré ; sont éliminés en effet les composants non présents dans le point de vue (les composants *bande* sur la figure).

cartographiques associées à ces composants en remplissant leur attribut de position (dans cette carte-là).

- L'attribut *l-positions* est mis à jour lors du calcul précédent avec la position relative de chaque composant qu'on peut placer.

13.4 Fonctionnalités graphiques

Les fonctionnalités graphiques à apporter sont nombreuses ; on a vu combien les représentations cartographiques pouvaient être différentes. Ces fonctionnalités recouvrent à la fois la personnalisation de l'interface, comme la couleur et la forme des entités en fonction de leur type, l'apparence globale des cartes (horizontales ou verticales, graduées, etc.) et les comportements à proposer pour la manipulation des cartes (fonctions de «zoom», «scrollers», action des clics de la souris, etc.). Ces opérations ont lieu après la création des entités graphiques (section précédente) car elles en sont indépendantes ; c'est seulement l'apparence de l'interface qui est modifiée.

13.4.1 Cartes et entités

Afficher une carte, une fois qu'on connaît les composants et leur position, peut se faire de multiples manières ; la carte peut apparaître horizontale, verticale, avoir un axe gradué, une orientation, les entités peuvent se regrouper par type, avoir une couleur, une forme particulière, etc. La plupart des logiciels étudiés ne proposent qu'une seule façon d'afficher la carte, qu'un utilisateur ne peut pas modifier ; SIGMA est une exception, car il est possible d'agir sur les couleurs des entités, de définir des styles de carte qui restreignent les entités affichées à un sous-ensemble des types constitutifs, de montrer le nom des entités ou pas si on veut éviter de surcharger la carte, etc.

En ce qui concerne l'apparence des entités, il suffit de définir un tableau de couleurs et de formes indexé par le type de l'entité et le point de vue. Ainsi, une entité a sa couleur et sa forme déterminées par la valeur du tableau suivant son type et le point de vue dans lequel elle apparaît. Si pour des entités possédant une longueur, la forme est le plus souvent un rectangle, pour les entités ponctuelles, il est intéressant de disposer de formes multiples permettant de distinguer les types des entités. De cette manière, un site de restriction sur une carte physique et un gène sur une carte génétique peuvent avoir deux représentations différentes.

13.4.2 Comportements

L'interface cartographique doit permettre la définition de comportements standards sur les cartes telles que le «zoom», le «scrolling» ou la sélection d'entités. De plus, les entités elles-mêmes doivent avoir des comportements liés aux actions de la souris. Par exemple, on peut souhaiter qu'un double-clic sur une entité active sa représentation propre si c'est une entité composite ou la récupération d'informations textuelles si ce n'en est pas une. La définition de ces comportements est à la charge du développeur de l'interface qui les spécifie en fonction des actions d'un utilisateur final. Ils ne sont limités que par les possibilités du langage de génération d'interfaces graphiques utilisé.

Le générateur d'interfaces cartographiques n'a pas pour but de redéfinir des comportements qui sont déjà fournis par ce langage, mais de faciliter la construction des cartes en pourvoyant le développeur de l'interface d'outils adaptés à cette tâche. Cela correspond en l'occurrence à la visualisation des cartes à partir des informations nécessaires (composants et positions).

Chapitre 14

Spécifications fonctionnelles du générateur d'interfaces

Ce chapitre va présenter l'interface fonctionnelle du générateur d'interfaces cartographiques, c'est-à-dire les fonctions à utiliser pour décrire une interface cartographique. Comme cela a été vu au chapitre précédent, ces fonctions agissent sur divers aspects de l'interface, au niveau de sa spécification propre (définition du modèle lié aux fonctions *const*, *dim*, etc.) d'une part, au niveau de sa personnalisation graphique d'autre part.

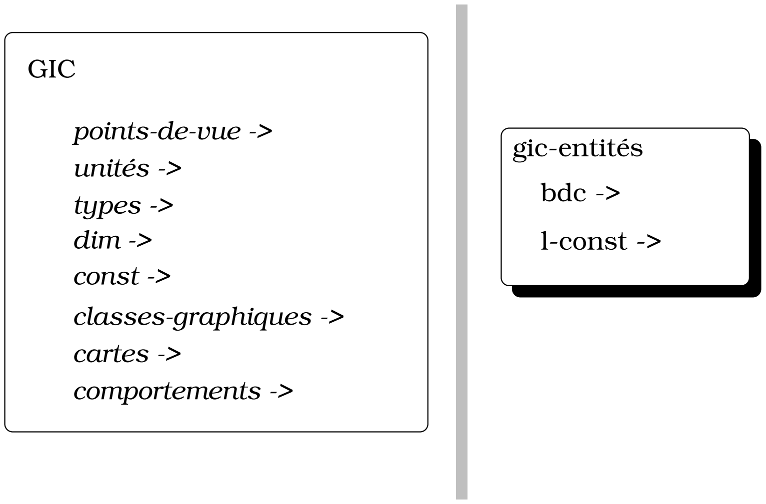
14.1 Initialisation des données

Les fonctions de cette section sont destinées à initialiser l'interface cartographique en cours de développement, en créant un objet spécifique, réceptacle de la connaissance sur le modèle des données et des liens vers la base de connaissances. Cet objet, vide au départ, est progressivement rempli par les appels fonctionnels suivants.

Les fonctions sont écrites dans une syntaxe issue de Lisp, mais ne sont bien sûr pas dépendantes de ce langage. La seule restriction faite est la disponibilité supposée d'un langage de programmation par objets ; elle n'est pas restrictive car la plupart des langages permettant de définir des interfaces sont basés sur la programmation par objets (jusqu'à la bibliothèque de fonctions X). Chacune des fonctions est adjointe d'un exemple correspondant au modèle du chapitre 3.

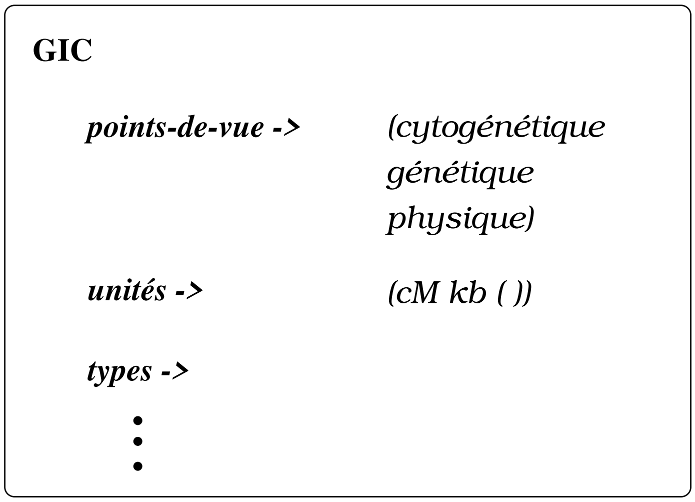
- *(créer_interface_carto <nom-interface>$_{chaine}$)* : cette fonction crée un objet vide contenant un certain nombre d'attributs dont la valeur sera donnée par la suite ; cet objet est le réceptacle de toute l'interface cartographique et sera le seul moyen d'accès aux entités de cette interface. La fonction crée également la classe racine des classes graphiques et les attributs *bdc* et *l-constituants*.

```
(créer_interface_carto "GIC")
```



- $(\mathit{maj\_pdvs}\ <\!\mathit{nom\text{-}interface}\!>_{interface}\ (<\!\mathit{points\text{-}de\text{-}vue}\!>_{chaine})^{+})$: cette fonction, et les suivantes mettent à jour les attributs de l'objet dépositaire de l'interface, avec les valeurs fournies.

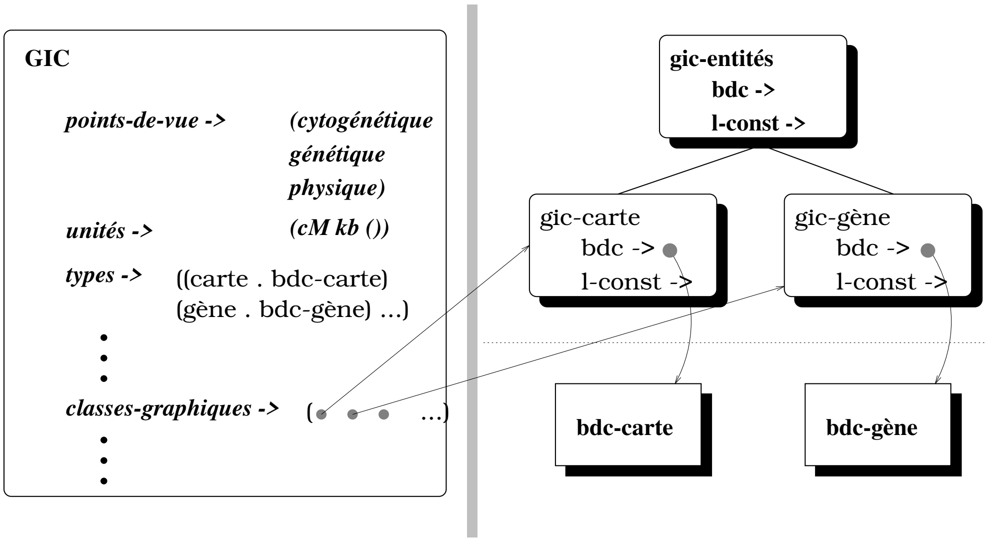
```
(maj_pdvs "GIC" ("cytogénétique"          (maj_unités "GIC"
        "génétique"  "physique"))            ("cM" "kb" ()))
```



- $(\mathit{maj\_unit\acute{e}s}\ <\!\mathit{nom\text{-}interface}\!>_{interface}\ (<\!\mathit{unit\acute{e}s}\!>_{chaine})^{+})$.

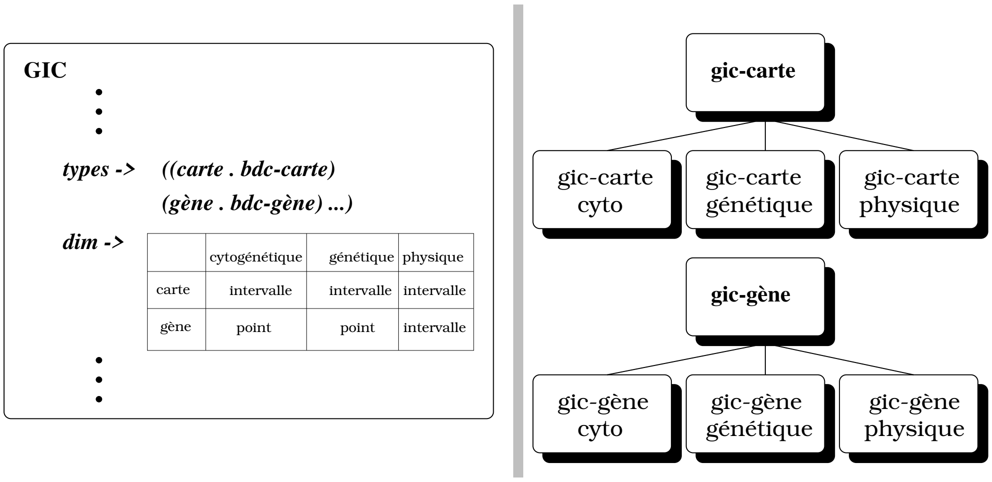
- $(\mathit{maj\_types}\ <\!\mathit{nom\text{-}interface}\!>_{interface}\ (<\!\mathit{types}\!>_{chaine}\ .\ <\!\mathit{bdc}\!>_{bdc\_type})^{+})$: cette fonction met à jour l'attribut *types* de l'objet cartographique, tout en faisant le lien avec la représentation dans la base de connaissances ; à tout type est associé un pointeur vers l'objet de la base de connaissances (classe de manière générale, ou concept dans le cas de TROPES) qui modélise ce type. Cette fonction crée aussi les classes graphiques correspondantes, et lie l'attribut *classes-graphiques* à ces classes.

```
(maj_types "GIC" (("carte" . bdc-carte) ("gène" . bdc-gène) ...))
```



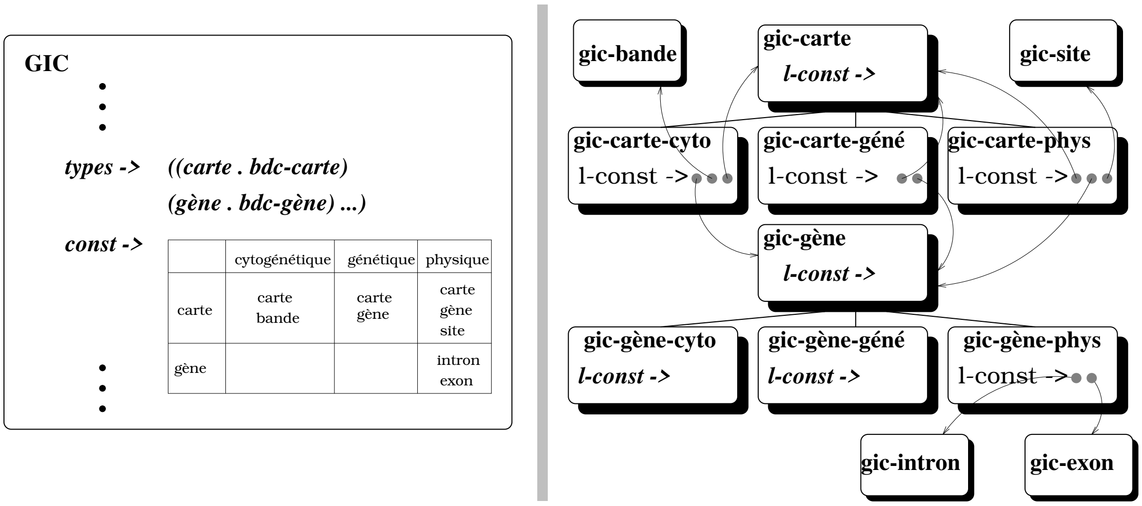
- *(maj_dim <nom-inter>$_{inter}$<nom-type>$_{type}$ (<nom-pdv>$_{pdv}$. <dimension>$_{dim}$)$^{+}$)* : *maj_dim* spécifie la pertinence des types dans les points de vue en donnant de plus leur dimension dans les points de vue où ils apparaissent, et agit sur les classes graphiques en créant des sous-classes des classes initiales à chaque fois qu'un type apparaît dans un point de vue.

```
(maj_dim "GIC" "carte"  ("cytogénétique" . "intervalle")
                        ("génétique" . "intervalle")
                        ("physique" . "intervalle")))

(maj_dim "GIC" "gène"  ("cytogénétique" . "point")
                       ("génétique" . "point")
                       ("physique" . "intervalle")))
```



- *(maj_const <nom-inter>$_{inter}$<nom-type>$_{type}$ (<nom-pdv>$_{pdv}$. (<nom-type>$_{type}$)$^{+}$)$^{+}$)* : comme la précédente, cette fonction complète l'objet cartographique en son attribut *const*, associant à chaque type ses types constitutifs dans chacun des points de vue. Elle a pour action de mettre à jour l'attribut *const* avec un tableau contenant ces informations, et de remplir l'attribut *l-const* des classes graphiques avec les types constitutifs.

```
(maj_const "GIC" "carte"   ("cytogénétique" . ("carte" "bande"))
                           ("génétique" . ("carte" "gène"))
                           ("physique" . ("carte" "gène" "site")))

(maj_const "GIC" "gène"    ("physique" . ("intron" "exon")))
```



Les classes graphiques qui ont été définies vont, en s'instanciant, créer des objets qui contiendront toutes les informations nécessaires pour s'afficher. Pour cela, il leur manque encore de nombreux éléments tant concernant la récupération des entités de la base de connaissances que de leur apparence à l'écran (section suivante 14.2). Retrouver les entités de la base de connaissances se fera grâce à un ensemble d'accesseurs associés aux attributs des classes graphiques. Pour être le plus générique possible, ces accesseurs sont définis dynamiquement par le développeur de l'interface cartographique, et utilisent des lambda-fonctions qui accèdent aux entités de la base de connaissances ; de cette manière, cette récupération n'est pas soumise au choix qui a été fait pour la représentation des données, mais est configurable. Ces accesseurs sont précisés maintenant.

- *def-accesseur-inst* : cette fonction doit permettre de trouver une instance de la base de connaissances à partir de la donnée de sa classe (ou de son concept) d'appartenance et d'arguments (qui ne se limitent *a priori* pas au seul nom).

- *def-accesseur-const* : cette fonction a pour but de récupérer les constituants d'une instance de la base de connaissances. Elle prend comme unique argument une telle instance et renvoie une liste d'instances de la base de connaissances.

- *def-accesseur-positions* : cet accesseur récupère l'ensemble des positions des constituants d'une entité composite à partir d'une instance composite de la base de connaissances.

- *def-accesseur-type* : accès au type d'une instance de la base de connaissances.

Ces quatre fonctions sont l'interface minimale à définir entre l'interface cartographique et la base de connaissances (ou de manière plus générale, les données sous quelque forme que ce soit).

Elles permettent, à partir d'une des classes graphiques définies précédemment, de construire des instances de ces classes qui seront directement affichables à l'écran. En effet, la chaîne qui lie une classe graphique *gic-type* à une instance complète de cette classe (c'est-à-dire dont les attributs ont reçu une valeur adéquate) est la suivante. D'abord, la valeur de l'attribut *bdc* de la classe donne la représentation dans la base de connaissances *bdc-type* ; on peut alors retrouver une instance de cette classe (ou concept) grâce à l'accesseur *accesseur-inst* ; ensuite, les composants de cette instance sont directement récupérés à l'aide de *accesseur-const*, de même que les positions par *accesseur-positions*. Il suffit pour finir de déterminer la classe de la base de connaissances à laquelle appartient chaque composant en utilisant *accesseur-type* et celle qui, parmi l'ensemble des classes graphiques, lui est associée grâce au pointeur inverse. On peut enfin créer les instances qui composent l'instance initiale dans l'interface.

14.2 Construction de l'image des cartes

Les classes d'objets graphiques, en plus des attributs qui précisent les constituants et leur position, disposent d'attributs purement graphiques concernant la couleur et la forme à prendre lors de leur affichage. Le premier attribut *couleur* est rajouté au niveau de la classe *gic-type*, car par défaut, toute instance d'un même type biologique a la même couleur ; néanmoins, il est possible de redéfinir cette couleur tant au niveau des sous-classes de points de vue qu'au niveau des instances elles-mêmes. L'attribut *forme* dépend quant à lui du point de vue, et sera donc valué dans les classes adéquates. Les objets-intervalles auront typiquement une forme rectangulaire, d'hauteur variable, les objets-points un tiret ou une flèche.

Deux fonctions permettent de mettre à jour ces attributs ; ce sont *maj_couleur*, qui prend comme arguments un type et une couleur et *maj_forme*, qui ajoute aux arguments précédents un point de vue.

Pour les objets composites, il est important de faire la distinction entre leur représentation graphique propre, c'est-à-dire en tant que carte sur laquelle apparaissent les composants, et celle en tant qu'entité d'une autre carte, et qui est donnée par ces attributs *couleur* et *forme*.

L'affichage d'une carte (ou plus généralement d'un objet composite) nécessite encore de nombreux éléments de spécification de cet affichage. Ceux-ci sont énumérés ci-dessous :

- axe de la carte : la carte est-elle affichée horizontalement ou verticalement ?
- taille de l'image ;
- graduation de l'axe : de tant à tant ;
- types visibles : par défaut, tous ;
- noms : les noms des entités constitutives sont-il affichés sur la carte, sont-ils affichés s'il ne dépassent pas le cadre qui les représente ?

- affichage des distances deux à deux : si la carte n'a pas la propriété d'additivité, affiche-t-on aussi les distances entre deux entités non consécutives ?

À partir de tous ces attributs, une instance de carte dispose de la totalité des informations nécessaires à son affichage. L'envoi du message *creer_carte* à une telle instance va assembler divers composants pour créer une image de la carte. Le développeur de l'interface cartographique a alors la charge d'encapsuler cette image dans une application plus grande, de la positionner dans un «scroller», de la juxtaposer à d'autres cartes, etc. Néanmoins, et ce sera le dernier point à aborder, cette carte doit pouvoir répondre à un certain nombre de messages qui en déterminent le comportement graphique. Ces messages et les comportements correspondants sont résumés dans le tableau 14.1.

Message	*Comportement*
zoom-in	Multiplication par un facteur constant
zoom-out	Multiplication par l'inverse
zoom souris	La zone décrite occupe tout l'espace
clic sur une entité	Sélection de cette entité (pour action ultérieure)
double-clic sur une entité	Éditeur d'instances ou appel de la création d'une carte s'il s'agit d'une entité composite (définissable par l'usager)
supprimer type	Suppression du type
affichage vertical	Reconstruction de la carte pour affichage vertical
affichage horizontal	Reconstruction de la carte pour affichage horizontal

Tableau 14.1 - : Actions des messages envoyés à une image de carte. Ces messages pour la plupart sont des méthodes de la classe, et modifient parfois les attributs qui caractérisent l'affichage. C'est le développeur de l'interface cartographique qui détermine certaines des actions à effectuer quand plusieurs sont possibles, comme par exemple celle d'un double-clic sur une entité.

14.3 Implémentation

L'implémentation du logiciel de cartographie a été confiée à un stagiaire CNAM, Alain Garreau. Celle-ci est malheureusement détachée des parties concernant la représentation et l'algorithmique des cartes génomiques. En effet, ces deux parties ont été traitées dans l'environnement Lisp, dont TROPES est dépendant. Par contre, tout le développement informatique sur l'interface cartographique a été réalisé en IlogViews, produit de la société Ilog dont le langage hôte est C++.

Il n'existe pas pour le moment de connexion entre Talk3.0 et IlogViews, même si elle est effectivement prévue à terme. Quand elle sera possible, il faudra bien évidemment lier la connaissance sur les cartes à leur représentation graphique.

L'intégration de deux sera sans doute même encore plus forte, puisque la société Ilog, dans le cadre du projet GREG dont nous avons déjà parlé, élabore un produit basé sur le modèle de représentation de connaissances TROPES, et qui servira à représenter l'ensembles des bases de connaissances biologiques utiles au projet.

Le développement du logiciel baptisé APIC (pour *Aide à la Présentation d'Interfaces Cartographiques*) est encore en cours. Dans l'état actuel des choses, il permet, à partir d'informations de distances entre entités de différentes cartes, d'afficher des cartes superposées. De nombreuses fonctions graphiques existent qui profitent de la présence d'entités communes sur plusieurs cartes ; par exemple, le défilement d'une carte «accrochée» à une autre conditionne le défilement de cette dernière, à l'aide d'un facteur d'échelle constant (donc approximatif), jusqu'à ce que une entité commune soit atteinte, auquel cas la seconde carte se recale par rapport à la première.

La figure 14.1 montre l'allure générale de l'interface cartographique.

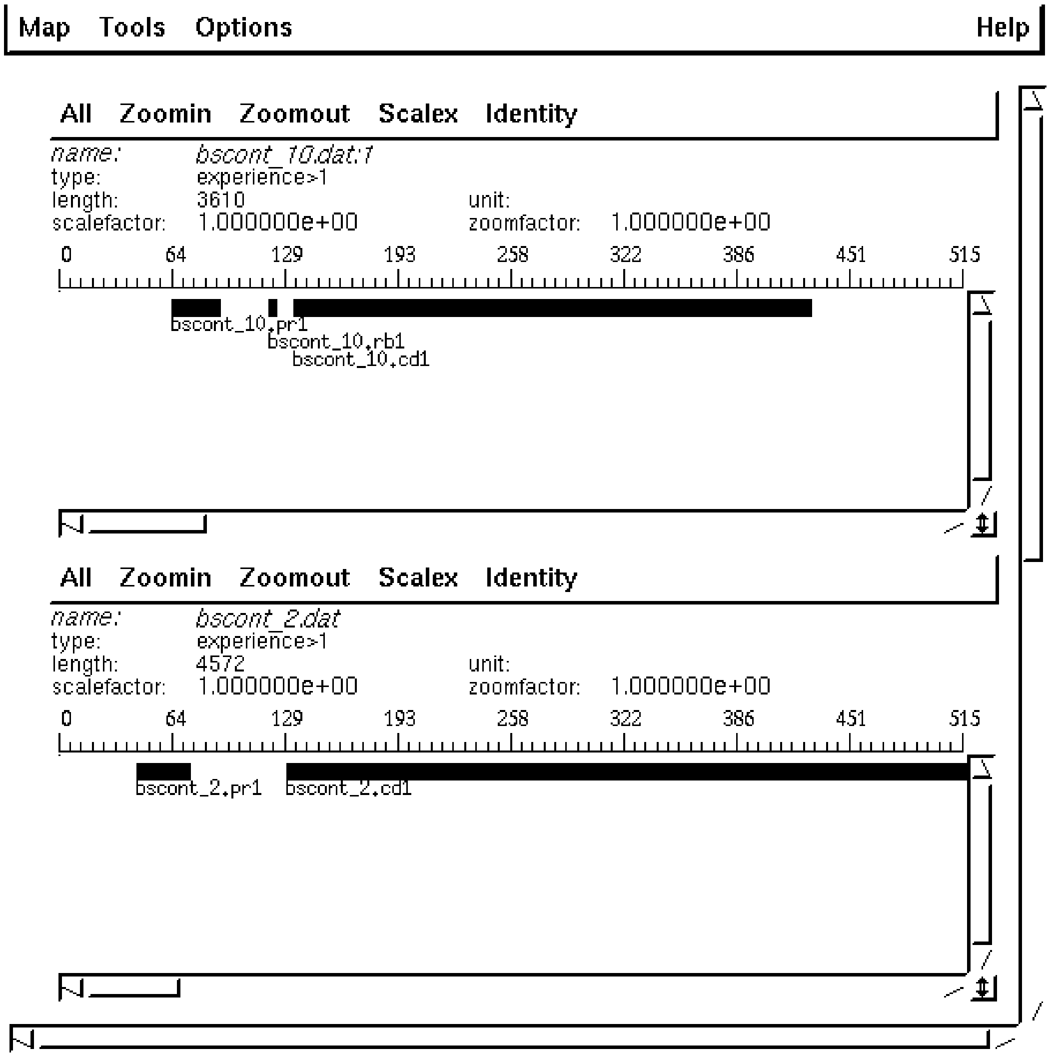


Figure 14.1 - : Interface cartographique provisoire du projet GREG. Plusieurs cartes superposées sont visibles sur cette copie d'écran ; chacune dispose d'un filtre permettant de ne garder que certains types d'entité. De plus, les cartes peuvent être liées grâce à des «hooks» qui conditionnent les défilements.

Conclusion et perspectives

Avant d'aborder les perspectives de ce travail, il est bon de rappeler brièvement les différents problèmes qu'il soulève et les solutions qu'il propose.

Cheminement du mémoire

Le point de départ de ce travail est la constatation du besoin de modéliser les cartes génomiques à la fois pour leur représentation et leur construction. La formalisation qui en a été faite a mis en évidence ces deux aspects, par la spécification détaillée des éléments formant le concept de carte génomique et des relations permettant la construction des cartes. Cette dernière a été réalisée en s'appuyant sur des travaux importants en raisonnement temporel, mais qui ont dû être adaptés aux caractéristiques des cartes pour aboutir à un algorithme à même d'ordonner les entités qui y apparaissent et de déduire les distances qui les séparent.

L'implémentation de la formalisation dans un système de représentation de connaissances a été facilitée par la richesse des mécanismes que ce système, TROPES, propose, et permet de conserver la déclarativité des descriptions des cartes, aussi bien sur le côté représentationnel que sur celui du raisonnement. Le résultat est une base de connaissances aisément extensible en ce qui concerne les deux aspects précédents.

Pour terminer, ce mémoire présente les fonctionnalités d'un générateur d'interfaces cartographiques, restant ainsi résolument du côté de la généricité, contrairement à (à notre connaissance) tous les autres logiciels de visualisation de cartes, même ceux qui intègrent des fonctionnalités de construction. Un tel générateur d'interfaces cartographiques est un aboutissement indispensable à la visualisation des cartes génomiques, représentées dans une base de connaissances, et construites grâce à l'algorithme proposé.

Ce travail a contribué à réunir, s'il en était encore besoin, la biologie moléculaire et l'informatique (et plus particulièrement la modélisation des connaissances) ; il a mis au service de la modélisation des cartes des outils de représentation de connaissances et de raisonnement. Ces outils ne sont pas un luxe inutile car la complexité du problème nécessitait effectivement des moyens de haut niveau pour permettre, sinon de le résoudre complètement, du moins de le caractériser et d'apporter des mécanismes partiels de résolution. De plus, de cette manière a pu être justifiée la nécessité de disposer de mécanismes de représentation évolués tels que ceux pourvus par TROPES.

Quoi qu'il en soit, un tel travail a pour vocation de s'intégrer à un cadre plus large et n'est pas une fin en soi. Le projet du GREG intitulé «Système coopératif pour l'analyse de séquences génomiques» constitue ce cadre englobant.

Vers un outil générique d'analyse de séquences

Le but du projet est le développement d'un environnement générique d'aide à l'analyse de séquences. Cette analyse consiste principalement, une fois qu'une séquence d'ADN a été obtenue, à la soumettre à des algorithmes de recherche d'entités particulières, la plupart du temps en mettant en évidence des régularités dans la suite de nucléotides. Le système SCARP [Willamowski94] permet de définir déclarativement des tâches de résolution, de les enchaîner, de les décomposer ; ce n'est donc pas simplement une boîte à outils remplies de méthodes d'analyse mais bien un environnement de résolution de problèmes en ce sens qu'il permet à un utilisateur de définir dynamiquement sa stratégie de résolution pour de nombreux problèmes. De plus, cet environnement est coopératif, car l'utilisateur peut à tout moment interrompre la résolution en cours, dont il suit le déroulement à l'écran, pour changer des paramètres, modifier le raisonnement, voire résoudre à la place de la machine les tâches pour lesquelles il se sent plus compétent.

SCARP repose sur le système de représentation de connaissances Shirka ; une tâche est un objet Shirka dont la résolution se fait en l'instanciant complètement grâce à un moteur d'inférence propre aux tâches, mais qui profite des fonctionnalités de Shirka, en particulier la classification. Néanmoins, il n'est pas envisagé de garder Shirka comme système de représentation de connaissances, en raison de ses limitations, mais plutôt de s'appuyer sur un système comme TROPES en y intégrant un modèle de résolution de problèmes s'apparentant à SCARP (Cf. §10.3.3).

Quel que soit le modèle utilisé, les entités mises en évidence par les méthodes d'analyse vont devoir être positionnées sur des cartes. C'est l'aboutissement du processus d'analyse, qui mène à une visualisation «classique» des résultats. C'est à ce moment-là qu'apparaissent le traitement des cartes et leur modélisation ; la structuration des résultats obtenus et la représentation de résultats partiels passent par la formalisation qui a été montrée. De plus, il subsiste de nombreux besoins de mécanismes de résolution : la construction des contigs en est un, il y en a d'autres moins liés à la séquence, mais dont l'importance est grande, comme cela a été vu précédemment. Pour finir, l'interface cartographique est l'outil indispensable à la visualisation des cartes ; devant la pléthore d'interfaces, il a paru nécessaire de dépasser la simple construction d'une nouvelle interface, forcément limitative, et développer plutôt un outil permettant de décrire ses propres interfaces personnalisées. L'intégration de tous ces éléments, un système de représentation de connaissances disposant de points de vue, de la relation de composition, d'un modèle de tâches, de contraintes, une implémentation dans ce système de la formalisation des cartes, le développement de méthodes d'analyse des séquences, la visualisation personnalisée de résultats à travers une interface cartographique, cette intégration donnera lieu à un système dont les biologistes moléculaires ont un besoin pressant et qui leur fait défaut.

Annexe A

Implémentation réalisée

Les développements informatiques, que ce soit au niveau de TROPES, des algorithmes de construction de cartes ou de l'interface cartographique, ont souffert pour différentes raisons.

A.1 Tropes

TROPES a été implémenté en Lisp, dans un langage développé par la société Ilog. Malheureusement, les versions de ce Lisp se sont succédées, et il a fallu régulièrement faire des traductions d'une version à une autre[1]. De plus, certaines des versions ont été utilisées en β-test, ce qui nous a obligés à résoudre des bugs du logiciel. Actuellement, le code est en cours de traduction dans la dernière version de Lisp, appelée *Talk3.0* ; cette traduction rencontre de multiples problèmes.

Néanmoins, il existe une version stable intégrant tout le modèle, une gestion des types, un langage de contraintes arithmétiques.

A.2 L'algorithme de construction de cartes

Celui-ci a été presque intégralement implémenté. Tout d'abord, il a fallu étendre le modèle et développer les concepts détaillés dans la formalisation (Cf. annexe B). Étant donné qu'il est apparu que l'utilisation du système de contraintes de TROPES était impossible car les contraintes qu'on a à traiter sont symboliques, nous avons récupéré un système de raisonnement temporel (dont les algorithmes sont de toute façon *a priori* plus efficaces que des algorithmes généraux de satisfaction de contraintes). Ce système s'appelle MATS et a été développé par Henry Kautz à AT&T ; il suit précisément les spécifications donné par lui-même et Peter Ladkin dans [Kautz et al.91]. En l'occurrence, il permet de traiter à la fois les relations qualitatives d'Allen grâce à l'algorithme de chemin-consistance et les relations quantitatives formalisées par Dechter *et al.* [Dechter et al.91] dans le cas des réseaux simples (à un seul intervalle).

Le code de MATS ayant été écrit en Common Lisp, il a fallu avant tout le transcrire en Le-Lisp pour des raisons de compatibilité avec TROPES. À part ces modifications, il

[1] Merci a Jérôme Euzenat de tout le travail qu'il a pu (dû !) effectuer à ce niveau-là.

n'a nécessité aucun remaniement. En l'état actuel, il existe un module Le-Lisp compilé des algorithmes de raisonnement temporel de MATS dont l'interface fonctionnelle est disponible de n'importe quel programme Le-Lisp, pour peu que le module soit chargé. L'algorithme de construction de cartes appelle un certain nombre des fonctions de cette interface.

Comment se fait alors le passage de l'information de TROPES à MATS et réciproquement ? Les relations exprimées dans le formalisme de TROPES sont traduites en expressions Lisp reconnues par MATS, de la forme `(nom-int-1 nom-relation nom-int-2)`, ceci au sein de chaque îlot. Ce formalisme de description est alors conservé tout au long de la résolution, même au niveau de la fusion des îlots. Pour le moment, il n'existe pas de traduction après la résolution pour repasser à la description des relations en TROPES. On obtient donc à la fin de la résolution un ensemble d'îlots contenant les relations valides en leur sein ou dans lesquels une incohérence a été trouvée.

A.3 L'interface cartographique

Dans ce cas aussi, des problèmes de compatibilité de langages ont fait que les développements actuels réalisés par Alain Garreau, dans le cadre d'un stage CNAM, ne sont pas fidèles aux spécifications décrites dans le chapitre 14.

L'implémentation actuelle est faite dans un langage de génération d'interfaces graphiques appelé *IlogViews*, en C++. Ce langage n'est lié qu'à la dernière version du Lisp dans laquelle la traduction du code de TROPES n'a pas encore été achevée. De plus, une version en Talk3.0 a récemment été livrée.

Annexe B

Extension du modèle Tropes

L'implémentation initiale du modèle a été faite par quelques-uns des thésards de l'équipe, après de nombreuses discussions sur son intérêt et les moyens de la réaliser. Il s'agit de Jérôme Gensel, Pierre Girard et moi-même. Des modifications *a posteriori* de cette implémentation ont eu lieu, et bien sûr des ajouts pour tenir compte des développements ultérieurs comme le système de gestion des types ou le traitement de contraintes.

L'extension du modèle TROPES a été réalisée en modifiant deux fichiers de construction des entités de ce modèle. Le premier fichier trctes.ll réunit toutes les déclarations des variables du modèle ; celles-ci sont donc étendues pour tenir compte des nouveaux éléments. Le second fichier trinit.ll remplit les champs de ces variables par les bonnes valeurs, en faisant les liens entre les nombreuses entités du méta-modèle ; ici aussi, les ajouts ont permis d'inclure les descriptions des nouvelles entités nécessaires à la modélisation des cartes génomiques.

Annexe C

Définition de nouveaux concepts dans Tropes

C.1 Le concept Relation

Le concept de relation est implémenté en tant que base TROPES ; il se présente donc sous la forme d'un fichier de description de base de connaissances et en respecte la syntaxe. Celle-ci se comprend d'elle-même, la lecture du fichier permettant de comprendre immédiatement les concepts représentés.

```
----------------------
 FICHIER RELATION.BDF
----------------------


<relation
    points-de-vue = {relation} ;
    concepts = {
             <relation
                    clefs = {nom-relation, entites} ;
                    attributs = {
                              <nom-relation
                                    dans    = chaine ;
                                    nature  = propriete ;
                                    const   = un ;
                              >,
                              <entites
                                    dans    = instance ;
                                    nature  = propriete ;
                                    const   = liste-de ;
                              >,
                              <distance
                                    dans    = distance ;
```

```
                    nature  = propriete ;
                    const   = ens-de ;
              >,
              <extremite1
                    dans    = chaine ;
                    nature  = propriete ;
                    const   = un ;
              >,
              <extremite2
                    dans    = chaine ;
                    nature  = propriete ;
                    const   = un ;
              >
                } ;
    points-de-vue = {
                  <relation
                    classe-racine =
                                  <relation
                                    attributs = {
                                        <nom-relation
                                            facettes = {
                    domaine = {"avant", "apres",
                     "recouvre-avant", "recouvre-apres",
                     "contient", "contenu-dans", "ordre",
                     "orien-oppo", "meme-orien",
                     "distance"} ;
                                                       } ;
                                        >,
                                        <entites
                                          facettes = {} ;
                                        >
                                                {};
                                    >;
                    classes = {
                            <relation-quali
                                    sorte-de = relation ;
                                    attributs = {
                                      <nom-relation
                                              facettes = {
                            domaine = {"avant", "apres",
                             "recouvre-avant",
                             "recouvre-apres", "contient",
                             "contenu-dans", "ordre",
                             "meme-orien","orien-oppo"} ;
                                                        };
```

```
          >
                    };
>,
<relation-quanti
        sorte-de = relation ;
        attributs = {
          <nom-relation
                  facettes =
{domaine = {"distance"} ;
};
          >,
          <distance
                facettes = {};
          >,
          <extremite1
                facettes =
{domaine = {"origine", "fin"};
};
          >,
          <extremite2
                facettes =
{domaine = {"origine", "fin"};
};
          >
                     };
>,
<relation-quali-ordre
   sorte-de = relation-quali ;
        attributs = {
          <nom-relation
                  facettes =
{domaine = {"avant", "apres",
 "recouvre-avant",
 "recouvre-apres"} ;
} ;
          >
                    };
>,
<relation-quali-sans
   sorte-de = relation-quali ;
        attributs = {
          <nom-relation
                  facettes =
{domaine = {"contient",
 "contenu-dans", "ordre",
```

```
                                     "meme-orien","orien-oppo"} ;
                                    } ;
                                              >
                                                        };
                                    >
                                      };
                             >
                               };
                passerelles = {};
                instances = {};
        >
         };
>
```

C.2 Le concept Distance

Le concept de distance est également implémenté en tant que base TROPES :le fichier suivant montre sa description.

```
 ----------------------
  FICHIER DISTANCE.BDF
 ----------------------

<distance
    points-de-vue = {distance} ;
    concepts = {
             <distance
                    clefs = {distance, incertitude, point-de-vue} ;
                    attributs = {
                              <distance
                                    dans    = reel ;
                                    nature  = propriete ;
                                    const   = un ;
                              >,
                              <incertitude
                                    dans    = reel ;
                                    nature  = propriete ;
                                    const   = un ;
                              >,
                              <point-de-vue
                                    dans    = chaine ;
                                    nature= propriete ;
                                    const   = un ;
                              >
```

```
            } ;
points-de-vue = {
              <distance
                classe-racine =
                              <distance
                                attributs = {
                                    <distance
                                      facettes = {} ;
                                    >,
                                    <incertitude
                                      facettes = {} ;
                                    >,
                                    <point-de-vue
                                      facettes = {} ;
                                    >
                                             };
                                >;
                                classes = {};
                >
                  };
passerelles = {};
instances = {
        <
        est-dans =
                { distance distance
                } ;
        distance = 9.1 ;
        incertitude = 0. ;
        point-de-vue = "genetique" ;
        >,
        <
        est-dans =
                { distance distance
                } ;
        distance = 3.2 ;
        incertitude = 0. ;
        point-de-vue = "genetique" ;
        >,
        <
        est-dans =
                { distance distance
                } ;
        distance = 8.5 ;
        incertitude = 3.1 ;
        point-de-vue = "genetique" ;
```

```
                               >
                                 };
              >
               };
>
```

Annexe D

Les concepts biologiques

Les concepts biologiques sont inclus dans un fichier de base de connaissances, qui décrit les différents points de vue, et les concepts mentionnés précédemment : intron, exon, gène, marqueur, site, bande et carte. Les liens avec les entités ajoutées dans le méta-modèle se font par l'intermédiaire de fonctions de mises à jour détaillées à la fin de cette section.

```
 -----------------
  FICHIER BIO.BDF
 -----------------

<biologie-moleculaire
    points-de-vue = {genetique, cytogenetique, physique} ;
    concepts = {
             <intron
                    clefs = {nom-intron} ;
                    attributs = {
                            <nom-intron
                                    dans    = chaine ;
                                    nature  = propriete ;
                                    const   = un ;
                            >
                                };
                    points-de-vue = {
                            <physique
                                    classe-racine =
                                            <intron
                                                    attributs = {
                                                        <nom-intron
                                                          facettes = {} ;
                                                        >
                                                                };
                                            >;
                                    classes = {};
```

```
                >
                    };
        passerelles = {};
        instances = {
                <
                est-dans =
                        { intron physique
                        } ;
                nom-intron = "intron1" ;
                >,
                <
                est-dans =
                        { intron physique
                        } ;
                nom-intron = "intron2" ;
                >
                   };
>,
<exon
        clefs = {nom-exon} ;
        attributs = {
                <nom-exon
                        dans    = chaine ;
                        nature  = propriete ;
                        const   = un ;
                >
                    };
        points-de-vue = {
                <physique
                        classe-racine =
                                <exon
                                        attributs = {
                                            <nom-exon
                                              facettes = {} ;
                                            >
                                                    };
                                >;
                        classes = {};
                >
                    };
        passerelles = {};
        instances = {
                <
                est-dans =
                        { exon physique
```

```
                        } ;
                nom-exon = "exon1" ;
                >,
                <
                est-dans =
                        { exon physique
                        } ;
                nom-exon = "exon2" ;
                >,
                <
                est-dans =
                        { exon physique
                        } ;
                nom-exon = "exon3" ;
                >
                   };
>,
<gene
        clefs = {nom-gene} ;
        attributs = {
                <nom-gene
                        dans    = chaine ;
                        nature  = propriete ;
                        const   = un ;
                >,
                <introns
                        dans    = intron ;
                        nature  = composant ;
                        const   = ens-de;
                >,
                <exons
                        dans    = exon ;
                        nature  = composant ;
                        const   = ens-de;
                >
                    };
        points-de-vue = {
                <cytogenetique
                        classe-racine =
                                <gene
                                        attributs = {
                                            <nom-gene
                                             facettes = {};
                                            >
                                                    };
```

```
                        >;
                classes = {};
        >,
        <genetique
                classe-racine =
                        <gene
                                attributs = {
                                    <nom-gene
                                     facettes = {};
                                    >
                                            };
                        >;
                classes = {};
        >,
        <physique
                classe-racine =
                        <gene
                                attributs = {
                                    <nom-gene
                                     facettes = {};
                                    >,
                                    <introns
                                     facettes = {};
                                    >,
                                    <exons
                                     facettes = {};
                                    >
                                            };
                        >;
                classes = {};
        >
            };
passerelles = {};
instances = {
        <
        est-dans =
                { gene cytogenetique,
                  gene genetique,
                  gene physique
                } ;
        nom-gene = "D1S10" ;
        >,
        <
        est-dans =
                { gene cytogenetique,
```

```
          gene genetique,
          gene physique
        } ;
nom-gene = "D1S14" ;
>,
<
est-dans =
        { gene cytogenetique,
          gene genetique,
          gene physique
        } ;
nom-gene = "D1S17" ;
>,
<
est-dans =
        { gene cytogenetique,
          gene genetique,
          gene physique
        } ;
nom-gene = "D1S19" ;
>,
<
est-dans =
        { gene cytogenetique,
          gene genetique,
          gene physique
        } ;
nom-gene = "D1S21" ;
>,
<
est-dans =
        { gene cytogenetique,
          gene genetique,
          gene physique
        } ;
nom-gene = "D1S39" ;
>,
<
est-dans =
        { gene cytogenetique,
          gene genetique,
          gene physique
        } ;
nom-gene = "PGM1" ;
>,
```

```
                <
                est-dans =
                        { gene cytogenetique,
                          gene genetique,
                          gene physique
                        } ;
                nom-gene = "AMY2B" ;
                >,
                <
                est-dans =
                        { gene cytogenetique,
                          gene genetique,
                          gene physique
                        } ;
                nom-gene = "TSHB" ;
                >
                   } ;
>,
<marqueur
        clefs = {nom-marqueur} ;
        attributs = {
                <nom-marqueur
                        dans    = chaine ;
                        nature  = propriete ;
                        const   = un ;
                >
                    };
        points-de-vue = {
                <genetique
                        classe-racine =
                                <marqueur
                                        attributs = {
                                            <nom-marqueur
                                              facettes = {} ;
                                            >
                                                    };
                                >;
                        classes = {};
                >
                        };
        passerelles = {};
        instances = {
                <
                est-dans =
                        {marqueur genetique
```

```
                        } ;
                nom-marqueur = "AMY2B" ;
                >
                   } ;
>,
<site
        clefs = {nom-site} ;
        attributs = {
                <nom-site
                        dans    = chaine ;
                        nature  = propriete ;
                        const   = un ;
                >
                    };
        points-de-vue = {
                <physique
                        classe-racine =
                                <site
                                        attributs = {
                                            <nom-site
                                              facettes = {} ;
                                            >
                                                  };
                                >;
                        classes = {};
                >
                    };
        passerelles = {};
        instances = {
                <
                est-dans =
                        { site physique
                        } ;
                nom-site = "XhoI" ;
                >,
                <
                est-dans =
                        { site physique
                        } ;
                nom-site = "NotI" ;
                >,
                <
                est-dans =
                        { site physique
                        } ;
```

```
                  nom-site = "NaeI" ;
                  >,
                  <
                  est-dans =
                          { site physique
                          } ;
                  nom-site = "SstII" ;
                  >,
                  <
                  est-dans =
                          { site physique
                          } ;
                  nom-site = "EagI" ;
                  >
                     };
  >,
  <bande
          clefs = {no-bande} ;
          attributs = {
                  <no-bande
                          dans    = chaine ;
                          nature  = propriete ;
                          const   = un ;
                  >,
                  <bandes
                          dans    = bande ;
                          nature  = composant ;
                          const   = ens-de ;
                  >
                      };
          points-de-vue = {
                  <cytogenetique
                          classe-racine =
                                  <bande
                                          attributs = {
                                              <no-bande
                                                facettes = {} ;
                                              >,
                                              <bandes
                                                facettes = {} ;
                                              >
                                                      };
                                  >;
                          classes = {};
                  >
```

```
                };
passerelles = {};
instances = {
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "13.1" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "13.2" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "13.3" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "13" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "22.1" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "22.2" ;
        >,
        <
        est-dans =
                { bande cytogenetique
                } ;
        no-bande = "22.3" ;
        >,
```

```
                <
                est-dans =
                        { bande cytogenetique
                        } ;
                no-bande = "22" ;
                >,
                <
                est-dans =
                        { bande cytogenetique
                        } ;
                no-bande = "21" ;
                >,
                <
                est-dans =
                        { bande cytogenetique
                        } ;
                no-bande = "31" ;
                >
                   };
>,
<carte
        clefs = {nom-carte} ;
        attributs = {
                <nom-carte
                        dans    = chaine ;
                        nature  = propriete ;
                        const   = un ;
                >,
                <cartes
                        dans    = carte ;
                        nature  = composant ;
                        const   = ens-de ;
                >,
                <genes
                        dans    = gene ;
                        nature  = composant ;
                        const   = ens-de ;
                >,
                <marqueurs
                        dans    = marqueur ;
                        nature  = composant ;
                        const   = ens-de ;
                >,
                <sites
                        dans    = site ;
```

```
                        nature  = composant ;
                        const   = ens-de ;
                >,
                <bandes
                        dans    = bande ;
                        nature  = composant ;
                        const   = ens-de ;
                >
                  };
        points-de-vue = {
                <genetique
                        classe-racine =
                                <carte
                                        attributs = {
                                            <nom-carte
                                              facettes = {} ;
                                            >,
                                            <cartes
                                              facettes = {} ;
                                            >,
                                            <genes
                                              facettes = {} ;
                                            >,
                                            <marqueurs
                                              facettes = {} ;
                                            >
                                                    };
                                >;
                        classes = {};
                >,
                <cytogenetique
                        classe-racine =
                                <carte
                                        attributs = {
                                                <nom-carte
                                                  facettes = {} ;
                                                >,
                                                <cartes
                                                  facettes = {} ;
                                                >,
                                                <genes
                                                  facettes = {} ;
                                                >,
                                                <bandes
                                                  facettes =
```

```
                                                              {} ;
                                                            >
                                                                };
                                            >;
                                    classes = {};
                            >,
                            <physique
                                    classe-racine =
                                            <carte
                                                    attributs = {
                                                        <nom-carte
                                                          facettes = {} ;
                                                        >,
                                                        <cartes
                                                          facettes = {} ;
                                                        >,
                                                        <sites
                                                          facettes = {} ;
                                                        >
                                                              };
                                            >;
                                    classes = {};
                            >
                                   };
                    passerelles = {};
                    instances = {
                            <
                            est-dans =
                                    {carte cytogenetique,
                                     carte genetique
                                    } ;
                            nom-carte = "chromosome21" ;
                            >
                               } ;
            >
              };
>


 --------------------
  FICHIER MAJ-BIO.LL
 --------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Ce fichier sert a mettre a jour les valeurs de l'attribut 'est-un'
```

```
;;; des concepts cartographiques en pointant sur les bonnes classes
;;; a savoir les sous-classes du concept 'concept' qui definissent les
;;; concepts ordonnes, et de meme avec les point de vue.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(in-package '#:tropes:init)


(setf
  (:val-est-un (tr-find-concept "carte"))
  (vector (static @co-concept@) (static @cl-concept-carto@)))

(setf
  (:val-est-un (tr-find-concept "gene"))
  (vector (static @co-concept@) (static @cl-concept-carto@)))

(setf
  (:val-est-un (tr-find-concept "marqueur"))
  (vector (static @co-concept@) (static @cl-concept-carto@)))

(setf
  (:val-est-un (tr-find-concept "site"))
  (vector (static @co-concept@) (static @cl-concept-carto@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "carte") "genetique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-intervalle@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "carte") "cytogenetique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-intervalle@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "carte") "physique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-intervalle@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "gene") "genetique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-point@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "gene") "cytogenetique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-point@)))

(setf
```

```
  (:val-est-un (tr-find-conceptview (tr-find-concept "gene") "physique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-intervalle@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "marqueur") "genetique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-point@)))

(setf
  (:val-est-un (tr-find-conceptview (tr-find-concept "site") "physique"))
  (vector (static @co-point-de-vue@) (static @cl-pdv-point@)))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Mise a jour des valeurs de l'attribut "unite" des points de vue des
;;; entites cartographiques.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(tr-set-value (tr-find-conceptview (tr-find-concept "intron") "physique")
  "unite" "Mb")

(tr-set-value (tr-find-conceptview (tr-find-concept "exon") "physique")
  "unite" "Mb")

(tr-set-value (tr-find-conceptview (tr-find-concept "gene") "physique")
  "unite" "Mb")

(tr-set-value (tr-find-conceptview (tr-find-concept "site") "physique")
  "unite" "Mb")

(tr-set-value (tr-find-conceptview (tr-find-concept "carte") "physique")
  "unite" "Mb")

(tr-set-value (tr-find-conceptview (tr-find-concept "gene") "genetique")
  "unite" "cM")

(tr-set-value (tr-find-conceptview (tr-find-concept "marqueur") "genetique")
  "unite" "cM")

(tr-set-value (tr-find-conceptview (tr-find-concept "carte") "genetique")
  "unite" "cM")

(tr-set-value (tr-find-conceptview (tr-find-concept "gene") "cytogenetique")
  "unite" "\% chr")
```

```
(tr-set-value (tr-find-conceptview (tr-find-concept "bande") "cytogenetique")
  "unite" "\% chr")
```

Bibliographie

[Allen83] J. F. Allen. – Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, vol. 26, n° 11, 1983, pp. 832–843.

[Altman et al.94] R. B. Altman, B. Weiser et D. J. States. – Constraint Satisfaction Techniques for Modeling Large Complexes: Applications to the Central Domain of 16S Ribosomal RNA. *Proc. of the 2^{nd} International Conference on Intelligent Systems for Molecular Biology*. Stanford, CA, pp. 10–18. – MIT Press, août 1994.

[Benne et al.92] R. Benne et H. van der Spek. – L'editing des messages génétiques. *La Recherche*, vol. 23, n° 245, juillet 1992, pp. 846–854.

[Blake et al.87] E. Blake et S. Cook. – On Including Part Hierarchies in Object-Oriented Languages, with an Implementation in Smalltalk. *European Conference on Object-Oriented Programming*, pp. 45–54. – Paris, France, 1987.

[Brachman et al.91] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick et A. Borgida. – Living with CLASSIC: When and How to Use a KL-ONE-Like Language. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, éd. par J. F. Sowa, pp. 401–456. – San Mateo, Morgan Kaufmann, 1991.

[Brachman79] R. J. Brachman. – On the Epistemological Status of Semantic Networks. *Associative Networks: Representation and Use of Knowledge by Computers*, éd. par N. V. Findler, pp. 3–50. – Orlando, Academic Press, 1979.

[Brachman83] R. J. Brachman. – What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*, vol. 16, n° 10, 1983, pp. 30–36.

[Capponi94] C. Capponi. – Exploitation des types dans un modèle de représentation de connaissances par objets. *RFIA '94*. Paris, France, pp. 171–183. – janvier 1994.

[Clark et al.94] D. A. Clark, C. J. Rawlings et S. Doursenot. – Genetic Map Construction with Constraints. *Proc. of the 2^{nd} International Conference on Intelligent Systems for Molecular Biology*. Stanford, CA, pp. 78–86. – MIT Press, août 1994.

[Cui94] Z. Cui. – Using Interval Logic for Order Assembly. *Proc. of the 2^{nd} International Conference on Intelligent Systems for Molecular Biology*. Stanford, CA, pp. 103–111. – MIT Press, août 1994.

[Danchin et al.84] A. Danchin et P. Slonimski. – Les gènes en morceaux. *La Recherche*, vol. 15, n° 155, mai 1984, pp. 616–626.

[Danchin93] A. Danchin. – Le séquençage des petits génomes. *La Recherche*, vol. 24, n° 251, février 1993, pp. 222–232.

[Dechter et al.91] R. Dechter, I. Meiri et J. Pearl. – Temporal constraint networks. *Artificial Intelligence*, vol. 49, n° 1-3, janvier 1991, pp. 61–95.

[DeLisi88] C. DeLisi. – Computers in Molecular Biology: Current Applications and Emerging Trends. *Science*, vol. 240, avril 1988, pp. 47–52.

[Donis-Keller et al.87] H. Donis-Keller, P. Green, C. Helms, S. Cartinhour, S. Weiffenbach et al. – A genetic linkage map of the human genome. *Cell*, vol. 51, octobre 1987, pp. 319–337.

[Doolittle85] R. Doolittle. – Les protéines. *Pour la Science*, no98, décembre 1985, pp. 54–65.

[Dorkeld94] F. Dorkeld. – *MultiMap : un modèle objet dédié à la cartographie comparée des génomes des mammifères.* – Lyon, France, Thèse de doctorat, Université Claude Bernard, Lyon 1, 1994.

[Douthart et al.94] R. J. Douthart, J. E. Pelkey et G. S. Thomas. – Database Integration and Visualization of Maps of the Human Genome Using the GnomeView Interface. *Proc. of the 27th Annual Hawaii International Conference on System Sciences*. pp. 49–57. – IEEE Computer Society Press, 1994.

[Durbin et al.92] R. Durbin et J. Thierry-Mieg. – *ACeDB - A C. Elegans Database I. Users' guide II. System Manual*, 1992.

[Emg94] The Jackson Laboratory. – *Encyclopedia of the Mouse Genome*, 1994. Disponible par ftp anonyme sur encyclo.jax.org.

[Erickson92] D. Erickson. – Le déchiffrage du génome humain. *Pour la Science*, no176, juin 1992, pp. 80–87.

[Euzenat93] J. Euzenat. – Représentation granulaire du temps. *Revue d'Intelligence Artificielle*, vol. 7, n° 3, 1993, pp. 329–361.

[Euzenat94] J. Euzenat. – *Granularité dans les représentations spatio-temporelles.* – Rapport technique n° RR-2242, Grenoble, France, INRIA, 1994.

[Freksa92] C. Freksa. – Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, vol. 54, n° 1-2, 1992, pp. 199–227.

[Frenkel91] K. A. Frenkel. – The Human Genome Project and Informatics. *Communications of the ACM*, vol. 34, n° 11, novembre 1991, pp. 41–51.

[Freuder78] E. C. Freuder. – Synthesizing constraint expressions. *Communications of the ACM*, vol. 21, n° 11, novembre 1978, pp. 958–966.

[Friedland et al.85] P. Friedland et L. H. Kedes. – Discovering the secrets of DNA. *Communications of the ACM*, vol. 28, n° 11, novembre 1985, pp. 1164–1186.

[Genographics93] Argonne National Laboratory. – *GenoGraphics for OpenWindow - v1.1 alpha*, 1993. Disponible par ftp anonyme sur info.mcs.anl.gov (140.221.10.1) dans /pub/GenoGraphics.

[Gensel et al.92] J. Gensel et P. Girard. – Expression d'un modèle de tâches à l'aide d'une représentation par objets. *Représentation Par Objets, RPO '92*. – La Grande-Motte, France, juin 1992.

[Gensel et al.93] J. Gensel, P. Girard et O. Schmeltzer. – Integrating constraints, composite objects and tasks in a knowledge representation system. *Proc. of the 5th IEEE International Conference on Tools with Artificial Intelligence*. Boston, MA, pp. 127–130. – novembre 1993.

[Gnomeview93] Pacific Northwest Laboratory, Richland, Washington. – *GnomeView - User's Manual/Tutorial*, 1993. Pour recevoir des informations, envoyer un courrier électronique à gv-request@gnome.pnl.gov.

[Golumbic et al.93] M. C. Golumbic et R. Shamir. – Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *Journal of the ACM*, vol. 40, n° 5, novembre 1993, pp. 1108–1133.

[Graves93] M. Graves. – Integrating Order and Distance Relationships from Heterogeneous Maps. *Proc. of the 1st International Conference on Intelligent Systems for Molecular Biology*. Washington, DC, pp. 154–162. – MIT Press, juillet 1993.

[Grivaud et al.92] Sylvain Grivaud et François Rechenmann. – Navigation dans les bases de connaissances associant objets et hypertexte. *Représentation Par Objets, RPO '92*. – La Grande-Motte, France, juin 1992.

[Group92] NIH/CEPH Collaborative Mapping Group. – A Comprehensive Genetic Linkage Map of the Human Genome. *Science*, vol. 258, octobre 1992, pp. 67–80.

[Guainville92] G. Guainville. – Le génome à portée de main. *La Recherche*, vol. 23, n° 249, décembre 1992, pp. 1438–1439.

[Guidi et al.93] J. N. Guidi et T. H. Roderick. – Inference of Order in Genetic Systems. *Proc. of the 1st International Conference on Intelligent Systems for Molecular Biology*. Washington, DC, pp. 163–169. – MIT Press, juillet 1993.

[Hayes-Roth et al.86] B. Hayes-Roth, B. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan et O. Jardetzky. – PROTEAN: Deriving Protein Structure from Constraints. *Proc. of the 5th National Conference on Artificial Intelligence, AAAI-86*, pp. 904–909. – Philadelphia, PA, 1986.

[Hearne et al.94] C. Hearne, Z. Cui, S. Parsons et S. Hajnal. – Prototyping a Genetics Deductive Database. *Proc. of the 2^{nd} International Conference on Intelligent Systems for Molecular Biology.* Stanford, CA, pp. 170–178. – MIT Press, août 1994.

[Hélène84] C. Hélène. – Les structures de l'ADN. *La Recherche*, vol. 15, n° 155, mai 1984, pp. 670–685.

[Holbrook et al.93] S. R. Holbrook, S. M. Muskal et S.-H. Kim. – Predicting Protein Structural Features With Artificial Neural Networks. *Artificial Intelligence and Molecular Biology*, éd. par L. Hunter, chap. 4, pp. 161–194. – AAAI Press, 1993.

[Honda et al.93] S. Honda, N. W. Parrot, R. Smith et C. Lawrence. – An Object Model for Genome Information at All Levels of Resolution. *Proc. of the 26^{th} Annual Hawaii International Conference on System Sciences.* pp. 564–573. – IEEE Computer Society Press, 1993.

[Hunter91] L. Hunter. – Artificial Intelligence and Molecular Biology. *AI Magazine*, vol. 11, n° 5, janvier 1991, pp. 27–36.

[Hunter93] L. Hunter. – Molecular Biology for Computer Scientists. *Artificial Intelligence and Molecular Biology*, éd. par L. Hunter, chap. 1, pp. 1–46. – AAAI Press, 1993.

[Jordan89] B. Jordan. – Les cartes du génome humain. *La Recherche*, vol. 20, n° 216, décembre 1989, pp. 1486–1494.

[Jordan92] B. Jordan. – *Voyage autour du génome.* – Éditions Universitaires, 1992.

[Kaplan et al.90] J.-C. Kaplan et M. Delpech. – *Biologie moléculaire et médecine.* – Médecine-Sciences Flammarion, 1990.

[Karp93] P. D. Karp. – A Qualitative Biochemistry and Its Application to the Regulation of the Tryptophan Operon. *Artificial Intelligence and Molecular Biology*, éd. par L. Hunter, chap. 8, pp. 289–324. – AAAI Press, 1993.

[Kautz et al.91] H. A. Kautz et P. B. Ladkin. – Integrating Metric and Qualitative Temporal Reasoning. *Proc. of the 9^{th} National Conference on Artificial Intelligence, AAAI-91*, pp. 241–246. – Anaheim, CA, 1991.

[Kim et al.89] W. Kim, E. Bertino et J. F. Garza. – Composite Objects Revisited. *Proceedings of the ACM/SIGMOD International Conference on the Management of Data.* Portland, pp. 337–347. – 1989.

[Koile et al.89] K. Koile et G. C. Overton. – A Qualitative Model for Gene Expression. *Proc. of the 1989 Summer Computer Simulation Conference.* Society for Computer Simulation, pp. 415–421. – juillet 1989.

[Kourilsky et al.84] P. Kourilsky et G. Gachelin. – L'organisation de l'information génétique. *La Recherche*, vol. 15, n° 155, mai 1984, pp. 642–651.

[Kourilsky90] P. Kourilsky. – *Les artisans de l'hérédité.* – Odile Jacob, 1990.

[Ladkin et al.92] P. Ladkin et A. Reinefeld. – Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, vol. 57, n° 1, 1992, pp. 105–124.

[Ladkin et al.94] P. B. Ladkin et R. D. Maddux. – On Binary Constraint Problems. *Journal of the ACM*, vol. 41, n° 3, mai 1994, pp. 435–469.

[Ladkin86] P. Ladkin. – Primitives and Units for Time Specification. *Proc. of the 5th National Conference on Artificial Intelligence, AAAI-86*, pp. 354–359. – Philadelphia, PA, 1986.

[Ladkin90] P. B. Ladkin. – *Constraint Reasoning With Intervals: A Tutorial, Survey and Bibliography.* – Rapport technique n° TR-90-059, Berkeley, CA, International Computer Science Institute, 1990.

[Lander et al.91] E. S. Lander, R. Langridge et D. M. Saccocio. – Computing in molecular biology: mapping and interpreting biological information. *IEEE Computer*, vol. 18, n° 11, novembre 1991, pp. 6–13.

[Lee et al.93] A. J. Lee, E. A. Rundensteiner, S. Thomas et S. Lafortune. – An Information Model for Genome Map Representation and Assembly. *Proc. of the 2nd ACM International Conference on Information and Knowledge Management, CIKM '93*, pp. 75–84. – novembre 1993.

[Letovsky et al.92] S. Letovsky et M. B. Berlyn. – CPROP: a rule-based program for constructing genetic maps. *Genomics*, vol. 12, 1992, pp. 435–446.

[Mackworth77] A. K. Mackworth. – Consistency in Networks of Relations. *Artificial Intelligence*, vol. 8, n° 1, 1977, pp. 99–118.

[Mariño et al.90] O. Mariño, F. Rechenmann et P. Uvietta. – Multiple Perspectives and Classification Mechanism in Object-Oriented Representation. *Proc. of the 9 th ECAI*, pp. 425–430. – Stockholm, 1990.

[Mariño91] O. Mariño. – Classification d'objets composites dans un système de représentation de connaissances multi-points de vue. *RFIA '91.* Lyon. – novembre 1991.

[Mariño93] O. Mariño. – *Raisonnement classificatoire dans une représentation à objets multi-points de vue.* – Grenoble, France, Thèse de doctorat, Université Joseph Fourier, Grenoble 1, 1993.

[Markowitz et al.92] J. A. Markowitz, J. T. Nutter et M. W. Evans. – Beyond Is-A and Part-Whole: More Semantic Network Links. *Computers Math. Applic.*, vol. 23, n° 6-9, 1992, pp. 377–390.

[Masini et al.91] G. Masini, A. Napoli, D. Colnet, D. Léonard et K. Tombre. – *Object Oriented Languages.* – San Diego, CA, Academic Press, 1991.

[Meiri91] I. Meiri. – Combining Qualitative and Quantitative Constraints in Temporal Reasoning. *Proc. of the 9th National Conference on Artificial Intelligence, AAAI-91*, pp. 260–267. – Anaheim, CA, 1991.

[Minsky75] M. Minsky. – A Framework for Representing Knowledge. *The Psychology of Computer Vision*, éd. par P.H. Winston, pp. 211–281. – MacGrawHill, 1975.

[Montanari74] U. Montanari. – Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science*, vol. 7, 1974, pp. 95–132.

[Morton88] N. E. Morton. – Multipoint mapping and the emperor's clothes. *Ann. Hum. Genet.*, vol. 52, 1988, pp. 309–318.

[Mouchiroud91] D. Mouchiroud. – The distribution of genes in the human genome. *Gene*, vol. 100, 1991, pp. 181–187.

[Nebel et al.93] B. Nebel et H. J. Brückert. – *Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra.* – Rapport technique n° 11, Kaiserslautern et Saarbrücken, Allemagne, DFKI GmbH, 1993.

[Nökel88] K. Nökel. – *Convex Relations between Time Intervals.* – Rapport technique n° 88-17, Saarbrücken, Allemagne, SEKI, Universität des Saarlandes, 1988.

[Overton et al.90] G. C. Overton, K. Koile et J. A. Pastor. – GeneSys: A Knowledge Management System for Molecular Biology. *Computers and DNA, SFI Studies in the Sciences of Complexity*, éd. par G. Bell et T. Marr, pp. 213–239. – Addison-Wesley, 1990.

[Patel-Schneider91] P. F. Patel-Schneider. – What's Inheritance Got to Do with Knowledge Representation. *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, éd. par M. Lenzerini, D. Nardi et M. Simi, pp. 1–11. – Chichester, West Sussex, John Wiley & Sons, 1991.

[Quickmap94] Généthon. – *QuickMap*, 1994. Disponible par ftp anonyme sur ceph-genethon-map.genethon.fr.

[Rawlings et al.94] C. J. Rawlings et J. P. Fox. – Artificial intelligence in molecular biology: a review and assessment. *Phil. Trans. R. Soc. London*, vol. 344, 1994, pp. 353–363.

[Rechenmann et al.91] F. Rechenmann et P. Uvietta. – Shirka - An Object-Centered Knowledge Base Management System. *Artificial Intelligence in Numerical and Symbolic Simulation.* – Lyon, France, ALEAS, 1991.

[Rechenmann92] F. Rechenmann. – Method Inheritance and Selection in Class-Based Knowledge Models. *ERCIM Workshop on Theoretical and Experimental Aspects of Knowledge Representation*, pp. 205–208. – Pise, Italie, 1992.

[Robbins92] R. J. Robbins. – Challenges in the Human Genome Project. *IEEE Engineering in Medicine and Biology*, vol. 11, n° 1, 1992, pp. 25–34.

[Rougeon86] F. Rougeon. – La diversité des anticorps. *La Recherche*, vol. 17, n° 177, mai 1986, pp. 680–689.

[Sigma92] Los Alamos National Laboratory. – *SIGMA - System for Integrated Genome Map Assembly*, 1992. Disponible par ftp anonyme sur atlas.lanl.gov.

[Steeg93] E. W. Steeg. – Neural Networks, Adaptative Optimization, and RNA Secondary Structure Prediction. *Artificial Intelligence and Molecular Biology*, éd. par L. Hunter, chap. 3, pp. 121–160. – AAAI Press, 1993.

[Stefik81] M. Stefik. – Planning with Constraints (MOLGEN: Part 1). *Artificial Intelligence*, vol. 16, 1981, pp. 111–139.

[Sulston et al.92] J. Sulston, Z. Du, K. Thomas et al. – The *C. Elegans* genome sequencing project: a beginning. *Nature*, vol. 356, 1992, pp. 37–41.

[Suzuki et al.89] D. T. Suzuki, A. J. F. Griffiths, J. H. Miller et R. C. Lemontin. – *Introduction à l'analyse génétique.* – Éditions Universitaires, 1989.

[vanBeek et al.90] P. van Beek et R. Cohen. – Exact and approximate reasoning about temporal relations. *Computational Intelligence*, vol. 6, 1990, pp. 132–144.

[vanBeek89] P. van Beek. – Approximation algorithms for temporal reasoning. *Proc. of the 11 th IJCAI*, pp. 1291–1296. – Detroit, MI, 1989.

[vanBeek90a] P. van Beek. – *Exact and approximate reasoning about qualitative temporal relations.* – Ontario, Canada, Thèse de PhD, University of Waterloo, 1990.

[vanBeek90b] P. van Beek. – Reasoning about Qualitative Temporal Information. *Proc. of the 8^{th} National Conference on Artificial Intelligence, AAAI-90*, pp. 728–734. – Boston, MA, 1990.

[Vaysseix92] G. Vaysseix. – Généthon : vers une carte intégrée du génome humain. *Pour la Science*, no176, juin 1992, p. 86.

[Vilain et al.86] M. Vilain et H. Kautz. – Constraint propagation algorithms for temporal reasoning. *Proc. of the 5^{th} National Conference on Artificial Intelligence, AAAI-86*, pp. 377–382. – Philadelphia, PA, 1986.

[Waterman89] M. S. Waterman. – Sequence Alignments. *Mathematical Methods for DNA Sequences*, éd. par M. S. Waterman, chap. 3, pp. 53–92. – CRC Press, 1989.

[Weissenbach et al.94] J. Weissenbach et A. de Chenay. – Le génome humain balisé par des microsatellites. *La Recherche*, vol. 25, n° 262, janvier 1994, pp. 84–85.

[White et al.88] R. White et J.-M. Lalouel. – La cartographie des chromosomes. *Pour la Science*, no126, avril 1988, pp. 26–34.

[Willamowski94] J. Willamowski. – *Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur.* – Grenoble, France, Thèse de doctorat, Université Joseph Fourier, Grenoble 1, 1994.

[Winston et al.87] M. E. Winston, R. Chaffin et D. Herrmann. – A Taxonomy of Part-Whole Relations. *Cognitive Science*, vol. 11, 1987, pp. 417–444.

[Woods91] W. A. Woods. – Understanding Subsumption and Taxonomy: A Framework for Progress. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, éd. par J. F. Sowa, pp. 45–94. – San Mateo, Morgan Kaufmann, 1991.

[Xgrail94] Oak Ridge National Laboratory. – *User's Guide to Grail and GenQuest*, 1994. Disponible par ftp anonyme sur arthur.emp.ornl.gov.

[Zhang et al.94] J. Zhang, J. Ostell et K. Rudd. – ChromoScope: A Graphic Interactive Browser for E. Coli Data Expressed in the NCBI Data Model. *Proc. of the 27th Annual Hawaii International Conference on System Sciences.* pp. 58–67. – IEEE Computer Society Press, 1994.

Index