Un cadre formel pour l'annotation experte des gènes eucaryotes

Sarah Djebali

THESE de DOCTORAT

Specialité : Informatique

présentée pour obtenir

le GRADE de DOCTEUR EN SCIENCES DE l'UNIVERSITE D'EVRY VAL D'ESSONNE

par

Sarah Djebali

Sujet :

Un cadre formel pour l'annotation experte des gènes eucaryotes

En vue de soutenir le 22 septembre 2006, devant le jury composé de :

M. Daniel Gautheret	Rapporteur
M. Grégory Kucherov	Rapporteur
Mme Alessandra Carbone	Examinatrice
M. Roderic Guigo	Examinateur
Mme Marie-France Sagot	Examinatrice
M. Franck Delaplace	Directeur de thèse
M. Hugues Roest Crollius	Directeur de thèse

Table des matières

Chapitre 1

Introduction

Les gènes sont les éléments du génome à partir desquels s'effectue la synthèse des protéines. Ils en constituent donc des élements fonctionnels particulièrement importants. Pendant de nombreuses années, les séquences des génomes n'étaient pas disponibles, et accéder à leurs gènes signifiait donc procéder au séquençage de copies d'ARN messagers (ARNm) : les ADN complémentaires (ADNc). Certains projets se sont alors engagés dans d'importants programmes de séquençage massif d'extrémités d'ADNC, aussi appelées ESTs (Expressed Sequence Tag), qui ne représentent donc que des séquences partielles d'ARNm (consortium I.M.A.G.E. [4]). De plus ce séquençage massif avait lieu au prix à la fois d'une faible qualité de séquence, mais aussi d'une extrême redondance. Unigene [73] a alors joué un rôle prépondérant dans la gestion de cette redondance, en se penchant sur le problème difficile d'assigner à chaque gène un ensemble de séquences d'ESTs.

L'ère du séquençage des génomes a alors débuté, avec le séquençage du premier organisme vivant, la bactérie *Haemophilus influenza*, en 1995 [118], puis celui du premier organisme eucaryote, la levure *Saccharomyces cerevisiae*, en 1996 [24]. Les génomes étaient accessibles, et il s'agissait désormais de localiser les gènes dans l'ADN génomique.

Etant données les tailles des génomes séquencés, la localisation des gènes ou *annotation de gènes* devait nécessairement comporter une phase automatique. Les premiers programmes d'annotation sont alors apparus, et se sont focalisés sur le problème d'identifier les exons des gènes. Parmi ceux-là, citons le programme précurseur Grail, fondé sur l'utilisation de réseaux de neurones [137]. Puis les programmes se sont penchés sur le problème de l'annotation des gènes, comme par exemple le célèbre et efficace programme Genscan [44], qui utilise un modèle de Markov pour combiner différentes caractéristiques structurelles des gènes, comme l'utilisation préférentielles de certains codons ou encore la présence de signaux d'épissage à la frontière exon-intron.

Toutefois, la façon la plus fiable d'annoter les gènes dans une séquence d'ADN génomique, est de comparer (ou *aligner*) cette dernière à l'ensemble des séquences d'ADNc de l'organisme correspondant. Cependant cette stratégie nécessite très souvent l'intervention de l'humain, à

la fois du fait de la faible qualité des séquences d'ESTs, mais aussi de leur caractère partiel et très redondant. L'alignement des séquences d'ESTs avec l'ADN du génome correspondant, a également mis à jour le mécanisme d'*epissage alternatif*, prépondérant chez les eucaryotes supérieurs. Celui-ci conduit à la synthèse de plusieurs ARNm différents, et donc de plusieurs protéines différentes, à partir d'un même gène. L'existence de ce mécanisme complique beaucoup la prédiction de gènes dans les génomes eucaryotes, et l'alignement de séquences d'ESTs, et plus généralement d'ADNc, reste aujourd'hui le seul moyen de le prendre en compte.

L'estimation du nombre de gènes chez l'humain témoigne également du problème de l'annotation des gènes. En effet ce chiffre n'a cessé de fluctuer au cours du temps. En 2000, des estimations empiriques non publiées, souvent fondées sur les séquences d'ESTs, annonçaient un nombre de gènes humains voisin de 100000. A cette époque une équipe de chercheurs du Génoscope a eu l'idée de comparer des séquences du génome humain à celles d'un poisson en cours de séquençage au Génoscope : *Tetraodon nigroviridis*. Cette comparaison leur a permi d'estimer le nombre de gènes humains à un chiffre compris entre 28000 et 34000, ce qui était donc bien différent des estimations de l'époque [53] ! Cette estimation a été confirmée par la suite avec la publication de la première séquence du génome humain en 2001 [64].

L'annotation des gènes est la première étape après le séquençage du génome. Elle est fondamentale pour tenter de comprendre le fonctionnement de l'organisme, pour l a recherche médicale sur les maladies génétiques, mais aussi pour comprendre l'évolution. Nous nous intéressons ici à l'annotation des gènes dans les génomes eucaryotes. Cette tâche se révèle ardue pour plusieurs raisons. Tout d'abord la structure des gènes eucaryotes est complexe et très variable, à la fois au sein d'une espèce mais aussi entre les espèces. D'autre part les parties codantes des gènes n'y représentent souvent qu'une très faible proportion (1 à 2% du génome représentent des exons chez les mammifères). Il faut également mentionner la quantité parfois non négligeable de pseudogènes dans les génomes, pseudogènes qui sont des copies non fonctionnelles de gènes mais dont la composition peut être très voisine de celle des gènes fonctionnels, et qui peuvent être localisés au voisinage voire au sein-même de gènes fonctionnels. Enfin les mécanismes d'épissage et de début de traduction alternatifs doivent être pris en compte, et annoter un gène signifie donc localiser tous ses transcrits alternatifs.

Plusieurs méthodes ont été proposées pour annoter les gènes :

- les méthodes *ab initio*, qui se fondent sur un apprentissage à partir de séquences codantes connues,
- les méthodes par *similarité de séquence*, qui consistent à comparer la séquence à annoter à d'autres séquences biologiques,
- les méthodes *combinantes*, qui combinent les sorties de plusieurs programmes d'annotation.

Bien que ces méthodes se soient beaucoup améliorées au cours du temps, la référence actuelle en matière d'annotation reste l'expert humain. Annoter les gènes d'un génome est en effet une tâche critique puisqu'elle sert couramment de point de départ à des expériences biolo-

giques longues et délicates. De ce fait l'annotation des gènes d'une séquence doit garantir que toutes les séquences codantes de tous les transcrits codants de tous les gènes soient exacts. Vega (VErtebrate Genome Annotation [31]) est une base de données qui centralise l'ensemble des annotations de gènes des génomes de vertébrés dans le monde. Toutes ces annotations sont générées par des experts humains sur la base d'analyses longues et méthodiques de séquences d'ADN. Elles sont actuellement reconnues comme étant *les* annotations de référence.

Le meilleur annotateur étant actuellement l'expert humain, nous avons cherché à automatiser son protocole d'annotation. Pour annoter les gènes d'une séquence d'ADN, l'expert utilise des résultats de programmes d'annotation ou *ressources*, qui sont typiquement des alignements de séquences mais qui peuvent aussi être des résultats de programmes *ab initio*. Sa tâche consiste alors à parcourir la séquence à annoter, gène par gène, et pour chacun d'entre eux à analyser les ressources relatives à ce gène. Cette analyse consiste en la combinaison des ressources relatives à ce gène afin de reconstituer les différents transcrits alternatifs codants auxquels il peut donner lieu.

Pour passer des alignements de séquences aux transcrits, l'expert utilise des *règles heuristiques* fondées sur ses connaissances à la fois des ressources dont il dispose mais aussi des mécanismes de la transcription et de la traduction. Nous avons remarqué une double action particulièrement importante dans son protocole : il s'agit de la mise en relation d'ensembles d'alignements de séquences ou *transmods*, suivie du regroupement de certains d'entre eux sur la base des relations précédemment établies et qui témoignent de leur appartenance à la même structure de transcrit.

Pour automatiser cette double action nous avons choisi d'utiliser des multi-graphes orientés acycliques colorés ou *DACMs* (Directed Acyclic Coloured Multigraphs), dont les sommets sont des transmods (ensemble d'alignements de séquences) et dont les multiples couleurs d'arêtes sont les multiples relations que l'expert juge pertinent d'établir entre ces transmods. Ainsi l'étape de mise en relation de transmods effectuée par l'expert se modélise par une étape d'ajout de couleurs d'arêtes sur le DACM dont les sommets sont ces transmods, et l'étape de regroupement de certains transmods sur la base des relations qui les lient par une étape de parcours et de réduction de ce DACM selon certaines couleurs d'arêtes.

L'utilisation de constructions et de réductions de DACMs est au coeur d'un programme que nous avons développé pour automatiser le protocole d'annotation de l'expert : Exogean (Expert on gene annotation). Ainsi Exogean prend en entrée la séquence à annoter et les deux ressources considérées comme les plus fiables par l'expert : des alignements avec des ADNc de la même espèce et avec des protéines d'une autre espèce. Exogean agit sur ces alignements grâce aux mêmes règles heuristiques que l'expert humain et rend en sortie les transcrits codants alternatifs et les séquences codantes associées (CDS), des gènes de la séquence à annoter.

Exogean a participé au concours EGASP'05. EGASP'05 est lié au projet ENCODE dont le but est d'identifier tous les éléments fonctionnels contenus dans 44 séquences représentant 1% génome humain et appelées *régions ENCODE*. EGASP'05 a été organisé dans le double but d'évaluer les programmes d'annotation de gènes actuels, et de déterminer l'exhaustivité des annotations manuelles. En effet des annotateurs humains avaient au préalable procédé à une annotation méticuleuse des transcrits alternatifs des gènes des 44 régions ENCODE, et les annotations manuelles de 13 régions ENCODE parmi les 44 ont été mises à la disposition des participants quelque temps avant le concours. Les participants devaient alors soumettre leurs prédictions sur les 31 régions restantes avant une date donnée.

Dans la tâche essentielle de prédire au moins un transcrit codant par gène exactement (gène exact), Exogean a été classé premier sur vingt programmes devant Ensembl [54], Jigsaw [28] et Pairagon. De plus la spécificité d'Exogean en gène exact devance de beaucoup celle des autres programmes, et en particulier celles des cinq premiers programmes du concours (80% pour Exogean contre 68% pour le deuxième programme). Ce résultat montre que coder l'expertise humaine est aujourd'hui la meilleure façon, mais surtout la façon la plus fiable, d'annoter les gènes automatiquement. Cette méthode a de plus l'avantage de la **traçabilité**, puisque toutes les décisions prises au cours du processus d'annotation peuvent être retracées et expliquées. Ainsi un expert qui souhaiterait utiliser les résultats d'Exogean comme fondement de son annotation, gagnerait un temps précieux par rapport à l'utilisation de simples alignements de séquences.

Combiner les règles heuristiques utilisées par l'expert sur les alignements de séquences pose toutefois de nombreux problèmes, indiqués dans le tableau 1.1. L'expertise humaine en annotation nécessite donc un cadre formel d'expression à la fois :

- extrêmement flexible,
- expressif, explicatif et intuitif,
- générique et incrémental.

La double action de mise en relation et de regroupement de transmods évoquée plus haut est en réalité récurrente dans le protocole d'annotation de l'humain. Comme nous la modélisons par un ajout de couleurs d'arêtes sur un DACM suivie de sa réduction, il est commode de considérer que la réduction d'un DACM génère un nouveau DACM, puisque ce dernier peut alors à son tour subir une étape d'ajout de couleurs d'arêtes et de réduction. Les sommets du DACM obtenu par réduction d'un autre DACM, appelé DACM réduit, sont issus de groupes de sommets du DACM initial, et représentent donc des objets biologiques (transmods) «plus complexes» que les sommets du DACM initial.

Un langage dont une commande prend en entrée un DACM, le réduit selon la présence de certaines couleurs d'arêtes, et rend en sortie le DACM réduit sur lequel des couleurs d'arêtes ont été ajoutées, nous a semblé un cadre formel particulièrement bien adapté aux contraintes décrites ci-dessus, ainsi qu'à la façon dont l'expert humain procède. En effet un tel langage est à la fois flexible, expressif et générique et si un programme dans ce langage est une suite

Problème posé par le codage de l'expertise	**Nécessité pour le système de codage**
les règles heuristiques peuvent changer	le système d'annotation automatique doit être capable de modifier, d'ajouter ou de supprimer des règles
l'expert n'a pas l'habitude de formaliser les règles heuristiques qu'il utilise	le système doit permettre une expression aisée de ces règles
les règles heuristiques sont hétérogènes puisqu'elles peuvent porter sur la mise en relation ou le regroupement d'alignements de séquence, porter sur des propriétés structurelles ou de priorité des différents types d'alignements de séquences	le système doit être à la fois générique et incrémental

TAB. 1.1 – **Problèmes posés par le codage de l'expertise.**

ordonnée de commandes, il est l'expression d'une cascade de réductions et d'ajouts de couleurs d'arêtes sur des DACMs, autrement dit d'une cascade de regroupements et de mises en relation de transmods. Un programme dans ce langage modélise donc parfaitement le coeur de tout protocole d'annotation expert par mise en relation et regroupement d'alignements de séquences. Ce langage est nommé DACMLang.

Le présent manuscrit se décompose de la façon suivante :
- le chapitre 2 décrit l'état de l'art dans le domaine de l'annotation automatique de gènes eucaryotes,
- le chapitre 3 décrit le protocole et les règles heuristiques utilisées par les annotateurs humains, et la façon dont nous avons choisi de modéliser cette démarche. Les différentes étapes du programme Exogean y sont décrites,
- le chapitre 4 présente le langage DACMLang, dont une commande modélise l'unité de base du protocole d'annotation de l'humain : la mise en relation et le regroupement de transmods,
- le chapitre 5 présente les performances du programme Exogean évalué en référence aux annotations manuelles, et en particulier celles obtenues au concours EGASP'05,
- le chapitre 6 décrit l'état actuel de l'implantation d'Exogean,
- le chapitre 7 conclut le manuscrit et apporte différentes perspectives à ce travail.

Chapitre 2

Etat de l'art

Les méthodes d'annotation ont d'abord porté sur les organismes procaryotes, car ce sont les premiers à avoir été séquencés. Comme les gènes procaryotes ne comportent pas d'intron, ils peuvent se définir comme une suite de codons différents du codon stop, et suffisamment longue pour ne pas être apparue par hasard. Une telle région est appelée cadre ouvert de lecture ou ORF pour *Open Reading Frame*. Chez les eucaryotes, les gènes se composent d'une succession d'exons et d'introns (figure 2.1), ainsi la recherche d'ORF ne suffit pas à les annoter mais permet d'identifier les exons individuellement. Pour cette raison, nous nous intéressons d'abord à l'identification des exons, puis à l'annotation des gènes.

2.1 Identifier les exons

Les méthodes qui permettent d'identifier les exons se divisent en deux catégories : les méthodes *ab initio*, fondées sur un apprentissage à partir de séquences connues, et les méthodes *par similarité de séquence* fondées sur une comparaison de la séquence à annoter à d'autres séquences biologiques.

2.1.1 Méthodes *ab initio*

Les méthodes *ab initio* qui identifient les exons d'une séquence d'ADN s'appuient sur des mesures préliminaires de deux types : les *mesures par contenu*, qui analysent le contenu statistique global d'une séquence, le plus souvent pour déterminer si celle-ci est codante, et les *mesures par signal* qui sont plus locales et qui visent à identifier des séquences d'ADN de taille fixe représentant des *signaux fonctionnels* de l'ADN. Des exemples de signaux fonctionnels sont les sites donneur et accepteur d'épissage, ou encore les signaux d'initiation et de terminaison de la traduction (voir figure 2.1).

Mesures par contenu

Les mesures par contenu visent à déterminer le *potentiel codant* d'une séquence d'ADN, c'est-à-dire la vraisemblance que cette séquence code pour une protéine. Dans le cas où la

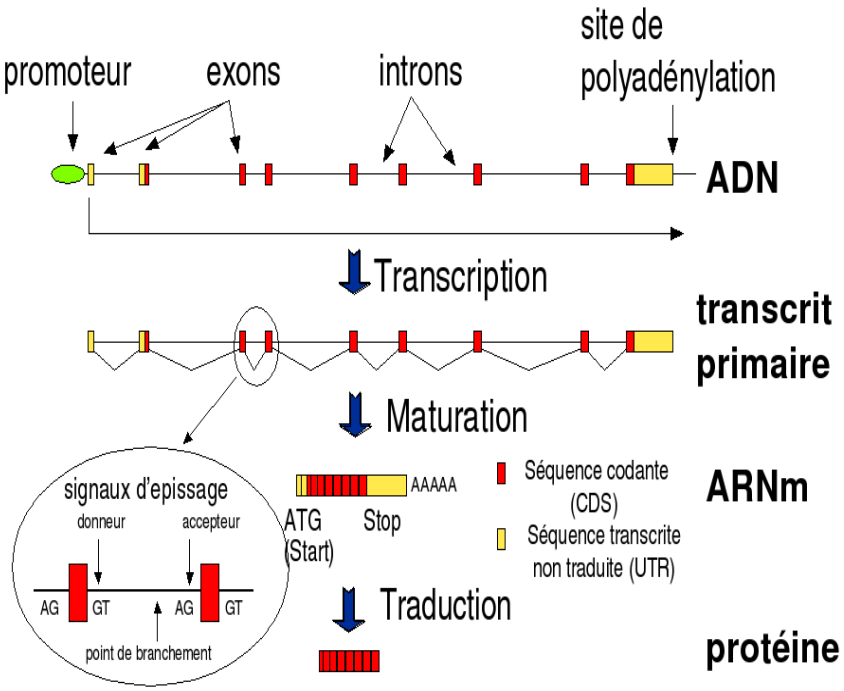


FIG. 2.1 – **Synthèse d'une protéine à partir d'un gène eucaryote.** Cette figure illustre les mécanismes cellulaires permettant la synthèse d'une protéine à partir d'un gène eucaryote. La partie supérieure du schéma comporte une séquence d'ADN génomique eucaryote comportant un gène. En amont (5') de celui-ci se trouve un promoteur, puis une succession d'exons et d'introns et enfin (3') un site de polyadénylation. Cette structure de gène subit d'abord un mécanisme de transcription qui a lieu de 5' en 3' et qui génère un transcrit primaire. Celui-ci a exactement la même structure que le gène présent sur l'ADN mais est dépourvu de promoteur. Il faut noter la présence de signaux d'épissage aux extrémités des introns de la structure de gène : un signal donneur d'épissage, le plus généralement GT, à l'extrémité 5' de l'intron, et un signal accepteur d'épissage, le plus généralement AG, à l'extrémité 3' de l'intron. Une étape de maturation du transcrit primaire donne alors lieu à un ARN messager (ARNm). Cette étape de maturation inclut une sous-étape d'épissage des introns du transcrit primaire. L'épissage consiste en la reconnaissance et l'excision des introns du transcrit primaire (ici matérialisée par des traits en V) et en le recollage des exons présents de part et d'autre des introns épissés. L'ARNm est donc la concaténation des exons de la structure de gène initiale. Au sein de cet ARNm seule une sous-séquence est alors traduite en protéine : c'est la séquence codante (CDS). Les extrémités 5' et 3' de l'ARNm non incluses dans le CDS sont appelées séquences UTR (UnTranslated Regions) 5' et 3'. Sur la séquence d'ADN initiale on a donc trois types d'exons : des exons totalement codants, des exons totalement non codants (UTR) et des exons partiellement codants.

séquence analysée est codante, ces mesures permettent aussi d'obtenir la phase de codage. Parmi les mesures par contenu, certaines utilisent un modèle d'ADN codant, d'autres pas. Utiliser un modèle d'ADN codant signifie réaliser un apprentissage à partir de données codantes vérifiées expérimentalement, et ainsi considérées comme des données de référence.

Les mesures qui n'utilisent pas un modèle d'ADN codant se fondent sur l'hypothèse que l'ADN codant est moins aléatoire ou plus homogène que l'ADN non codant. Parmi ces mesures nous pouvons citer la *mesure d'assymétrie dans la composition en bases entre les différentes positions du codon* [67], ou la *mesure de Fourier* [136]. Toutefois, les données codantes étant de plus en plus nombreuses dans les bases de données, il devient aujourd'hui de plus en plus simple de construire un modèle d'ADN codant. La plupart des mesures par contenu actuelles utilisent donc un tel modèle.

Les mesures par contenu se divisent en deux étapes :

1. une étape d'*apprentissage* à partir de données codantes et non codantes de référence, qui permet de déterminer le modèle codant et le contre-modèle non codant,
2. une étape de *test* sur une séquence d'ADN inconnue, qui permet de déterminer les sous-séquences codantes qu'elle contient.

Comme la partie codante d'un gène (CDS) est traduite en protéine codon par codon (figure 2.2), les mesures par contenu analysent successivement et indépendemment la séquence d'ADN dans les trois phases de lecture possible.

Définition 2.1.1 (S en phase p)
Soit $S = S_1 \ldots S_n$ une séquence de nucléotides ($n \geq 3,\ S_i \in \Sigma_{nt}$), et p un entier de $\{1, 2, 3\}$, S en phase p, noté $S_{phase\ p}$, est la séquence S privée de ses $p-1$ premières lettres :

$$S_{phase\ p} = S_p \ldots S_n$$

De manière similaire, pour un nucléotide S_i de S ($i \geq 1$), la phase de S_i dans S est définie par :

$$phase_S(S_i) = (i\ mod\ 3) + 1$$

Les mesures par contenu parcourent successivement les trois séquences $S_{phase\ p}$, $p \in \{1, 2, 3\}$, par fenêtre de taille N glissante par pas de longueur $Step$, afin de déterminer les régions codantes qu'elles contiennent. Ainsi, chaque séquence $S_{phase\ p}$ est décomposée en sous-séquences $s^{p,i}$ de taille N glissantes par pas de longueur $Step$ (la borne supérieure des indices i dépend de $\mid S \mid$, p et N, elle ne peut donc pas être spécifiée ici), comme illustré en figure 2.3.

Le score donné par une mesure par contenu pour une sous-séquence $s^{p,i}$ de taille N, s'obtient alors par un rapport de logvraisemblance :

$$score(s^{p,i}) = log\left(\frac{p_{codant}(s^{p,i})}{p_{non\ codant}(s^{p,i})}\right)$$

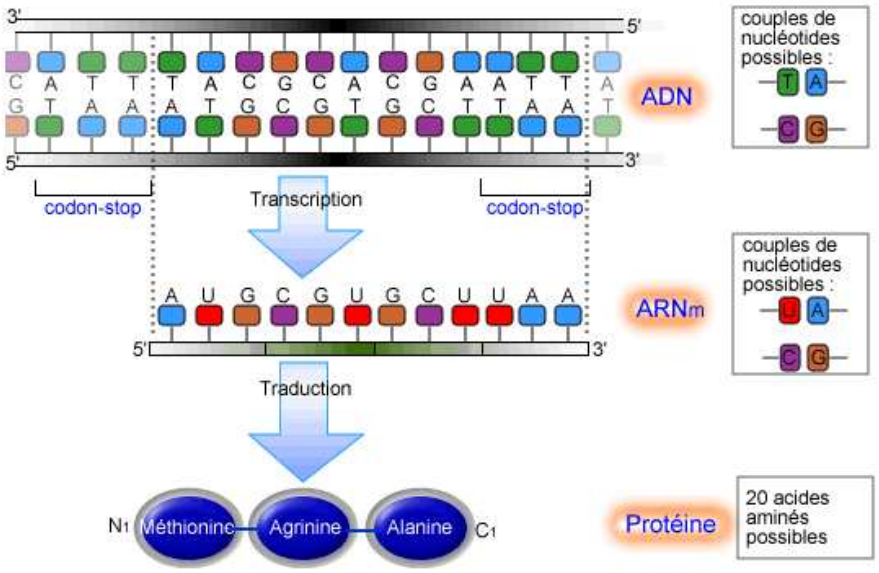


FIG. 2.2 – **Synthèse d'acides aminés à partir d'une région d'ADN codante.** Cette image représente une portion de molécule d'ADN double brin. La région d'ADN comprise entre deux codons stop est transcrite en ARNm, qui est lui-même traduit en protéine. Plus précisément chaque codon (triplet de nucléotides) de l'ARNm donne lieu à un acide aminé (code génétique).

où p_{codant} est la fonction de probabilité d'être codant, et $p_{non\ codant}$ est la fonction de probabilité d'être non codant. $score(s^{p,i}) > 0$ signifie ainsi que la sous-séquence $s^{p,i}$ a plus de chance d'être codante que non codante, et $score(s^{p,i}) < 0$ signifie le contraire.

Intervient alors le paramètre k. En effet les mesures par contenu se fondent sur l'idée que pour un entier k donné, les différents mots de longueur k, aussi appelés *k-mères*, n'apparaissent pas à la même fréquence dans les séquences codantes et dans les séquences non codantes. Pour $k = 3$ par exemple, où un k-mère correspond à un codon, le raisonnement est le suivant : étant donnée la dégénérescence du code génétique (plusieurs codons codent pour le même acide aminé), il est prévisible que les fréquences des codons dans les régions codantes soient différentes des fréquences des codons dans les régions non codantes. Cette supposition a été confirmée et se généralise en réalité à d'autres valeurs que $k = 3$.

Définition 2.1.2 (Décomposition en k-mères de S)
Soit $S = S_1 \ldots S_N$ une séquence de N nucléotides, telle que N est multiple de k. La décomposition de S en k-mères est donnée par :

$$S = X_1 \ldots X_{N/k}$$

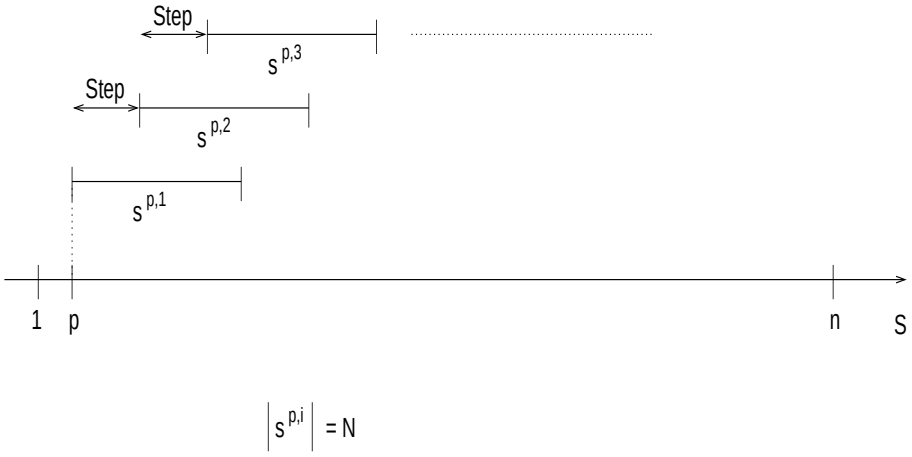


FIG. 2.3 – **Décomposition en fenêtres glissantes de taille N.** Cette figure représente une séquence d'ADN S orientée de 5' en 3'. Au dessus de cette séquence ont été représentées les différentes fenêtres de taille N glissantes par pas de $Step$ de la séquence S en phase p (segments horizontaux). Ces fenêtres sont des sous-séquences de $S_{phase\ p}$ de taille N dont la première débute au début de $S_{phase\ p}$ (donc en position p de S) et qui se chevauchent par pas de $Step$. La i-ème de ces séquences est nommée $s^{p,i}$.

où $X_j \in \Sigma_{nt}, j \in [1 : N/k]$, le j-ème k-mère de S, est donné par :

$$X_j = S_{(j-1)\times k+1} \ldots S_{j\times k}$$

Ainsi pour une séquence $s^{p,i}$ de taille N multiple de k, on a :

$$s^{p,i} = X_1^{p,i} \ldots X_{N/k}^{p,i}$$

où $X_j^{p,i}$ est le j-ème k-mère de $s^{p,i}$.

Pour poursuivre le calcul de $score(s^{p,i})$, il est nécessaire de distinguer le type de mesure par contenu. Il en existe principalement deux :

1. l'*utilisation des k-mères*,
2. le *modèle de Markov d'ordre* $(k-1)$ *et de période* 3, qui est une généralisation de la mesure précédente (d'où le $(k-1)$).

L'utilisation des k-mères. L'utilisation des k-mères est l'une des premières mesures par contenu à avoir été introduite. Sa phase d'apprentissage consiste en le calcul des fréquences d'apparition des différents k-mères possible dans des séquences codantes connues (modèle codant), et dans des séquences non codantes connues (contre-modèle non codant). La séquence à annoter est ensuite décomposée en k-mères dans les trois phases de lecture possible, et la

phase de test de l'utilisation des k-mères consiste en la comparaison de la composition en k-mères de ces trois séquences au modèle codant et au contre-modèle non codant.

L'utilisation des k-mères fait l'hypothèse de l'indépendance des k-mères adjacents. Ainsi nous obtenons les formules suivantes :

$$\begin{cases} p_{codant}(s^{p,i}) & = & \prod_{j=1}^{N/k} p_{codant}(X_j^{p,i}) \\ p_{non\ codant}(s^{p,i}) & = & \prod_{j=1}^{N/k} p_{non\ codant}(X_j^{p,i}) \end{cases}$$

où les fonctions $p_{codant}, p_{non\ codant} : \Sigma_{nt}^k \mapsto [0,1]$ s'obtiennent par apprentissage.

En réalité pour le codant on estime individuellement chacune des valeurs $p_{codant}(X_j^{p,i})$, ce qui correspond à un nombre de paramètres égal à $card(\Sigma_{nt}^k) = 4^k$. Pour le non codant on suppose en revanche l'indépendance des nucléotides adjacents, et une equiprobabilité de $1/4$ pour chaque nucléotide, ce qui donne :

$$\forall X \in \Sigma_{nt}^k,\ p_{non\ codant}(X) = (1/4)^k$$

En pratique les valeurs $k = 3$ et $k = 6$ sont les plus utilisées, et correspondent respectivement aux mesures d'utilisation des codons et d'utilisation des héxamères. De plus, il a été montré que la valeur $k = 6$, c'est-à-dire la mesure d'utilisation des héxamères, était celle qui discriminait au mieux le codant du non codant [51]. Elle est utilisée dans les programmes Grail [137], Geneparser [129, 130] et MZEF [146].

Le modèle de Markov d'ordre $(k-1)$ et de période 3. Comme l'utilisation des k-mères, le modèle de Markov d'ordre $(k-1)$ et de période 3 suppose l'indépendance des k-mères adjacents. Le modèle de Markov d'ordre $(k-1)$ et de période 3 représente une mesure plus fine que l'utilisation des k-mères car il fait aussi l'hypothèse qu'au sein de chaque k-mère, le dernier nucléotide dépend des $(k-1)$ nucléotides qui le précèdent (d'où le $(k-1)$). Ainsi on a :

$$\begin{cases} p_{codant}(s^{p,i}) & = & \prod_{j=1}^{N/k} p_{codant}(X_j^{p,i}(k), X_j^{p,i}(1)...X_j^{p,i}(k-1)) \\ p_{non\ codant}(s^{p,i}) & = & \prod_{j=1}^{N/k} p_{non\ codant}(X_j^{p,i}(k), X_j^{p,i}(1)...X_j^{p,i}(k-1)) \end{cases}$$

où $X_j^{p,i}(m)$ désigne le m-ième nucléotide de $X_j^{p,i}$, et $p(a,b)$ la probabilité conditionnelle de a sachant b.

Un modèle de Markov de période 3 distingue la phase du dernier nucléotide de chaque k-mère du modèle codant. Ainsi la formule permettant de calculer $p_{codant}(s^{p,i})$ utilise trois fonctions de probabilité conditionnelles, une pour chaque phase (on parle parfois de modèle de Markov «inhomogène»). Nous obtenons donc les formules suivantes :

$$\begin{cases} p_{codant}(s^{p,i}) & = & \prod_{j=1,\ phase_S(X_j^{p,i}(k))=l}^{N/k} \left(p^l{}_{k,codant}(X_j^{p,i}(k), X_j^{p,i}(1)...X_j^{p,i}(k-1)) \right) \\ p_{non\ codant}(s^{p,i}) & = & \prod_{j=1}^{N/k} \left(p_{non\ codant}(X_j^{p,i}(k), X_j^{p,i}(1)...X_j^{p,i}(k-1)) \right) \end{cases}$$

où $p^l_{codant}, p_{non\ codant} : \Sigma_{nt} \times \Sigma_{nt}^{k-1} \mapsto [0,1]$ pour $l \in \{1,2,3\}$, s'obtiennent par apprentissage.

Les valeurs des fonctions $p^l_{k,codant}$, paramètres du modèle codant, s'obtiennent en calculant des effectifs de mots dans les séquences d'apprentissage :

$$p^l_{k,codant}(N_k, N_1 \ldots N_{k-1}) = \frac{Nombre(N_1 \ldots N_k,\ phase_S(N_k) = l)}{Nombre(N_1 \ldots N_{k-1},\ phase_S(N_{k-1}) = (l-1)\ mod\ 3)} \quad (1)$$

Ainis le nombre de paramètres à estimer pour le modèle codant est $3 \times card(\Sigma_{nt} \times \Sigma_{nt}^k) = 3 \times 4 \times 4^{k-1} = 3 \times 4^k$, donc 3 fois supérieur à celui de l'utilisation des k-mères. Cela est la contrepartie du fait que le modèle de Markov d'ordre $(k-1)$ et de période 3 représente un modèle plus fin que l'utilisation des k-mères. Quant aux paramètres du modèle non codant, ils sont en général estimés sur des séquences non codantes connues.

Plus k est grand plus le modèle de Markov d'ordre $(k-1)$ est précis. Toutefois le nombre de paramètres à estimer est exponentiel en k ($\theta(4^{k+1})$), et le nombre de données d'apprentissage est par définition borné. Par conséquent les valeurs utilisées pour k restent limitées. Ceci se constate en pratique puisque les valeurs de k les plus utilisées sont $k = 5$, dans Genscan [44] et Genmark.hmm [98], et $k = 2$, dans Genscan, correspondant respectivement à un raffinement de l'utilisation des héxamères et de l'utilisation des codons.

La formule (1) montre que pour une bonne estimation d'un paramètre P du modèle de Markov d'ordre $(k-1)$ et de période 3, il est nécessaire que les mots intervenant dans le calcul de P soient présents en nombre suffisant dans les séquences d'apprentissage, ce qui n'est pas toujours le cas. Pour résoudre ce problème on utilise un modèle de Markov d'ordre $(k-1)$ et de période 3 dit *interpolé*, et noté IMM pour «Interpolated Markov Model». Le principe de l'IMM est d'estimer chaque paramètre P grâce à une combinaison de modèles de Markov d'ordre $k < k_P$ où k_P dépend de l'abondance des mots nécessaires à l'estimation de P dans les séquences d'apprentissage (formule (1)). Plus ces mots sont abondants plus k_P est élevé. Le premier programme à avoir utilisé l'IMM de période 3 est le programme Glimmer [58] dans le cadre de l'annotation des gènes microbiens. Ce sont ensuite les programmes GlimmerM [125], Augustus [133] et Eugene [126] qui l'on utilisé pour l'annotation de gènes eucaryotes.

Les modèles Markoviens peuvent être raffinés en utilisant des paramètres différents selon le pourcentage en (G+C) des séquences à analyser. Toutefois cela nécessite un apprentissage différent pour chacune des fenêtres du pourcentage en (G+C).

Etant donnée une séquence d'ADN S, les mesures par contenu représentent le potentiel codant de S par trois courbes : une pour chaque séquence $S_{phase\ p}$, $p \in \{1,2,3\}$. Pour un p donné, la courbe de $S_{phase\ p}$ a pour abscisse la position dans $S_{phase\ p}$, et pour ordonnée la valeur $score(s^{p,i})$ pour la fenêtre $s^{p,i}$ encadrant cette position. Comme $score(s^{p,i}) > 0$ indique la présence d'une région codante en phase p, il suffit pour localiser les exons codants

en phase p de S, de repérer les parties positives de la courbe de $S_{phase\ p}$.

Plus la taille N de la fenêtre d'analyse est petite, moins les mesures par contenu sont précises. Chez certaines espèces eucaryotes, de mammifères en particulier, la taille moyenne des exons est petite (comparée par exemple à celle des introns), ce qui peut poser des problèmes pour identifier les exons. Une solution serait l'utilisation d'une taille de fenêtre N petite, mais cela génèrerait aussi une imprécision quant aux *bornes* des exons codants identifiés. Comme des signaux spécifiques se situent aux bornes des exons codants, les mesures par contenu sont souvent combinés à des mesures par signal.

Mesures par signal

Les mesures par signal visent à identifier des signaux fonctionnels de l'ADN. En effet la principale difficulté de la prédiction de gènes eucaryotes provient de leur organisation en une succession d'exons et d'introns. Les signaux les plus étudiés sont les signaux d'épissage, puis viennent les signaux d'initiation et de terminaison de la traduction. Etant donné que nous nous intéressons à la prédiction du CDS des transcrits des gènes, la prédiction des signaux d'initiation de la transcription, et de polyadénylation (voir figure 2.1 page 12) ne seront pas abordés ici.

Les mesures par signal extraient d'un signal certaines propriétés, comme par exemple un motif consensus ou des dépendances entre certaines positions. La plupart d'entre elles utilisent un apprentissage du signal sur deux types de séquences : des séquences contenant le signal, dites *positives*, et des séquences ne contenant pas le signal, dites *négatives*. L'utilisation de ces méthodes consiste en une étape de test de la présence du signal sur une séquence inconnue.

Les mesures par signal se divisent en trois catégories [82] : la *stratégie probabiliste*, les *réseaux de neurones artificiels* et l'*analyse discriminante*.

La stratégie probabiliste. La stratégie probabiliste estime les probabilités des différentes positions du signal en calculant la vraisemblance de séquences signal candidates. Il existe principalement trois mesures par signal fondées sur une approche probabiliste : le *modèle de Markov inhomogène d'ordre k*, la *décomposition de dépendance maximale (MDD)* et les *réseaux bayesiens*. La première approche permet de modéliser des dépendances entre des positions adjacentes d'un signal, alors que les deux dernières prennent en compte des dépendances plus complexes.

Le modèle de Markov inhomogène d'ordre k, ou WMM pour Weight Matrix Method, est un modèle de Markov d'ordre k classique (voir mesures par contenu plus haut) dans lequel la probabilité d'un nucléotide dépend non seulement des k nucléotides qui le précèdent, mais aussi de sa position au sein du signal. Si la longueur du signal est N, il est donc nécessaire d'estimer $N - k$ fonctions de probabilité conditionelle, au lieu de 3 pour les mesures par

contenu de paramètre k. Exceptée cette différence, l'estimation se fait selon le même schéma que pour les mesures par contenu : en chaque position de la séquence signal les probabilités d'apparition de chacun des 4 nucléotides en fonction de l'apparition précédente de chacun des 4^k k-mères possible sont répertoriées sur des séquences d'apprentissage.

Le modèle de Markov inhomogène d'ordre k est utilisé dans de très nombreux programmes, souvent pour prédire les signaux d'epissage. On peut citer les programmes Grail [137], Geneparser [129, 130], Genemark [39], Genscan [44], Morgan [124] et GlimmerM [125]. Pour estimer les paramètres de ce modèle de façon plus fiable, certains programmes le combinent à des réseaux de neurones artificiels. C'est le cas de Netplantgene [79] et de NNSplice [119] avec l'ordre $k = 0$, ainsi que de [82] avec les ordres $k = 1$ et $k = 2$.

Le modèle de Markov inhomogène d'ordre 0 suppose l'indépendance des nucléotides adjacents. Il est plus connu sous le nom de *matrice de poids*, ou *Position Weight Matrix (PWM)*. Sa phase d'apprentissage se compose de deux étapes :

1. l'alignement de L séquences de tailles N contenant un signal (ensemble de données D),
2. le calcul des fréquences d'apparition de chaque nucléotide en chaque position du signal, ce qui donne une matrice de taille $4\times N$.

A partir de cette matrice de poids, on peut déduire une séquence consensus du signal (le nucléotide le plus fréquent en chaque position du signal), ou encore une expression régulière (voir figure 2.4). La phase de test consiste alors en le calcul d'un score pour la séquence inconnue afin de déterminer si celle-ci contient ou non le signal.

La matrice de poids est le fondement de nombreuses mesures probabilistes plus sophistiquées, telles que la *décomposition de dépendance maximale* ou les *réseaux bayesiens*.

La décomposition de dépendance maximale est plus fine que le modèle de Markov inhomogène d'ordre k car elle permet de modéliser des dépendances entre des positions du signal non nécessairement adjacentes. Elle débute par une première analyse des séquences d'apprentissage permettant de calculer la séquence consensus $C_1 \cdots C_N$ du signal[1]. Puis les séquences d'apprentissage sont parcourues une deuxième fois : pour chaque couple $i, j \in [1 : N]$, $i \neq j$ de positions distinctes du signal, si on note X_j le nucléotide apparaissant en position j des séquences d'apprentissage, on calcule la valeur du χ^2 entre les variables C_i et X_j. Si ces valeurs montrent une dépendance forte entre des positions i et j non adjacentes du signal, alors on réalise les deux étapes suivantes (dans les autres cas un modèle de Marov inhomogène d'ordre $k \geq 0$ est utilisé) :

1. pour chaque position i on calcule la somme :

$$S_i = \sum\nolimits_{j \neq i} \chi^2(C_i, X_j)$$

[1] On note C_i le nucléotide consensus en position i du signal.

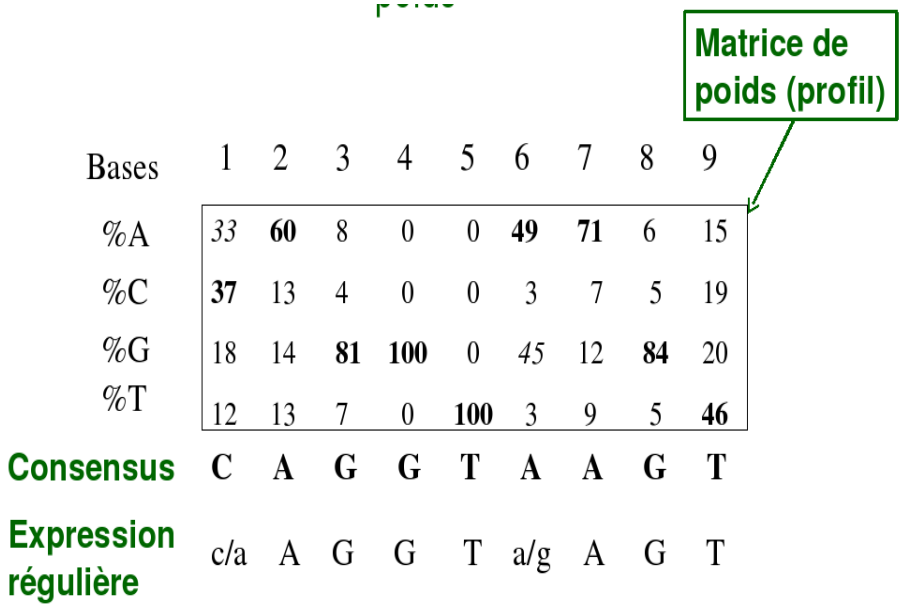


FIG. 2.4 – **Matrice de poids, consensus et expression régulière.** Cette figure représente la matrice de poids, le consensus et l'expression régulière relatifs à une séquence de 9 bases encadrant le signal donneur d'épissage. Plusieurs séquences d'ADN de vertébrés contenant un signal donneur d'épissage ont été alignées et en chaque position de cet alignement ont été répertoriées les fréquences d'apparition des 4 nucleotides. Ces fréquences constituent la matrice de poids ou profil. A partir de cette matrice de poids un signal consensus peut être calculé : il répertorié le nucleotide le plus fréquent en chaque position du signal. Enfin l'expression régulière correspondant au signal est calculée. Cette figure est extraite du papier décrivant le programme Genscan [44].

qui mesure la dépendance entre le nucléotide consensus C_i et les nucléotides des autres positions du signal,

2. la valeur i_1 telle que S_{i_1} est maximale est retenue, et l'ensemble de données D est partitionné en deux sous-ensembles : D_{i_1} incluant les séquences possédant le(s) nucléotide(s) consensus en position i_1, et $D_{i_1}^- = D - D_{i_1}$ incluant le reste des séquences.

Puis les étapes 1 et 2 sont répétées récursivement sur chacun des deux sous-ensembles D_{i_1} et $D_{i_1}^-$, formant ainsi un arbre de décision binaire, jusqu'à ce que l'une des conditions suivantes soit vérifiée :

a. le niveau $N-1$ de l'arbre est atteint,
b. on n'observe plus de dépendance significative entre les positions d'un sous-ensemble,
c. le nombre de séquences d'un sous-ensemble est trop petit pour qu'une fois subdivisé le calcul des fréquences de la matrice de poids associée soit fiable.

Ainsi le modèle final est un modèle composite formé d'autant de matrices de poids que de sous-ensembles de D aux différentes feuilles de l'arbre. La figure 2.5 donne un exemple de

décomposition de dépendance maximale.

Le programme Genscan [44] utilise la décomposition de dépendance maximale pour modéliser le signal donneur d'épissage. Pour former le modèle composite final, il est possible d'utiliser des modèles de Markov inhomogènes d'ordre k plutôt que des matrices de poids. Ce raffinement est utilisé par le programme Genesplicer [116] avec $k = 1$ pour les deux types de signaux d'épissage.

Comme la décomposition de dépendance maximale, les réseaux bayesiens généralisent le modèle de Markov inhomogène d'ordre k. En effet chaque position du signal peut ici dépendre de plusieurs positions précédentes, même si elles ne lui sont pas adjacentes. A chaque position i du signal, le réseau bayesien associe une variable aléatoire discrète X_i, prenant ses valeurs dans $\Sigma_{nt} = \{A, T, G, C\}$. La règle de Bayes donne la fonction de distribution jointe globale $P(x_1, x_2, \cdots, x_N)$ des variables aléatoires $X_1, X_2, \cdots, X_N$:

$$P(x_1, x_2, \cdots, x_N) = P(x_1)P(x_2|x_1)\cdots P(x_N|x_1, \cdots, x_{N-1})$$

Cette formule est valable dans le cas très général où chaque position i du signal dépend des $i-1$ positions qui la précèdent. En général ceci n'est pas le cas et une position donnée ne dépend pas que de quelques positions précédentes. Soit $I_i \subseteq \{1, 2, \cdots, i-1\}$ l'ensemble des positions dont i dépend, on note $E_{x_i} = \bigcup_{j \in I_i} \{x_j\}$ et la formule précédente devient :

$$P(x_i|x_1, \cdots, x_{i-1}) = P(x_i|E_{x_i}) \quad (2)$$

Un réseau bayesien peut donc se voir comme un graphe acyclique orienté, autrement dit un arbre, dont les sommets représentent les différentes positions du signal, et dont les arêtes orientées modélisent les dépendances entre celles-ci. Si la position i dépend des positions de $I_i \subseteq \{1, 2, \cdots, i-1\}$, alors le réseau comporte une arête orientée partant de chaque noeud d'indice $j \in I_i$ vers le noeud d'indice i. De plus chaque noeud d'indice i porte la probabilité conditionnelle $P(x_i|E_{x_i}$. Ainsi dans un réseau bayesien, une position donnée peut influer sur plusieurs positions suivantes, mais ne peut elle-même être influencée que par une seule position au maximum. La figure 2.6 donne un exemple de réseaux bayesien.

L'apprentissage du réseau bayesien permet d'obtenir l'arbre probabiliste décrit ci-dessus. Cai *et al.* [46] décrit une méthode simple d'apprentissage du réseau qui maximise la vraisemblance des données sachant le modèle. Cette méthode procède en cinq étapes :

1. calcul de l'information mutuelle $M(i, j)$ entre chaque couple de positions i, j :

$$M(i, j) = \sum\nolimits_{x,y} p(x, y) log(p(x, y))/p(x)p(y)$$

où x et y représentent toutes les valeurs possibles prises par les variable aléatoires X_i et X_j,

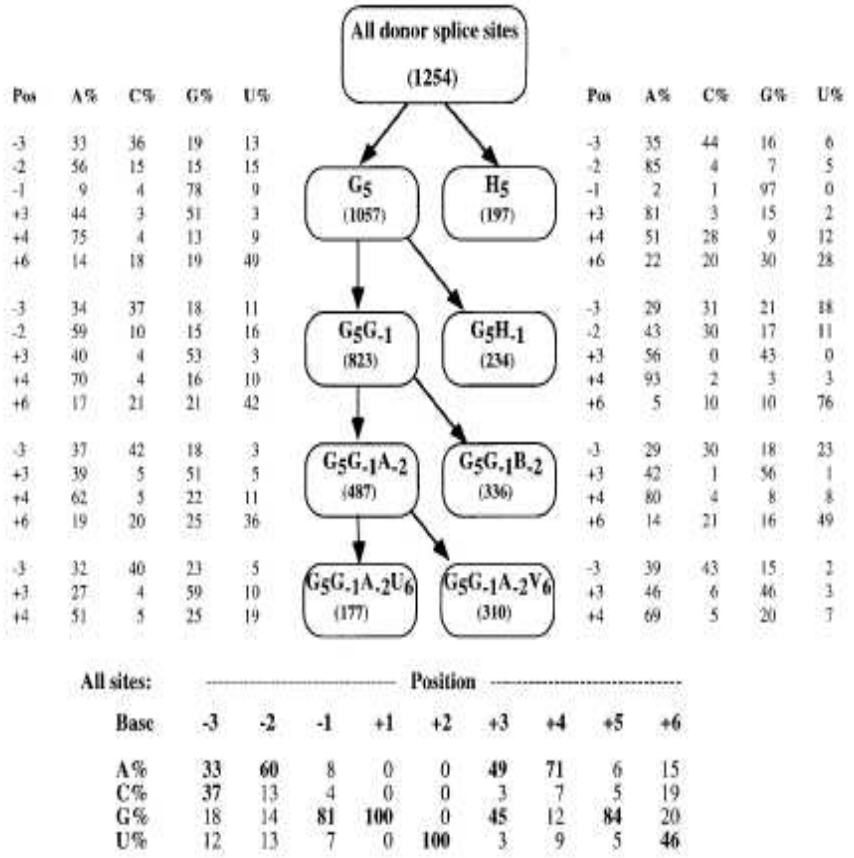


FIG. 2.5 – **Décomposition de dépendance maximale.** Ce schéma illustre la classification des séquences de signaux donneurs d'épissage effectuée par décomposition de dépendance maximale du programme Genscan [44]. Ces séquences sont considérées comme faisant partie d'ARN pré-messagers non épissés, et les quatre nucléotides possible sont donc A,U,G,C (et non A,T,G,C). Chaque noeud de l'arbre représente une sous-classe de séquences de signaux donneurs d'épissage associée à un motif d'appariements et de misappariements donné par rapport au(x) nucléotide(s) consensus du signal. G_5 par exemple, désigne l'ensemble des séquences de signaux donneurs d'épissage avec un G en position 5, et G_5G_{-1} celles avec un G à la fois en position 5 et en position -1. Ici H désigne l'un des nucléotides A,C,U, B désigne l'un des nucléotides C,G,U et V désigne l'un des nucléotides A,C,G. Le nombre de séquences dans chacune des sous-classes est indiquée entre parenthèses dans le noeud correspondant à cette sous-classe. Les fréquences (pourcentages) de chacun des quatre nucléotides en chacune des positions du signal pour chaque sous-classe sont indiquées à côté du noeud de cette sous-classe, et pour l'ensemble des séquences au bas de la figure. Le bas de la figure indique également les nucléotides les plus fréquents en chaque position du signal [44]. Ce schéma est extrait de la publication décrivant le programme Genscan [44].

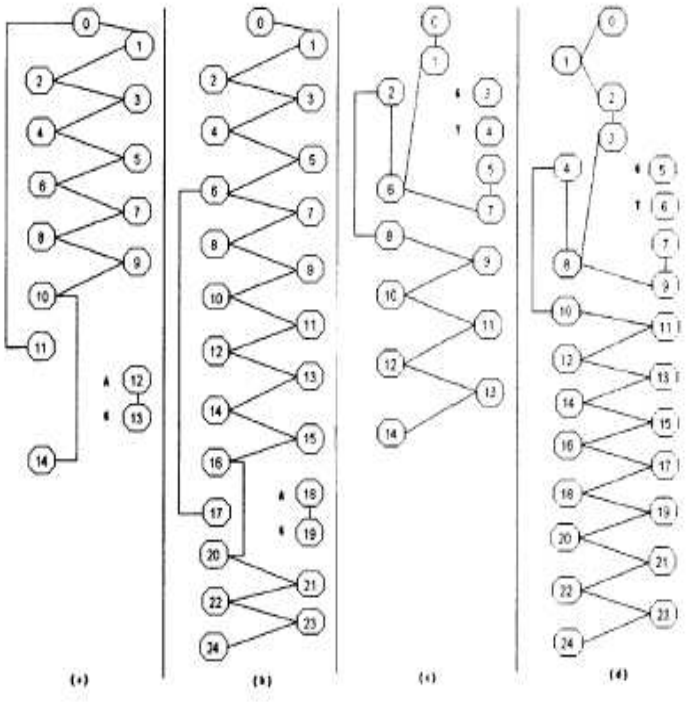


FIG. 2.6 – **Réseaux bayesien.** Cette figure représente quatre réseaux bayesiens (sous-figures (a), (b), (c), (d)) utilisés dans le programme de Cai *et al.* [46] pour modéliser les signaux accepteur et donneur d'épissage en utilisant différentes longueurs de signaux. Le numéro d'un noeud représente la position dans le signal. Les réseaux des sous-figures (a) et (b) modélisent le signal accepteur d'epissage en utilisant une longueur de signal de 15 et 25 nucléotides respectivement. Il en va de même pour les réseaux des sous-figures (c) et (d), mais cette fois pour le signal donneur d'epissage. Cette figure est extraite de [46].

2. construction d'un graphe pondéré G où le noeud i correspond à la variable aléatoire X_i. L'arête entre les noeuds i et j est pondéré par $M(i,j)$,
3. construction de l'arbre couvrant maximal de G,
4. orientation de l'arbre en choisissant comme racine la position 1, et en faisant en sorte que toutes les arêtes s'éloignent du noeud d'indice 1,
5. pour chaque couple de positions i, j tel que le noeud d'indice i est le parent du noeud d'indice j, calcul de la probabilité conditionnelle $p(x_j|x_i)$ en se fondant sur les fréquences des nucléotides observées.

Le test d'une séquence inconnue $S = S_1 \cdots S_N$ par le réseau bayesien consiste simplement en le calcul de sa probabilité d'apparition sachant le modèle/l'arbre. Celle-ci s'obtient par la

formule (2), autrement dit en suivant les arêtes de l'arbre depuis la racine.

Les réseaux bayesiens sont utilisées par Cai *et al.* [46], par Castelo *et al.* pour l'amélioration du programme Gene-ID [47], et par Chen *et al.* dans le cadre du programme DGSplicer [48]. Ce dernier programme utilise des réseaux bayesiens étendus par graphes de dépendance, afin de prendre en compte les dépendances cycliques entre les positions du signal.

Les réseaux de neurones artificiels. Les réseaux de neurones artificiels, que nous nommerons réseaux de neurones par la suite, représentent un ensemble de petites unités de traitement interconnectées cherchant à imiter le fonctionnement des neurones vivants. Ils représentent une méthode de prédiction fondée sur un apprentissage, généralement utilisée dans les domaines de la reconnaissance de motif et de la classification.

Plusieurs types de réseaux de neurones existent, mais les plus utilisés en génomique sont les réseaux de neurones dits «feed-forward». Un réseau de neurones «feed-forward» peut se voir comme un graphe pondéré orienté sans cycle dont les sommets, appelés *neurones*, sont structurés en trois types de couches, chacun possédant un rôle différent :

- une *couche d'entrée* ou couche 1 : elle sert à encoder certaines caractéristiques des séquences signal,
- $n \geq 0$ couches intermédiaires dites *cachées*, numérotées de 2 à $n+1$: par l'intermédiaire des poids de leurs arêtes elles contiennent la connaissance du réseau proprement dite, le *modèle*,
- une *couche de sortie* ou couche $n+2$: elle sert à encoder le résultat du réseau sur une séquence signal donnée.

Ces trois types de couches sont connectées entre elles dans cet ordre. Plus précisément des arêtes pondérées partent de tous les neurones de la couche i vers tous les neurones de la couche $i+1$ ($1 \leq i \leq n+1$). Au départ les poids de ces arêtes sont initialisés à une certaine valeur, puis l'entrée une à une des séquences d'apprentissage dans le réseau entraîne une modification de ces poids dont le but est d'améliorer la capacité de prédiction du réseau. La figure 2.7 illustre un réseaux de neurones multi-couche.

Il existe plusieurs types d'apprentissage pour les réseaux de neurones «feed-forward», mais le plus utilisé dans le domaine de la génomique est l'*apprentissage supervisé*. Dans un apprentissage supervisé, le résultat attendu sur chacune des séquences d'apprentissage est connu. L'apprentissage supervisé consiste ainsi pour le réseau à modifier dynamiquement ses poids de façon à minimiser la différence entre le résultat produit et le résultat attendu. La difficulté dans l'utilisation des réseaux de neurones «feed-forward» est le choix de l'architecture, autrement dit à la fois le nombre n de couches cachées et le nombre de neurones de chacune de ces couches. Une solution peut être de réaliser l'apprentissage avec différentes architectures puis de choisir celle qui donne le meilleur résultat.

La règle d'apprentissage supervisé la plus utilisée dans les réseaux de neurones «feed-

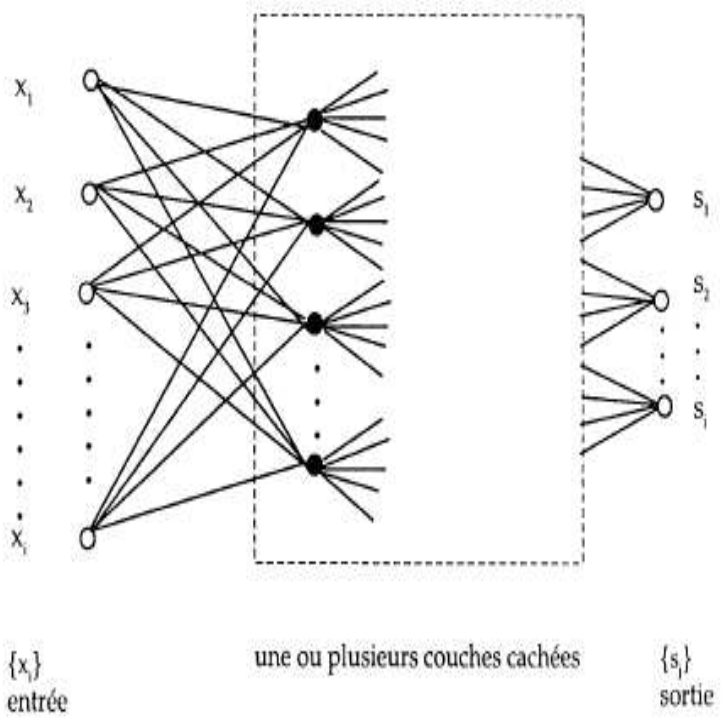


FIG. 2.7 – **Un réseau de neurones multi-couche.** Cette figure représente un réseau de neurones multi-couches. Seules la couche d'entrée, la première couche cachée et la couche de sortie sont représentées, mais il peut exister un nombre quelconque de couches cachées. Si l'on numérote les couches de gauche à droite (couche d'entrée vers couche de sortie), on peut dire que des arêtes orientées partent de tous les noeuds de la couche i vers tous les noeuds de la couche $(i+1)$. Cette figure est extraite de [13].

forward», est celle de la *correction d'erreur* : si l'on considère c comme étant la sortie calculée par le réseau, et a la sortie attendue, le principe est d'utiliser l'erreur $(c-a)$ pour modifier les connexions, diminuant ainsi l'erreur globale du système. Le réseau s'adapte donc jusqu'à ce que c soit égal à a. Pour plus de détails sur les réseaux de neurones artificiels se réferrer à [5].

Le principal inconvénient des réseaux de neurones est leur caractère opaque. En effet il est quasiment impossible d'en extraire une connaissance compréhensible, et d'autre part aucune connaissance biologique *a priori* ne peut leur être incorporée. Ces modèles sont utilisés pour la classification de signaux par les programmes suivants : le programme NetUTR [63] pour les signaux donneur et accepteur d'épissage des régions UTR 5', le programme de Pedersen *et al.* [114] pour le signal d'initiation de la traduction, et le programme de Zheng *et al.* [147]

pour les signaux donneur et accepteur d'épissage bordant les exons codants.

L'analyse discriminante. L'analyse discriminante est une méthode statistique de classification binaire utilisée dans le domaine de la reconnaissance de motifs. Comme les mesures *ab initio* présentées jusqu'ici, elle comporte une phase d'apprentissage et une phase de test. Il faut noter qu'elle ne nécessite pas la détermination d'une distribution de probabilité au préalable. Dans le cadre de la détection de signaux génomiques elle combine un ensemble de caractéristiques mesurables d'un signal dans le but de discriminer au mieux les séquences contenant le signal, dites positives ou de classe c_1, de celles qui ne le contiennent pas, dites négatives ou de classe c_2. Deux types d'analyse discriminante sont utilisées en génomique : l'*analyse discriminante linéaire (LDA)* et l'*analyse discriminante quadratique (QDA).*

Etant donné un signal, il s'agit d'abord de déterminer p caractéristiques réelles jugées importantes de ce signal. Pour chaque séquence signal sont ensuite relevées les valeurs de ces p caractéristiques et un vecteur caractéristique $x = (x_1, \cdots, x_p) \in \mathbb{R}^p$ est établi. Ainsi une séquence signal peut être vue comme un point de l'espace réel de dimension p. Chaque séquence signal, positive de classe c_1 et négative de classe c_2, est ainsi introduite dans l'espace des caractéristiques $\mathbb{R}^p$. Le but de l'analyse discriminante linéaire est alors de déterminer l'équation $\sum_{i=1}^{p} \alpha_i \times x_i = c$ de l'hyperplan qui sépare au mieux les points de classe c_1 des points de classe c_2.

Les p réels α_i de l'équation recherchée peuvent être vus comme des poids appliqués aux p caractéristiques du signal, et on note $\alpha = (\alpha_1, \cdots, \alpha_p)$ le vecteur de poids associé à ces réels. On introduit la fonction de discrimination linéaire $LDF_\alpha : \mathbb{R}^p \mapsto \mathbb{R}$ définie par :

$$\forall\ x \in \mathbb{R}^p,\ LDF_\alpha(x) = \sum_{i=1}^{p} \alpha_i \times x_i$$

L'apprentissage de la LDA consiste alors à déterminer la fonction LDF_α, ou le vecteur de poids α, et la constante c, telles que pour toute séquence signal d'apprentissage de vecteur caractétistique $x \in \mathbb{R}^p$:

$$\begin{cases} x \in c_1 & \Rightarrow & LDF_\alpha(x) < c \\ x \in c_2 & \Rightarrow & LDF_\alpha(x) \geq c \end{cases}$$

Soit $l \in \{1, 2\}$, on note $x_{i,k}^l$ la valeur de la i-ème caractéristique de la k-ième séquence signal de classe c_l, et n_l le nombre de séquences de classe c_l, on peut définir les deux paramètres suivants :

- μ^l, vecteur caractéristique moyen de la classe c_l par :

$$\mu^l = \frac{1}{n} \sum_{k=1}^{n_l} x_{i,k}^l$$

- $S = (s_{i,j})_{i,j \in [|1,p|]}$, matrice de covariance par :

$$s_{i,j} = \frac{1}{n_1 + n_2 - 2} \sum_{l=1}^{2} \sum_{k=1}^{n_l} (x_{i,k}^l - \mu_i^l)(x_{i,k}^l - \mu_j^l)$$

L'apprentissage de la LDA consiste ainsi à maximiser le ratio variation interclasse sur variation intraclasse, ce qui permet d'estimer α et c par les formules suivantes [132] :

$$\alpha = S^{-1}(\mu^1 - \mu^2),\ c = \frac{\alpha(\mu^1 - \mu^2)}{2}$$

Pour ce qui concerne l'analyse discriminante quadratique (QDA), le principe est exactement le même que pour l'analyse discriminante linéaire excepté le fait que la courbe de $\mathbb{R}^p$ recherchée pour séparer les signaux de classe c_1 des signaux de classe c_2 est une parabole et non un hyperplan. Ainsi pour réaliser l'apprentissage d'une QDA il suffit de prendre les mêmes formules que pour la LDA et de remplacer les termes x_i par des termes x_i^2. La figure 2.8 donne un exemple d'analyse discriminante.

Il est reconnu que la précision d'une méthode de classification se fait au détriment de la compréhensibilité ou transparence de la méthode. Dans ce contexte la LDA est reconnue pour représenter un bon compromis entre les deux [110]. Parmi les programmes utilisant l'analyse discriminante pour détecter les signaux, le programme Hexon [132] utilise la LDA et le programme JTEF [135] la QDA, pour reconnaître les signaux donneur et accepteur d'épissage. Quant au programme FirstEF [56], il utilise la QDA pour reconnaître le signal donneur d'épissage le plus en 5' du transcrit. Une stratégie fréquemment utilisée par les programmes est d'évaluer plusieurs types de combinaisons de caractéristiques d'un signal sur un ensemble d'apprentissage, et de conserver la combinaison qui donne le meilleur résultat : c'est la stratégie utilisée par le programme JTEF [135]. Remarquons que chacune des caractéristiques combinées par analyse discriminante s'accompagne en général d'une fenêtre d'analyse encadrant le signal, sur laquelle cette mesure est calculée.

Parmi les mesures d'analyse discriminante il faut noter l'intérêt croissant pour les *machines à support de vecteurs*, ou *SVMs* pour «Support Vector Machines». Les SVMs généralisent les techniques de LDA et de QDA : ce sont des méthodes de classification binaire qui visent à discriminer de façon linéaire des vecteurs d'un espace à p dimensions. Comme il n'est pas toujours possible de réaliser cette discrimination dans l'espace de dimension p initial, les SVMs translatent les vecteurs de caractéristiques des séquences signal dans une dimension p' grâce à des fonctions appelées *kernels*. Du fait de leur contenu technique les SVMs ne sont pas détaillées ici, citons néanmoins quelques programmes les utilisant pour détecter des signaux dans l'ADN génomique : le programme décrit dans [148] et le programme ATGpr [122] utilisent les SVMs pour les signaux d'initiation de la traduction, et le programme SpliceMachine [57] pour les signaux donneur et accepteur d'épissage.

Depuis une dizaine d'années, les mesures par signal se développent de façon beaucoup plus intense et diversifiée que les mesures par contenu, peut-etre du fait de leur plus grande précision pour localiser les exons. Il faut noter que la prédiction de signaux d'épissage est aujourd'hui un domaine presque aussi important que celui de la prédiction de gènes.Comme les mesures par contenu, les mesures par signal ont toutefois l'inconvénient de ne bien reconnaître que des séquences très similaires à celles utilisées lors de l'apprentissage. Du fait de leur

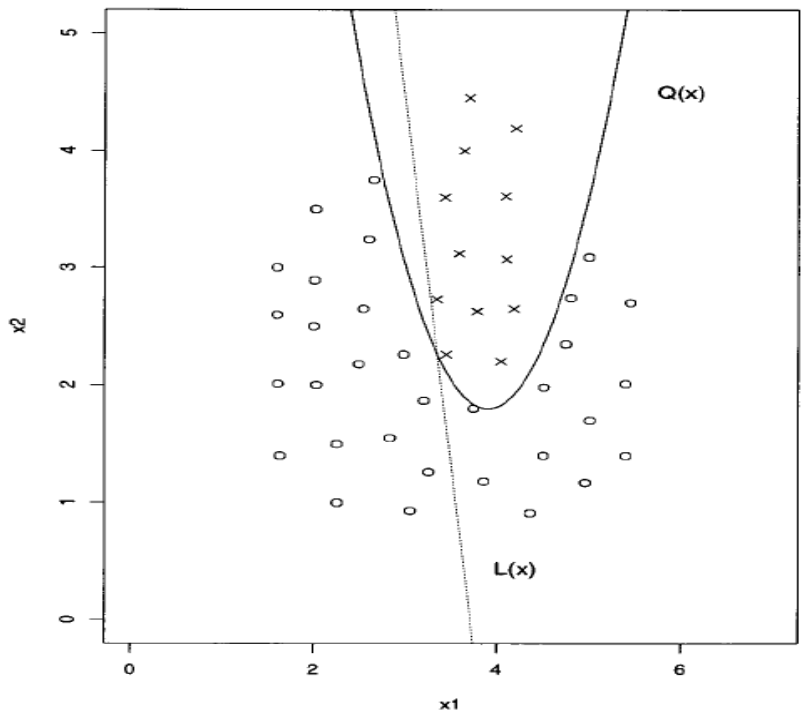


FIG. 2.8 – **Analyse discriminante.** Ce graphique représente un ensemble de signaux caractérisés par deux variables : x_1 en abscisses et x_2 en ordonnées. De plus ces signaux appartiennent à l'une des deux classes suivantes : c_1 s'ils sont représentés par une croix, et c_2 s'ils sont représentés par un rond. La fonction L, représentée par une droite en pointillé, désigne la meilleure fonction de discrimination linéaire de ces points. La fonction Q, elle, représentée par une parabole en traits pleins, désigne une fonction de discrimination quadratique de ces points. On constate ici que Q est une fonction de discrimination bien meilleure que L. Cette figure est extraite de la publication décrivant le programme MZEF [146].

complémentarité ces deux types de mesures sont souvent combinées dans le but d'annoter les exons de manière encore plus fiable.

Combiner les mesures pour identifier les exons

Il existe trois types de méthodes qui combinent les mesures par contenu et les mesures par signal pour annoter les exons des gènes : les réseaux de neurones, l'analyse discriminante, et l'*arbre de décision*. Les deux premiers types de méthodes ont été détaillés dans le cadre de la détection de signaux, cependant nous citons ici quelques programmes utilisant ces méthodes pour identifier les exons.

Les réseaux de neurones ont été utilisés par un programme précurseur dans le domaine de l'annotation automatique d'exons : Grail [137]. En effet le module de détection de pouvoir codant de Grail, appelé *CRM* pour «Coding Recognition Module», parcourt la séquence à analyser base par base, et détermine le potentiel codant de chaque base en combinant les valeurs de sept mesures par contenu et par signal calculées sur une fenêtre de 99 bases centrée sur la position de cette base. Les régions de potentiel codant supérieur à un certain seuil sont alors considérées comme des exons codants. Plus récemment le programme MoE (pour «Mixture of Experts») [110] utilise les réseaux de neurones pour combiner non plus des mesures statistiques telles que celles décrites plus haut, mais des sorties de programmes de détection d'exons (en l'occurence Genscan [44], MZEF [146] et GrailExp [85]).

L'analyse discriminante est utilisée en mode linéaire (LDA) par le programme Hexon [132] pour prédire des exons internes codants, et en mode quadratique (QDA) par les programmes MZEF [146] et JTEF [135] la QDA pour prédire des exons codants, respectivement internes et terminaux. Tous ces programmes utilisent les scores des signaux d'épissage bordant l'exon potentiel analysé, ces scores pouvant eux-mêmes résulter d'une analyse discriminante, comme par exemple pour le programme JTEF [135].

Le troisième type de méthode qui combine les mesures par contenu et les mesures par signal pour détecter les exons des gènes est l'arbre de décision. L'arbre de décision est une méthode de classification permettant de classer des objets dotés d'*attributs* ou caractéristiques, par division hiérarchique en sous-classes. Cette méthode se représente par un arbre dont :

- un noeud représente une classe de plus en plus fine depuis la racine,
- un arc représente un prédicat de partitionnement de la classe source, un ensemble de valeurs d'attributs.

On peut également voir un noeud de l'arbre comme une question portant sur certains attributs, et un arc sortant de ce noeud comme une réponse possible à cette question. La figure 2.9 représente un arbre de décision pour l'annotation d'exons codants.

Le processus a lieu en deux étapes : une étape d'apprentissage sur des données de classe connue, afin de construire l'arbre, et une étape de classification ou de test sur de nouvelles données. Le test d'un objet consiste en le parcours de l'arbre depuis la racine jusqu'à une feuille, en suivant en chaque noeud l'arc associé à la réponse à la question posée en ce noeud. Le problème réside dans l'apprentissage, et plus particulièrement dans le choix des attributs et des questions à poser en chaque noeud de l'arbre. En effet l'arbre commence à un noeud qui représente toutes les données, puis il s'agit (dans le cas où les objets n'appartiennent pas déjà à la même classe) de déterminer les attributs qui séparent au mieux les objets en classes homogènes, ce qui nécessite une fonction de qualité. Ce problème se produit récursivement jusqu'à ce que les objets soient assignés à une classe homogène. Pour choisir le meilleur attribut plusieurs stratégies sont possibles. Pour plus de détails sur le sujet voir [12].

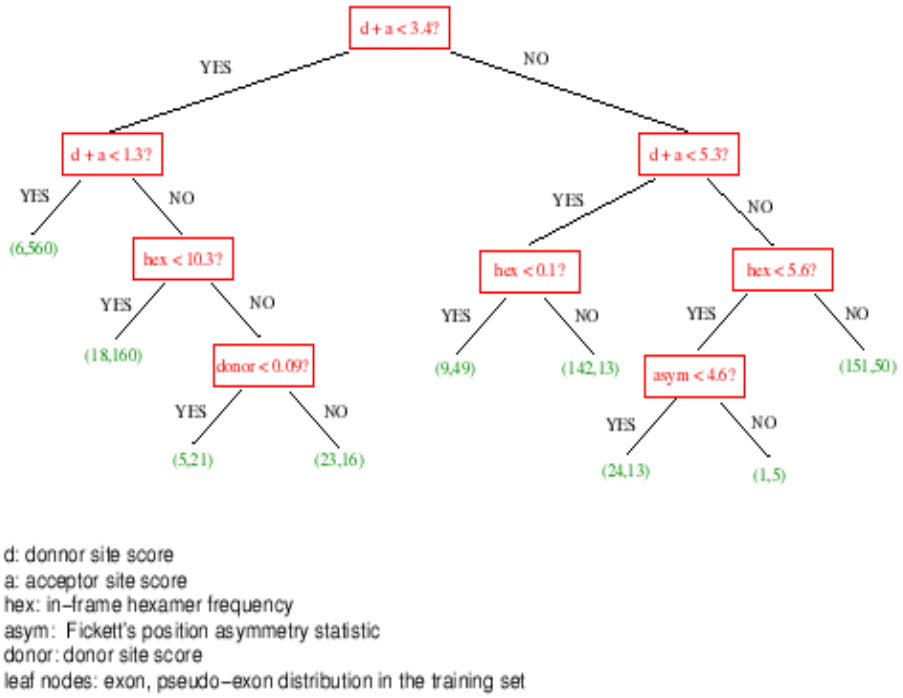


FIG. 2.9 – **Un arbre de décision.** Cette figure représente l'arbre de décision utilisé par le programme Morgan [124]. Cet arbre permet de partitionner un ensemble de données selon plusieurs critères. En chaque noeud de l'arbre une question est posée portant sur certains attributs des données. Cette figure est extraite de la publication décrivant le programme Morgan [124].

En réalité les feuilles obtenues sont rarement totalement homogènes et comportent le plus souvent plusieurs classes dont une est majoritaire. Aboutir à cette feuille signifie donc avoir une probabilité et non une certitude d'appartenir à cette classe, et cette probabilité se calcule sur la base des effectifs des différentes classes en ce noeud. Pour une donnée test de classe inconnue, il s'agit alors de descendre dans l'arbre depuis la racine, en suivant les branches correspondant aux réponses apportées aux questions sur cette donnée en chaque noeud. La classe de la donnée entrée est alors celle associée à la feuille à laquelle le parcours a abouti. L'avantage de l'arbre de décision est qu'il est compréhensible, mais aussi utilisable sur tout type de données (variables quantitatives ou qualitatives). Ses limites est qu'il est non incrémental (l'intégration de nouvelles données nécessitant de réitérer la construction de l'arbre), et qu'il ne s'applique que lorque le nombre de classes est relativement faible.

L'arbre de décision est souvent utilisé en combinaison avec d'autres techniques telles que les chaînes de Markov et la programmation dynamique. Le programme Morgan [124] est le

premier à avoir utilisé l'arbre de décision pour localiser des exons codants, et plus précisément des exons codants internes. Pour savoir si une séquence génomique est codante ou non, les attributs/caractéristiques suivants ont plus particulièrement été utilisés : le score de chaînes de Markov d'ordre 2 pour les signaux donneur et accepteur d'épissage, l'utilisation des héxamères et l'assymétrie dans la position des bases. La figure 2.9 issue de [124] illustre l'arbre de décision utilisé par le programme Morgan. Récemment le programme ATGpr [122] a également utilisé les arbres de décision pour la prédiction de signaux d'initiation de la traduction.

Les trois grandes classes de méthodes *ab initio* qui permettent d'identifier les exons d'une séquence d'ADN sont résumées dans le tableau 2.1.

Type	Méthode
Contenu	. Utilisation des k-mères . Modèle de Markov d'ordre $(k-1)$ et de période 3
Signaux	. Stratégie probabiliste . Réseaux de neurones . Analyse discriminante
Combinants	. Réseaux de neurones . Analyse discriminante . Arbre de décision

TAB. 2.1 – **Méthodes *ab initio* pour identifier les exons.**

Les méthodes présentées jusqu'ici, dites *ab initio*, se fondent sur les propriétés intrinsèques de la séquence à analyser afin de localiser les exons qu'elle comporte. Il existe un deuxième type de méthodes visant le même but, qui consiste à comparer la séquence à analyser avec d'autres séquences biologiques présentes dans les banques de données. Cette approche est dite par similarité de séquence.

2.1.2 Méthodes par similarité de séquence

Identifier les exons d'une séquence d'ADN par similarité de séquence, correspond à la comparaison de cette séquence à d'autres séquences biologiques, telles que :

1. des séquences de protéines[2] d'une autre espèce (voir annexe B),
2. des séquences d'ADNc de la même ou d'une autre espèce (voir annexe A),
3. des séquences d'ADN d'une autre espèce (voir annexe B).

En effet pour le cas 1 , le raisonnement est le suivant : si la traduction dans l'une des trois phases de lecture possible d'une sous-séquence s d'une séquence d'ADN, «ressemble beaucoup» à une sous-séquence de protéine, alors s a de forte chances de faire partie d'un exon codant.

Pour les cas 2 et 3, le raisonnement est quasiment identique : si s «ressemble beaucoup» à une sous-séquence d'ADNc de la même ou d'une autre espèce, ou encore à une sous-séquence d'ADN d'une autre espèce, alors s peut faire partie d'un exon.

Le problème revient donc ici à *quantifier la similarité* qui existe entre deux séquences, ou encore à *aligner* deux séquences.

Dans le domaine de l'annotation de gènes, les séquences sont vues commes des mots sur l'un des deux alphabets suivants :
- $\Sigma_{nt} = \{A, T, G, C\}$ (nucléotides),
- $\Sigma_{aa} = \{F, L, I, M, V, S, P, T, A, Y, H, Q, N, K, D, E, C, W, R, G\}$ (acides aminés),

et aligner deux séquences d'un même alphabet Σ de façon *globale* signifie informellement placer les deux séquences l'une en dessous de l'autre, et intercaler des espaces (caractère _) entre différents couples de positions de celles-ci, afin d'obtenir deux séquences de même longueur qui «se ressemblent».

Plus formellement, si l'on note Σ' l'alphabet Σ augmenté du caractère espace ($\Sigma' = \Sigma \cup \{\_\}$), un alignement global de deux séquences S et T sur Σ, est un couple (S^*, T^*), où $\begin{cases} S^*, T^* \in (\Sigma')^* \\ \mid S^* \mid = \mid T^* \mid \end{cases}$

D'autre part étant données deux séquences, on définit les trois opérations suivantes :
- la *substitution* d'une lettre de la première séquence par une lettre de la deuxième séquence,
- l'*insertion* d'une lettre dans la première séquence,
- la *suppression*, plus communément appelée *délétion*[3], d'une lettre de la première séquence.

Ces trois opérations sont regroupées sous le terme d'*opération d'édition*. Notons que l'opération de substitution d'une lettre par une autre inclut le cas d'identité de lettre (substitution de la lettre par elle-même), et que les opérations d'insertion et de délétion de lettre sont

[2]Pour comparer une séquence de nucléotides S à une séquence d'acides aminés P, il suffit de traduire S dans les trois phases de lecture et de comparer chacune de ces trois séquences à P.

[3]néologisme

appelés *indels*.

Il a été montré que disposer d'un alignement entre deux séquences était équivalent à disposer d'une séquence d'opération d'édition qui transforme la première séquence en la deuxième [21]. En effet soient $(S, T) \in \Sigma^{*2}$ deux séquences d'un même alphabet, et soit L_e une séquence d'opérations d'édition portant sur S, l'alignement (S^*, T^*) entre S et T correspondant s'obtient en parcourant linéairement L_e, et en effectuant l'un des actions suivantes en chacune de ses positions :

1. si la position contient une substitution, alors :
   - on ajoute la lettre courante de S dans S^* et on avance dans S,
   - on ajoute la lettre courante de T dans T^* et on avance dans T ;
2. si la position contient une insertion, alors :
   - on ajoute _ dans S^*,
   - on ajoute la lettre courante de T dans T^* et on avance dans T ;
3. si la position contient une délétion, alors :
   - on ajoute la lettre courante de S dans S^* et on avance dans S ;
   - on ajoute _ dans T^* ;

Pour désigner une suite de plusieurs espaces dans une séquence, on utilise le terme anglais de *gap*. Ainsi la longueur d'un gap est égale au nombre d'espaces qu'il contient. Ici l'alignement a lieu sur la totalité des séquence à aligner et est dit *global*. Certains alignements ne portent que sur des *sous-séquences* des séquences à aligner, ils sont dits *locaux*. Notons qu'un alignement peut également tolérer ou non les indels.

Ainsi pour un type d'alignement donné, il existe généralement plusieurs alignements possibles entre deux séquences S et T. Pour déterminer le(s) alignement(s) le(s) plus significatif(s) de S et T, étant donné un modèle d'alignement, il est nécessaire d'attribuer des scores, déterminés selon ce même modèle, aux différents alignements de S et T. Ainsi le meilleur alignement entre S et T est celui de *score maximal*. Si l'on suppose l'indépendance des positions adjacentes de l'alignement (S^*, T^*), et si l'on assigne un score à chaque type d'opération d'édition, on montre que le score de l'alignement de S et T est égal à la somme des scores des opérations d'édition correspondantes [18]. Il s'agit donc d'abord d'attribuer un score à chaque opération d'édition, autrement dit à la substitution et à l'indel (ou plus généralement le gap).

Score de substitution. Le score d'une substitution est donné par une fonction, *score* : $\Sigma \times \Sigma \mapsto \mathbb{R}$, que l'on peut représenter par une matrice de taille $| \Sigma | \times | \Sigma |$ et qui répertorie les scores de toutes les substitutions possibles de lettres sur l'alphabet Σ. Cette matrice est appelée *matrice de substitution*.

Pénalité de gap. Comme il n'est pas souhaitable d'avoir de nombreux indels dans un alignement, du moins si l'on suppose que les séquences à aligner se ressemblent, on attribue souvent un score négatif, une *pénalité* à l'indel. Comme les indels sont des insertions d'espaces dans une séquence et que ces espaces apparaissent souvent par blocs, c'est-à-dire sous forme de *gap*, on utilise souvent une fonction de pénalité de gap qui associe un score à un gap en fonction de sa longueur. La fonction de pénalité de gap $\gamma : \mathbb{N} \mapsto \mathbb{Z}$ est décroissante : plus le gap est long plus il est pénalisé. Certaines fonctions γ supposent les espaces du gap indépendants, d'autres qu'il est plus coûteux de débuter un gap, l'*ouvrir*, que de le prolonger.

En théorie, il est possible d'utiliser une fonction γ quelconque, cependant pour des raisons de complexité en temps et en espace, deux modèles de pénalité de gap sont principalement utilisés :

- le modèle de pénalité de gap *linéaire* : la pénalité d'un gap est proportionnelle à sa longueur :
$$\gamma(g) = -d \times g$$
où g est la longueur du gap et d est un entier positif représentant la valeur absolue de la pénalité d'un espace,
- le modèle de pénalité de gap *affine* : la pénalité d'un gap est une fonction affine de sa longueur :
$$\gamma(g) = -(d + e \times (g - 1))$$
où g est la longueur du gap, et où d et e sont des entiers positifs. d représente la valeur absolue de la pénalité du premier espace dit «espace d'ouverture du gap», et e représente la valeur absolue de la pénalité des $(g-1)$ espaces suivants du gap ou «espaces de prolongement du gap». Comme on suppose qu'il est plus coûteux d'ouvrir un gap que de le prolonger, on a $e < d$.

Pour des raisons de clarté nous utilisons dans la suite un modèle de pénalité de gap linéaire. Dans un tel modèle, la pénalité d'un espace ou gap de taille 1, est donc de $-d$[4].

L'alignement de score maximal entre S et T, dit *alignement optimal* de S et T, est donné par des algorithmes d'alignement dits *exacts*.

Algorithmes d'alignement exacts

Etant données deux séquences S et T de taille n et m sur un alphabet Σ, l'alignement global optimal (S^*, T^*) de S et T, s'obtient par l'algorithme de *Needleman-Wunsch* [107]. L'algorithme de Needleman-Wunsch produit en effet un couple de séquences (S^*, T^*) où
$$\begin{cases} S^* = S_1^*.\cdots.S_l^* \\ T^* = T_1^*.\cdots.T_l^* \\ l \geq max(n,m) \\ S_i^*, T_j^* \in \Sigma',\ 1 \leq i,j \leq l \end{cases}$$

[4]Un modèle affine de pénalité de gap conduit à une représentation par machine d'états finis des séquences à aligner.

L'algorithme de Needleman-Wunsch se fonde sur l'idée que pour obtenir l'alignement global optimal de $S_1 \ldots S_i$ et $T_1 \ldots T_j$, dont le score est ici noté $M(i,j)$, il suffit d'obtenir l'alignement global optimal entre $S_1 \ldots S_{i-1}$ et $T_1 \ldots T_{j-1}$. En effet, si l'on dispose de l'alignement global optimal entre $S_1 \ldots S_{i-1}$ et $T_1 \ldots T_{j-1}$, alors il y a trois possibilités pour la position courante de l'alignement entre S et T :

1. on aligne S_i avec T_j, ce qui donne le score d'alignement :

$$M(i-1, j-1) + score(S_i, T_j)$$

2. on aligne S_i avec un espace _, ce qui donne le score d'alignement :

$$M(i-1, j) - d$$

3. on aligne T_j avec un espace _, ce qui donne le score d'alignement :

$$M(i, j-1) - d$$

Pour obtenir $M(i,j)$ il suffit donc de prendre le maximum de ces trois scores, et l'alignement global optimal à ce stade de prendre celle des trois options qui produit ce maximum. On obtient ainsi la formule récursive suivante :

$$M(i,j) \;=\; max \begin{cases} M(i-1, j-1) + score(S_i, T_j) \\ M(i-1, j) - d \\ M(i, j-1) - d \end{cases} \qquad i,j \geq 1 \qquad (E)$$

D'autre part les valeurs initiales de M sont :

$$\begin{cases} M(i,0) &= -i \times d \\ M(0,j) &= -j \times d \end{cases}$$

et correspondant respectivement au préfixe $S_1 \ldots S_i$ de S aligné avec un gap de taille i dans T, et au préfixe $T_1 \ldots T_j$ de T aligné avec un gap de taille j dans S.

On obtient ainsi une matrice M de dimension $(n+1) \times (m+1)$, que l'on peut construire à partir de la première cellule $M(0,0) = 0$ grâce à la formule (E), jusqu'au dernier élément $M(n,m)$ qui, par définition, est le score de l'alignement global optimal entre S et T recherché. Pour obtenir l'alignement correspondant à ce score, il suffit de retrouver le chemin des choix de l'équation (E) qui ont mené à $M(n,m)$ depuis $M(0,0)$: on remonte dans la matrice M en partant de la dernière cellule $M(n,m)$, jusqu'à la première cellule $M(0,0)$, en suivant à partir d'une cellule $M(i,j)$ donnée, celle des trois cellules $M(i-1, j-1)$, $M(i-1, j)$ et $M(i, j-1)$ qui a conduit à $M(i,j)$ par (E).

En même temps que l'on remonte dans la matrice M, on reconstitue l'alignement optimal global inverse entre S et T. En effet si l'on se place à la fin des deux séquences S et T et en la cellule $M(n,m)$ de M, alors, pour chaque cellule $M(i,j)$ du chemin de cellules de $M(n,m)$ à $M(0,0)$, on retient une paire de lettres dans l'alignement inverse de S et T. Plus précisément si on remonte de la cellule $M(i,j)$ à :

- la cellule $M(i-1,j-1)$: on stocke (S_i, T_j) dans (S^*, T^*),
- la cellule $M(i-1,j)$: on stocke $(S_i, \_)$ dans (S^*, T^*),
- la cellule $M(i,j-1)$: on stocke $(\_, T_j)$ dans (S^*, T^*).

Si S et T représentent des séquences qui ont divergé de manière ancienne au cours de l'évolution, il se peut que S et T aient accumulé un grand nombre de mutations. Ainsi S et T ne s'aligneront pas bien sur toute leur longueur, mais uniquement sur certaines de leurs sous-séquences. Pour cette raison il est nécessaire d'effectuer un alignement *local* de S et T : c'est l'algorithme de *Smith-Waterman* [128].

L'algorithme de Smith-Waterman permet d'obtenir tous les alignements locaux de S et T de score supérieur à un certain seuil. L'algorithme de Smith-Waterman est une extension de l'algorithme de Needleman-Wunsch qui modifie ce dernier de la façon suivante :

- si pour une position (i,j) de l'alignement, c'est-à-dire au niveau de la cellule $M(i,j)$ de la matrice M, les trois valeurs $M(i-1,j-1)$, $M(i-1,j)$ et $M(i,j-1)$, sont toutes strictement négatives, on autorise $M(i,j)$ à prendre la valeur 0. Cela correspond à débuter un nouvel alignement en la position (i,j) des séquences S et T. L'initialisation de M devient donc $M(i,0) = M(0,j) = 0$;
- l'alignement local optimal pouvant se terminer en n'importe quelles positions des séquences S et T, il n'est plus nécessaire de partir de la dernière cellule $M(n,m)$ de M pour trouver le meilleur alignement : il suffit de partir de la cellule de score $max_{i,j}\ M(i,j)$.

Les algorithmes exacts décrits ci-dessus fournissent la solution optimale au problème de l'alignement, global ou local, entre deux séquences. Cependant leur complexité en temps et en espace est $\Theta(n \times m)$, où m et n sont les tailles des séquences à aligner. Ainsi il sera très difficile, voire impossible, de les utiliser sur des séquences de génomes entiers ou des bases de données de plusieurs dizaines de milliers de protéines. Les heuristiques d'alignement apportent une solution à ce problème.

Heuristiques d'alignement

Pour fournir des alignements de score élevé en utilisant moins de temps et d'espace mémoire que les algorithmes exacts d'alignement, les heuristiques d'alignement restreignent l'application de la programmation dynamique à un sous-ensemble de séquences significatives, autrement dit à une sous-matrice de M. Pour cette raison elles produisent toutes des alignements *locaux*.

Les heuristiques d'alignement se fondent sur l'hypothèse que de bons alignements locaux (i.e. de score élevé) entre deux séquences S et T, doivent très certainement inclure des sous-séquences de S et T identiques ou très voisines. Elles recherchent donc d'abord ce type de sous-séquences dans S et T, dites *séquences d'ancrage*, puis cherchent à les étendre dans les deux directions. Ainsi les heuristiques d'alignement comportent en général trois étapes :

1. Recherche des séquences d'ancrage,
2. Extension des séquences d'ancrage dans les deux directions (5' et 3'),
3. Utilisation de la programmation dynamique sur ces extensions afin de produire des alignements avec indels, le cas échéant.

Les heuristiques d'alignement Fasta [113] et Blast [29] fonctionnent de cette manière, et sont très largement utilisées pour comparer une séquence à un ensemble d'autres séquences. Plus précisément la première version de Blast, Blast1, produit des alignements sans indels, alors que Fasta et la deuxième version de Blast, Blast2 [30], produisent des alignements avec indels. Il faut noter que la perte en sensibilité générée par ces heuristiques par rapport aux algorithmes exacts est généralement tolérée car peu importante au regard du gain en temps et en espace mémoire qu'elles permettent.

Blast a l'avantage d'attribuer une valeur de confiance statistique aux alignements locaux qu'il produit. Les alignements locaux sans gap de Blast1, autrement dit les sous-séquences d'ancrage étendues en 5' et en 3' et de longueur maximale, sont appelées *HSPs* pour *High scoring Segment Pairs*. Le terme **HSP** sera très largement utilisée par la suite, et sera étendu à tous les résultats de programmes d'alignements locaux heuristiques. Pour des détails sur le fonctionnement du programme Blast1 voir annexe C.

Il est possible de réaliser un alignement local entre une séquence d'ADN génomique S et une séquence de protéine P. En effet il suffit de traduire la séquence S en acides aminés dans les trois phases de lecture possible, puis de comparer P à ces trois séquences. Si l'on nomme $Ps_p,\ p \in \{1,2,3\}$ les séquences traduites de S dans les trois phases de lecture possible, alors un HSP sur Ps_p indique la présence d'un exon codant en phase p sur S. Dans le cas d'un alignement entre une séquence d'ADN S et une base de données de protéines, il arrive que plusieurs protéines s'alignent au niveau d'une même région de S, ce qui représente une redondance. Le programme Exofish [53] élimine cette redondance en formant pour chaque ensemble maximal d'HSPs chevauchants, une *ecore* ou Evolutionary COnserved REgion. Les ecores d'Exofish peuvent être vues comme *modèles d'exons codants* de S.

Une autre manière de localiser les exons de S par alignement local, est d'utiliser des séquences d'ADN génomique appartenant à d'autres espèces que celle de S (voir annexe B). En effet des régions similaires entre deux génomes sont des régions qui ont été conservées au cours de l'évolution, et représentent donc des *régions fonctionnelles*. Cette approche a l'avantage de comparer les séquences d'ADN brutes, mais ne détecte pas *que* les exons codants.

De façon générale la stratégie par similarité de séquence localise les exons dans les séquences d'ADN génomique avec une assez grande fiabilité par rapport à la stratégie *ab initio*. Cependant les algorithmes par similarité décrits ci-dessus ne fournissent pas les bornes exactes des exons, contrairement aux algorithmes *ab initio* décrits en partie 2.1.1, et nécessitent de constantes mises à jour quand les bases de données publiques changent.

Aujourd'hui la localisation d'exons codants dans les génomes n'est pas un but en soi, mais souvent une étape préliminaire à la détermination de structures de gènes.

2.2 Annoter les gènes

De même que les méthodes d'annotation d'exons, les méthodes d'annotation de gènes se divisent en deux grandes catégories : d'une part les méthodes *ab initio* qui réalisent un apprentissage à partir de séquences connues afin d'être capable d'annoter les gènes de séquences inconnues, et d'autre part les méthodes par *similarité de séquence* qui comparent la séquence à annoter à d'autres séquences biologiques dans le but d'inférer sur la première certaines propriétés. Un troisième type de méthodes est récemment apparu, qui considère les sorties de programmes d'annotation existants comme des sources d'information, qu'ils combinent pour obtenir des annotations de gènes plus fiables. Nous les nommons méthodes *combinantes*.

2.2.1 Méthodes *ab initio*

Les méthodes qui déterminent les structures de gènes de façon *ab initio* se divisent en deux catégories : les méthodes *fondées sur les exons*, qui recherchent d'abord les exons de la séquence à annoter puis les assemblent en structures de gènes, et les méthodes *fondées sur les signaux* qui déterminent simultanément les exons et les structures de gènes de la séquence à annoter [76].

Méthodes fondées sur les exons

Les méthodes fondées sur les exons procèdent de la manière suivante :

1. elles recherchent d'abord tous les exons de S de façon *ab initio*, très souvent par une combinaison de mesures par contenu et par signaux (voir section 2.1.1),
2. elles attribuent ensuite à ces exons : d'une part une position dans le futur gène (par exemple initial ou interne) et d'autre part un score, le plus souvent sur la base des mesures ayant permi de les localiser,
3. enfin elles sélectionnent parmi tous les assemblages cohérents (un exon interne ne peut suivre qu'un exon initial ou interne) d'exons, les assemblages de scores maximaux : ce sont les structures de gènes de S.

Notons que cette dernière étape peut être vue comme la recherche d'un chemin optimal dans un graphe acyclique orienté dont les sommets représentent des exons et dont les arêtes représentent des relations de compatibilité entre ces exons [102].

Les méthodes fondées sur les exons ont l'avantage de découpler les étapes d'identification d'exons et d'identification de gènes, ce qui permet de les faire évoluer indépendamment. Les deux principaux programmes qui utilisent cette approche sont GeneID [111] et Fgene [131], toutefois la plupart des programmes d'annotation de gènes *ab initio* actuels utilisent une approche fondée sur les signaux.

Méthodes fondées sur les signaux

Les méthodes fondées sur les signaux utilisent les connaissances biologiques sur la structure des gènes eucaryotes, pour identifier ceux présents dans des séquences d'ADN génomiques inconnues. Les informations utilisées sont les suivantes :

- deux gènes successifs d'une séquence sont séparés l'un de l'autre par une régions dite *intergénique*,
- le gène est une succesion d'exons et d'introns, séparés les uns des autres par des signaux de transcription,
- la région codante d'un gène est délimitée par des signaux de traduction.

Ainsi les méthodes fondées sur les signaux considèrent l'ADN comme une succession de régions homogènes, ou *segments*, séparés les uns des autres par des *signaux*. Les exons, introns, régions UTR 5' et UTR 3', et régions intergéniques sont des exemples de segments, alors que les codons Méthionine et Stop, ou les dinucléotides GT et AG sont des exemples de signaux. La succession de segments représentée par cet ADN doit aussi être biologiquement valide/cohérente, c'est-à-dire suivre les règles énumérées plus haut. Ainsi un segment de type *exon* ne pourra être suivi que par un segment de type *intron* ou *UTR 3'*.

Les méthodes fondées sur les signaux modélisent l'ADN par un *automate* dont chaque état représente un type possible de segment, et dont chaque transition entre deux états représente une succession biolgiquement valide/cohérente entre les deux segments correspondants. L'automate permet de générer les nucléotides d'une séquence d'ADN par un processus :

- non déterministe : une séquence d'ADN peut être générée de plusieurs manières,
- Markovien d'ordre 1 : la probabilité de transiter vers un état ne dépend que de l'état courant,
- caché : la succession des états traversés pour générer une séquence n'est pas connue à l'avance.

Ainsi les méthodes fondées sur les signaux sont des *modèles de Markov cachés* ou *HMM* pour *Hidden Markov Model*. Enter une séquence d'ADN S au HMM génère la segmentation optimale de S selon le modèle du HMM, autrement dit les structures de gènes de S. La figure 2.10 issue de [44] illustre l'automate utilisé par le programme Genscan.

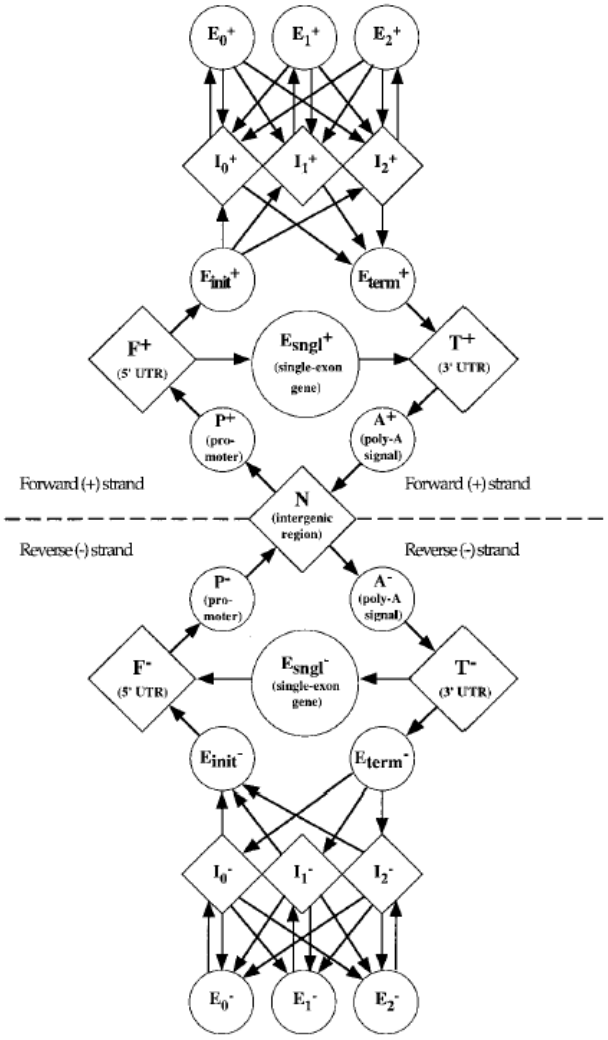


FIG. 2.10 – **Un modèle de Marov caché généralisé.** Cette figure représente le GHMM utilisé par le programme Genscan [44] pour déterminer les structures de gènes d'une séquence d'ADN génomique. Les ronds et losanges représentent les états du GHMM et les flèches représentent les transitions possibles entre états. Le trait pointillé horizontal sépare les états et transitions du brin + de la séquence de ceux de son brin -. Les ronds représentent des états exoniques, promoteurs ou polyA alors que les losanges représentent des états introniques, UTR ou inntergéniques. Sur le brin + de la séquence on distingue trois états pour les exons et trois états pour les introns en fonction de la phase. Cette figure est extraite du papier décrivant le programme Genscan [44].

Plus formellement un HMM est un quintuplet $\langle E, \Sigma, g, t, \pi \rangle$, où :

- $E = \{E_1, \ldots, E_n\}$ est un ensemble fini d'états,
- Σ est un alphabet de symboles (ici des nucléotides),
- $g : E \times \Sigma \mapsto [0,1]$ est une fonction de probabilité de génération de symbole associés à chaque état,
- $t : E \times E \mapsto [0,1]$ est une fonction de probabilité de transition d'un état vers un autre,
- $\pi : [1:n] \mapsto [0,1]$ est une fonction de probabilité pour l'état initial.

Un HMM permet donc de générer une séquence de symboles $S = s_1 \ldots s_L$ de la façon suivante :

1. Choisir comme état initial l'état E_i qui maximise la fonction de probabilité initiale π,
2. Choisir le symbole s_1 qui maximise la probabilité de génération de symbole dans l'état E_i grâce à la fonction g,
3. Transiter vers l'état E_j (éventuellement égal à E_i) qui maximise la probabilité de transition depuis l'état E_i grâce à la fonction t,
4. Revenir à l'étape 2 pour choisir le symbole s_2 de S, et ainsi de suite jusqu'à atteindre la taille L de la séquence S à générer.

La succesion des états traversés lors de la génération de S représente un chemin dans le HMM, ou encore une segmentation de S en une suite d'états du HMM. Etant donnée une séquence $S = s_1 \ldots s_L$ connue à l'avance, et en supposant le HMM appris sur un ensemble de séquences de segmentation connue, segmenter S selon le HMM consiste à calculer le chemin d'états ϕ de probabilité maximale.

Ainsi dans un HMM classique, la probabilité de rester dans un état, ou *modèle de durée*, suit une loi géométrique. En effet, si E_i est un état de E, si p_i désigne la probabilité de rester dans E_i ($p_i = t(E_i, E_i)$), et $p_i(d)$ celle de rester dans E_i pendant la génération de d symboles, alors :

$$p_i(d) = (p_i)^{d-1} \times (1 - p_i)$$

Dans le domaine de la prédiction de gène, ce modèle de durée n'est pas suffisamment fin, car il ne permet pas de prendre en compte les distributions des longueurs des segments de l'ADN, en particulier des exons et des introns. Pour pouvoir les prendre en compte, le HMM est modifié de la façon suivante [117] :

- le paramètre $f : E \times \mathbb{N} \mapsto [0,1]$, correspondant à la distribution des probabilités des durées de chaque état est ajouté,
- la probabilité de transition d'un état vers lui-même est mise à 0 :

$$\forall E_i \in E,\ t(E_i, E_i) = 0$$

- la fonction de probabilité de génération de symboles g devient une fonction de probabilité de génération de sequence :

$$g : E \times \Sigma^* \mapsto [0,1]$$

Un HMM comportant un modèle de durée pour chaque état est donc un sextuplet $\langle E, \Sigma, g, t, \pi, f \rangle$ et est appelé modèle de Markov caché *généralisé* (parfois *semi-Markovien*) ou *GHMM* pour *Generalised Hidden Markov Model*.

Chaque état E_i d'un GHMM étant associé à une durée d_i, une segmentation $s_1 \ldots s_l$ obtenue par un GHMM ne désigne plus une suite finie de symboles, mais une suite finie de sous-séquences de longueur d_i. Ainsi pour une longueur de séquence L fixée, si l'on considère l'espace $\Omega = \Phi_L \times \Psi_L$, où Φ_L est l'ensemble de toutes les segmentations possible des séquences de longueur L, et où Ψ_L est l'ensemble de toutes les séquences possible de longueur L, un GHMM peut être considéré comme une mesure de probabilité sur Ω.

Pour une séquence S de Ψ_L, il est possible de calculer la probabilité conditionnelle d'une segmentation particulière $\phi_i \in \Phi_L$ de S sous la mesure de probabilité induite par le GHMM en utilisant la règle de Bayes :

$$P(\phi_i|S) = \frac{P(\phi_i, S)}{P(S)} = \frac{P(\phi_i, S)}{\sum_{\phi_j \in \Phi_L} P(\phi_j, S)}$$

Maximiser $P(\phi_i|S)$ revient dont à maximiser la probabilité jointe $P(\phi_i, S)$.

Pour une séquence S de Ψ_L, la probabilité jointe, $P(\phi_i, S)$, de générer la segmentation ϕ_i et la séquence S, est donnée par :

$$P(\phi_i, S) = \pi_{E_1} f(E_1, d_1) g(E_1(d_1), s_1) \times \prod_{k=2}^{n} t(E_{k-1}(d_{k-1}), E_k) g(E_k(d_k), s_k)$$

où $\{E_1, \ldots, E_n\}$ désignent les états de ϕ_i, associés respectivement aux durées $\{d_1, \ldots, d_n\}$, et qui segmentent S en $s_1, \ldots, s_l$, $s_i \in \Sigma^*$, chaque s_i étant de longueur d_i et dans l'état E_i. Remarquons que $L = \sum_{i=1}^{n} d_i$.

Le but est alors de déterminer la segmentation ou chemin d'états ϕ_{opt}, qui maximise $P(\phi_i, S)$. Pour cela on utilise l'algorithme de programmation dynamique dit *de Viterbi* [139], qui est lui-même un cas particulier de l'algorithme des plus courts chemins de Bellman [15]. Si le modèle utilisé est un HMM simple la complexité en temps de cet algorithme est linéaire en la longueur de la séquence, en revanche si le modèle est un GHMM, la complexité en temps est quadratique en cette dernière. Si S est une séquence d'ADN génomique, la segmentation ainsi réalisée correspond aux différentes structures de gènes de S.

Il faut noter que dans un GHMM la fonction g de probabilité de génération de séquence peut être, pour chaque état de E un modèle arbitrairement complexe, comme par exemple une chaîne de Markov. Cela permet une grande flexibilité de modélisation, et en particulier de pouvoir prendre en compte les distributions des longueurs d'exons.

Aujourd'hui les GHMMs sont les méthodes *ab initio* qui fournissent les meilleurs résultats dans le domaine de la prédiction de gènes. Les programmes qui utilisent les GHMMs pour

annoter les gènes sont nombreux : Genscan [44] et Genie [96, 119, 120] sont des précurseurs du domaine, mais nous pouvons également citer Genemark.hmm [98], Fgenesh [123], GRPL [83], Exonomy [99], GlimmerHMM et Tigrscan [101], et plus récemment Augustus [133] qui utilise un modèle d'intron original. Un HMM de type *CHMM* pour *Class HMM*, est aussi utilisé par les programmes Veil [80], Unveil [99] et HMMgene [95].Notons que tous ces programmes utilisent différentes combinaisons d'états et de modèles pour ceux-ci, et que ces modèles sont souvent raffinés en sous-modèles selon le pourcentage en (G+C).

Le principal avantage des GHMMs est leur rapidité et leur facilité d'utilisation, en effet seule la séquence à annoter est nécessaire (une fois l'apprentissage réalisé). Ils ont aussi la capacité de prédire des gènes sans homologue connu, donc hors de portée des approches par similarité de séquence. En revanche leur principale limitation est qu'ils prédisent souvent une grande quantité de gènes incorrects, ce qui est probablement dû à la difficulté de prédire les signaux des gènes. Deux autres inconvénients importants est qu'ils ne prédisent en général qu'un seul transcrit par gène, et que leurs performances dépendent de l'espèce pour laquelle ils ont été calibrés [93].

Le nombre toujours croissant d'organismes nouvellement séquençés, et donc pour lesquels peu de données existent, fait de la prédiction de gènes dans ces génomes un problème d'actualité. Nous citons ici trois approches intéressantes : SNAP [93], Genemark.hmm ES [25] et Agene [87]. Le programme SNAP utilise un GHMM très simple par défaut, par exemple calibré sur l'espèce humaine, avec un nombre minimal d'états. Puis, pour une séquence à annoter d'une espèce quelconque, ce GHMM prédit des gènes préliminaires qui servent de données d'apprentissage à ce même GHMM. Ainsi SNAP représente un GHMM qui s'adapte à l'espèce étudiée, et est souvent meilleur que les GHMMs existants [93]. Une approche similaire est utilisée par le programme Genemark.hmm ES. Enfin, le programme Agene [87] est capable de créer automatiquement un nouveau CHMM pour chaque nouvelle espèce à partir de son génome et d'un ensemble d'ARNm. Son originalité est de créer automatiquement un ensemble d'apprentissage pour l'espèce à annoter, ce qui est une tâche délicate généralement laissée à l'utilisateur, et d'ajuster son modèle à la qualité des données d'apprentissage.

Comme dans le domaine de l'identification des exons, il existe un deuxième type de méthodes pour annoter les gènes d'une séquence d'ADN, qui consiste à comparer cette dernière à d'autres séquences biologiques présentes dans les bases de données : c'est la stratégie par similarité de séquence.

2.2.2 Méthodes par similarité de séquence

Annoter les gènes d'une séquence d'ADN génomique par similarité de séquence signifie comparer cette séquence, appelée *séquence-cible*, à une ou plusieurs autres séquences biologiques, appelées *séquences-sources*. Il existe trois types de séquences-sources pour l'annotation des gènes :

– des séquences de protéines d'une autre espèce (voir annexe B),

– des séquences d'*ADN complémentaire (ADNc)* (voir annexe A),
– des séquences d'ADN génomique d'une autre espèce (voir annexe B).

Cependant il existe une différence fondamentale entre les deux premières stratégies et la troisième, puisque dans un cas la séquence-source est épissée et de nature codante, c'est la séquence codante (CDS) ou l'ARN messager (ARNm) d'un transcrit, alors que dans l'autre la séquence-source n'est même pas épissée, à l'image de la séquence-cible[5]. Nous appelons ces deux types d'approches *comparaison à une séquence épissée codante* et *comparaison à une séquence d'ADN génomique*, et les décrivons dans les deux sous-sections suivantes.

Comparaison à une séquence épissée codante

Une séquence épissée codante est une séquence de protéine ou une séquence d'ADN complémentaire (ADNc), de type EST ou pleine longueur (voir annexe A). Chacun de ces types de ressource a un avantage qui lui est propre :
– utiliser une séquence de protéine permet de localiser des exons codants, mais aussi de connaître leur phase (voir section 2.1.2),
– utiliser une séquence d'ADNc permet d'identifier les exons contigus d'une structure de transcrit. Les exons identifiés peuvent être des exons codants, partiellement codants ou totalement non codants (UTR). Dans le cas d'ADNc provenant d'une espèce suffisamment proche de celle de la séquence à annoter (au mieux de l'espèce de la séquence à annoter), les bornes des exons sont déterminés de façon asseez précise, ce qui permet de prédire les structures des transcrits alternatifs des gènes de la séquence à annoter.

La comparaison de S à une séquence épissée codante, est parfois appelé *alignement épissé*.

La comparaison de S à une séquence épissée codante T nécessite une extension de l'alignement local, décrit en section 2.1.2 dans le cadre de l'identification des exons. Comme les transcrits eucaryotes sont constitués d'une succession d'exons et d'introns (voir figure 2.1), les alignements utilisés pour déterminer leur structure doivent pouvoir modéliser les introns. Le problème revient donc à déterminer sur S une succession d'exons qui une fois concaténés possèdent un score de similarité maximal avec T. Deux types d'approches ont tenté de résoudre ce problème :
– les modèles qui *étendent le modèle d'alignement exact de Smith-Waterman* pour prendre en compte les introns et leur site d'épissage, équivalents aux *modèles de Markov cachés doubles (pair HMM)*. Ces modèles produisent une solution exacte au problème qui vient d'être décrit, parfois coûteuse en temps,
– les *alignements heuristiques*, qui ne garantissent pas de donner une solution exacte à ce problème mais s'en approchent le plus souvent, et ont l'avantage d'être plus rapides.

Malgré une différence conceptuelle importante, ces deux stratégies ont en commun la nécessité d'une première étape d'alignement de la séquence épissée codante T avec la séquence S. Cet alignement est en général heuristique, rapide, sévère et peu précis, et a pour but de

[5]La première et la troisième approche sont des stratégies de *génomique comparative*.

déterminer la région homologue à T dans S de façon approximative.

Les modèles qui étendent l'alignement exact de Smith-Waterman pour prendre en compte les introns et leur site d'épissage, sont utilisés par les programmes Procrustes [74], PairWise [36] et Genewise [37] pour une comparaison avec des protéines d'une autre espèce, et par les programmes EST_GENOME [106], Geneseqer [138, 41], Genomewise [37] et GigoGene [50] pour une comparaison avec des transcrits de la même ou d'une autre espèce.

Les alignements épissés heuristiques sont utilisés par les programmes SIM4 [69], Spidey [140] et Blat [90] pour une comparaison avec des ADNc de la même ou d'une autre espèce. Parmi ces programmes, seul Spidey tente de prédire un CDS dans la structure de transcrit ainsi obtenue sur la séquence à annoter.

Une application importante de l'alignement épissé d'une séquence d'ADN à annoter avec un ensemble de séquences d'ADNc, est la détection des transcrits alternatifs des gènes de cette séquence. Le principe est de tenter d'inférer, à partir d'un tel alignement, l'ensemble minimal des structures de transcrits non redondantes de la séquence à annoter. Les programmes TAP (Transcript Assembly Tool) [88] et Altmerge [141], et plus récemment les programmes du problème de multi-assemblage [143], ESTGenes [66] et AIR [70], tentent de résoudre ce problème. Ces trois derniers programmes utilisent aussi de façon avantageuse un cadre formel fondé sur les graphes orientés acycliques colorés, dont les sommets sont des alignements d'ADNc (l'alignement d'un un exon de l'ADNc ou de l'ADNc entier), et les arêtes sont des relations fournies par l'expertise entre ces alignements. L'ensemble minimal non redondant des transcrits alternatifs des gènes de la séquence à annoter s'obtient alors par un parcours de ce graphe.

Comparaison à une séquence d'ADN génomique

La comparaison de la séquence d'ADN à annoter à une séquence d'ADN génomique d'une autre espèce fait l'hypothèse que des séquences conservées entre deux espèces au cours de l'évolution ont de grandes chance d'être fonctionnelles. Ainsi les séquences issues de cette comparaison pourraient être les exons codants de transcrits. Contrairement à la comparaison à une séquence épissée codante, cette approche ne nécessite pas de connaître les protéines ou les ADNc correspondant aux transcrits des gènes de la séquence à annoter.

Comme les approches par comparaison à une séquence épissée codante, les approches par comparaison à une séquence d'ADN génomique etendent l'alignement local de Smith-Waterman pour prendre en compte les introns des transcrits. Toutefois il existe une différence fondamentale entre ces deux approches : dans la comparaison à une séquence d'ADN génomique, seule la composition en nucléotides des deux séquences d'ADN prises en entrée est connue, mais rien ne l'est de leur fonctionnalité. Ainsi, une parfaite symétrie existe entre les deux séquences d'ADN prises en entrée par ces programmes, et ceux-ci sont donc capables de prédire les gènes de ces séquences simulatanément.

Si l'on nomme S la séquence d'ADN à annoter et S' la séquence d'ADN génomique auxiliaire, le problème revient ici, étant donnés un modèle de gène et un modèle d'alignement, à déterminer à la fois sur S et S', une succession de couples d'exons satisfaisant le modèle de gène et qui, une fois concaténés sur S et S', possède un score de similarité maximal. Deux types d'approches ont tenté de résoudre ce problème :

- les *modèles fondés sur les signaux ou modèle de Markov cachés doubles (pair HMM)*, qui déterminent en une passe, grâce à une phase d'apprentissage préalable, les structures de transcrits de S et S',
- les *modèles fondés sur les exons*, qui déterminent d'abord les paires d'exons de S et S' qui satisfont le modèle de gène, puis les structures de transcrits de S et S'. Cette dernière étape est effetuée en sélectionnant la combinaison de ces exons qui maximise le score de similarité (éventuellement additionné à un score de pouvoir codant et/ou de signaux).

Comme pour la comparaison à une séquence épissée codante, ces deux approches nécessitent une première étape d'alignement heuristique rapide, sévère et peu précis, de S et S', afin de déterminer les régions orthologues (voir annexe B) de ces séquences sur lesquelles appliquer plus précisément leur algorithme.

Les modèles de Markov cachés doubles (pair HMM) ont été utilisés par les programmes Doublescan [104], SLAM [26] et Twain [100]. Les modèles de Markov utilisés par les programmes SLAM et Twain sont de type généralisé (voir section 2.2.1 sur l'annotation de gène *ab initio*), c'est-à-dire permettent de modéliser de façon arbitraire la probabilité de rester dans un état, ce qui permet notamment de prendre en compte la distribution des longueurs des exons des transcrits. Le programme Doublescan a récemment été étendu pour prendre également en entrée les protéines orthologues (voir annexe B) aux protéines des gènes de S, c'est le programme Projector [103].

Les modèles fondés sur les exons ont été utilisés par les programmes Rosetta [34], CEM [32], SGP1 [142], SGP2 [112], Pro-Gen [108], de Pedersen *et al.* [115] et Utopia [38]. Notons que ces deux derniers programmes reposent sur un formalisme mathématique plus important. Tous ces programmes utilisent la programmation dynamique pour réaliser les combinaisons d'exons optimales représentant les différents transcrits de S et S'. Parmi ces programmes, les programmes SGP1 et SGP2 qui reposent sur le modèle de gène *ab initio* GeneID, sont sans doute les plus utilisés.

Les méthodes d'annotation de gènes *ab initio* et par similarité de séquence sont périodiquement évaluées sur des données de référence [45, 77, 33, 121, 65], témoignant de performances très acceptables en nucléotide (de l'ordre de 90%), acceptables en exons (de l'ordre de 80%), mais dans tous les cas très insuffisantes en transcrits (de l'ordre de 40%). Pour cette raison et du fait de la complémentarité des méthodes *ab initio* et par similarité de séquence, un nouveau type de méthode est apparu qui considère les sorties de certains programmes

d'annotation de gènes comme des sources d'information, qu'il s'agit de combiner au mieux en tenant compte à la fois des scores donnés par ces sources, mais aussi de la phase des gènes prédits par ces sources.

2.2.3 Méthodes combinantes

Les méthodes combinantes considèrent les sorties de certains programmes d'annotation comme des sources d'information à combiner au mieux pour tirer parti des avantages de chacun et ainsi produire des annotations de gènes plus fiables. On distingue quatre types de méthodes combinantes :

- la *combinaison linéaire*,
- l'*arbre de décision*,
- les *modèles de Markov cachés*,
- le *pipeline d'heuristiques*.

La combinaison linéaire est utilisée par les programmes LC1 et LC2 de la suite Combiner [27]. Ces programmes se fondent sur les signaux prédits par différents programmes (signaux d'initiation et de terminaison de la traduction, signaux donneur et accepteur d'épissage) pour déterminer les structures de gènes les plus probables d'une séquence. Ces structures sont vues comme les combinaisons acceptables optimales des signaux précédants et sont déterminées par programmation dynamique. A chaque segment exonique formé par une paire de signaux acceptables (ou pour LC2 par une extrémité d'alignement de séquence) est attibué un score par un vote (codant/non codant) de chaque programme en chacune de ses bases. Notons que ce score peut éventuellement être pondéré. Le score d'une structure de gène s'obtient alors par la somme des scores de chacun de ses exons. Il existe deux différences importantes entre LC1 et LC2 : tout d'abord LC1 combine exclusivement des sources *ab initio*, alors que LC2 autorise des sources de similarité et des programmes de prédictions de signaux, mais aussi LC1 utilise un vote non pondéré (correspondant à un poids de 1 pour chaque source), alors que LC2 peut attribuer un poids différent à chaque source. La complexité de LC1 est $\theta(mn^2)$ où m est le nombre de sources et n est le nombre de signaux détectés.

L' arbre de décision est utilisé par le programme SC de la suite Combiner pour combiner des sources d'annotation quelconques de façon plus fine que les combinateurs linéaires LC1 et LC2. Cette méthode se fonde sur une phase d'apprentissage et a été décrite en section 2.1.1 dans le cadre de la combinaison de mesures pour la détection d'exons.

Les modèles de Markov cachés (HMM) sont les combinateurs de sources les plus utilisés dans le domaine de l'annotation de gènes. Ils se subdivisent en deux catégories : les modèles de Markov cachés existants étendus pour prendre en compte la similarité de séquence, et les modèles de Markov cachés nouveaux spécifiquement dédiés à la combinaison de sources quelconque. Les premiers intègrent les résultats de similarité par conditionnement de leurs probabilités primitives, alors que les deuxièmes nécessitent un apprentissage différent pour chaque type de combinaison possible. Les programmes GenieHom [97], Twinscan [92, 68], Genomescan [145], Nscan [43], et Eugene'HOM [71] et Eugene_M [72], appartiennent à la

première catégorie et étendent respectivement les modèles de Markov cachés généralisés de Genie [96], Genscan [44] et Eugene [126]. Les programmes Digit [144], ExonHunter [40] et Jigsaw [28], appartiennent à la deuxième catégorie. Le programme Digit utilise une procédure Bayesienne pour inférer le score des exons et un HMM pour déterminer les structures de gènes à partir de ces exons. Il n'a été appliqué qu'à la combinaison de sources *ab initio*, en l'occurence aux programmes Fgenesh, Genscan et HMMgene. Les programmes Exonhunter et Jigsaw ont été appliqués à la combinaison de sources quelconques, et Jigsaw donne d'excellents résultats sur le génome humain.

Le pipeline d'heuristiques est le deuxième type de combinaison de sources le plus utilisé dans le domaine de l'annotation de gènes. Comme son nom l'indique il consiste en l'application successive de règles heuristiques portant sur différentes sources d'annotation afin de produire une annotation finale. Les heuristiques utilisées dépendent fortement des sources d'annotation sur lesquelles elles s'appliquent. Ces sources doivent donc être prédéfinies, ce qui rend les programmes fondés sur cette technique non génériques. Parmi de tels programmes citons :

- Genecomber [127], qui combine les sources *ab initio* Genscan et HMMgene,
- Ensembl [54], qui combine la source *ab initio* Genscan avec des sources de similarité portant sur des protéines et des ADNc,
- Eannot [59], qui combine des sources de similarité portant sur des protéines et des ADNc,
- PSEP [49], qui combine des sources de similarité portant sur l'ADN génomique et des ADNc de type ESTs,
- EGPred [86] qui combine des sources *ab initio*, des sources de similarité portant sur des protéines, et des prédicteurs d'introns et de signaux d'épissage.

Il faut noter que parmi ces quatre types d'approches combinantes, les plus utilisées sont les modèles de Markov cachés généralisés et les pipelines d'heuristiques.

Il faut également noter le développement récent d'un cadre pour la combinaison de programmes d'annotation : Gaze [84]. En effet ce cadre a l'avantage d'être capable de combiner des sources d'annotations *quelconques*, et est ainsi complètement générique. Pour cette raison et même s'il possède l'inconvénient de n'intégrer aucune connaissance biologique et donc de nécessiter une étape de paramétrage longue et délicate, nous pensons qu'il représente une approche intéressante et novatrice.

Pour conclure, nous avons répertorié les principaux avantages et limites des trois types de méthodes d'annotation de gènes dans le tableau 2.2.

Etant donné l'état de l'art dans le domaine de l'annotation de gènes, nous pensons qu'il est nécessaire de développer *un cadre formel pour la combinaison d'annotations*. De plus, nous pensons que ce cadre devra être suffisamment intuitif pour que son fonctionnement soit compris par des personnes de disciplines différentes. Comme nous l'avons vu dans ce

Méthodes	Avantages	Limites
Ab initio	– rapide – formel – systématique	– performances aléatoires – peu fiable seule
Par similarité de séquence	– robuste – fiable	– non exhaustive – lente – nécessite de fréquentes mises à jour
Combinante	– un peu meilleure que chacune des stratégoes précédentes prise séparément	– paramètres inexistants ou peu intuitifs

TAB. 2.2 – **Avantages et limites des différentes méthodes d'annotation de gènes.**

chapitre, les méthodes d'annotation de gènes qui se fondent sur l'utilisation de graphes sont nombreuses, et ce sont souvent les méthodes à la fois les plus formelles et les plus élégantes. D'autre part, les graphes permettent souvent une approche intuitive. Ainsi un cadre formel pour la combinaison d'annotations fondé sur les graphes pourrait être une approche fructueuse.

Chapitre 3

Annotation experte automatique de gènes

3.1 Les connaissances fondamentales de l'expert

Le gène idéal

L'expert en annotation est un biologiste qui connaît parfaitement les mécanismes de biologie moléculaire, et en particulier ceux qui conduisent à la synthèse d'une protéine à partir d'un gène (figure 2.1). Rappelons que nous nous interessons exclusivement ici aux gènes eucaryotes codant des protéines. Chez les eucaryotes supérieurs, un gène peut donner lieu à **plusieurs** protéines par deux processus :

- l'épissage alternatif, décrit en figure 3.1,
- le début de traduction alternatif.

Annoter un gène signifie donc pour l'annotateur :

- déterminer ses différents transcrits alternatifs[1],
- pour chaque transcrit alternatif déterminer :
  - le début et la fin[2] de chacun de ses exons,
  - s'il code ou non pour une protéine (un transcrit qui code pour une protéine est dit *codant*),
- pour chaque transcrit codant pour une protéine déterminer le début et la fin de sa séquence codante (CDS).

Chez les eucaryotes supérieurs, la structure des transcrits codants biologiquement validés possède le plus souvent les caractéristiques suivantes :

- une grande étendue génomique,
- des exons courts,
- des introns longs,
- un grand nombre d'exons,

[1] Plus exactement les empreintes de ses ARNm alternatifs sur la séquence à annoter.

[2] Ici et dans toute la suite du texte, début et fin sont relatifs au début de la séquence à annoter.

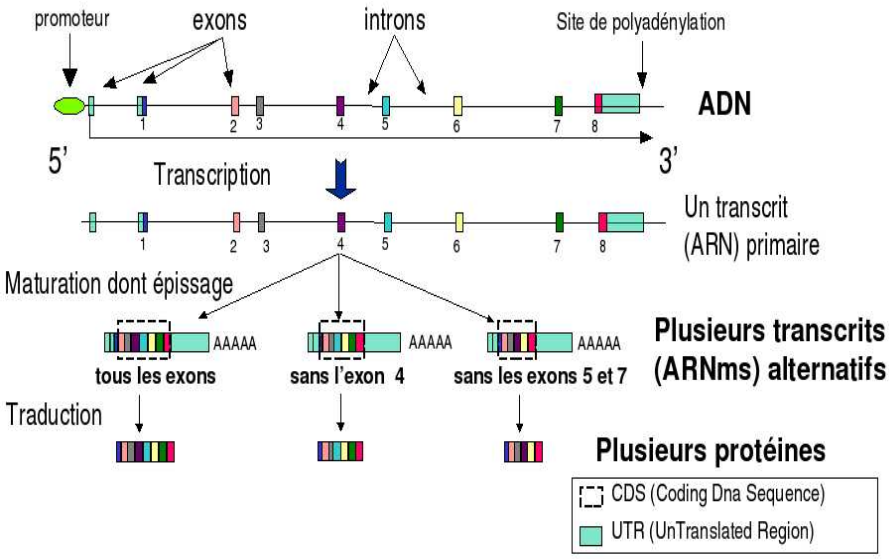


FIG. 3.1 – **Epissage alternatif.** La partie supérieure de cette figure représente une séquence d'ADN génomique eucaryote. Cette séquence comporte une structure de gène qui débute par un promoteur, est suivie d'une succession d'exons et d'introns, et se termine par un site de polyadénylation. Ce gène subit une étape de transcription qui produit un ARN primaire (parfois appelé pré-messager) qui a exactement la même structure que le gène présent sur l'ADN mais est dépourvu de promoteur. Ce transcrit primaire subit alors une étape de maturation incluant une sous-étape d'épissage qui excise les introns du gène et met les exons bout à bout. Un intron est bordé par des signaux donneur et accepteur d'épissage qui sont reconnus par la machinerie cellulaire comme les points entre lesquels une excision doit avoir lieu. Cet épissage peut avoir lieu de manière alternative c'est-à-dire générer plusieurs ARNm différents. Ceci est dû au fait que certains exons peuvent être excisés en même temps que les introns qui les bordent et donc ne pas apparaître dans l'ARNm résultant. Ici le gène initial donne lieu à trois ARNm alternatifs : celui de gauche dans lequel tous les introns ont été excisés et qui comporte donc tous les exons du gène intial, celui du milieu dans lequel l'exon 4 a été excisé en même temps que les introns 4 et 5, et celui de droite dans lequel l'exon 5 a été excisé en même temps que les introns 5 et 6 et l'exon 7 a été excisé en même temps que les introns 7 et 8. Chaque ARNm alternatif donne lieu à une protéine par traduction en acides aminées de sa séquence codante ou CDS.

- un CDS long,
- un grand nombre d'exons codants, dans l'absolu et relativement au nombre d'exons UTR ou non codants,
- un CDS débutant par un codon ATG et se terminant par un codon Stop.

Ainsi les annotateurs recherchent en priorité des structures de transcrits dotées de ces caractéristiques, structures que nous considérerons donc comme **idéales**.

Les ressources

Génération. Pour annoter les gènes d'une séquence d'ADN, l'annotateur utilise les résultats de programmes d'annotation. Ces annotations, appelées *ressources*, peuvent être les résultats de programmes d'annotation par similarité de séquence ou encore *ab initio* (voir chapitre 2), et peuvent être considérées comme des annotations de gènes préliminaires et imparfaites sur lesquelles l'annotateur s'appuie pour constituer sa propre annotation. Le travail de l'annotateur consiste donc en la **combinaison experte de ressources** dans le but d'annoter les transcrits d'une séquence de la façon la plus fiable possible.

En fonction de l'espèce étudiée, l'expert décide de quelles ressources il a besoin, et associe implicitement un degré de **confiance** à chacune d'elle. Le degré de confiance d'une ressource dépend à la fois du type de programme qui a permi de la générer, mais aussi des sources de données externes qu'elle a éventuellement nécessité (qualité des séquences, distance phylogénétique avec la séquence à annoter (voir annexe B, ...etc). Les ressources sont d'autant plus fondamentales dans le protocole d'annotation de l'expert que celui-ci se doit d'attacher à toute structure de transcrit qu'il annote, les ressources qui attestent de sa présence. Ces ressources sont autant de preuves que la structure de transcrit existe en cette position de la séquence, et permet de déterminer une **classe de confiance** pour le transcrit.

Le protocole d'annotation de l'expert est donc **semi-automatique** et implique que l'expert ait, en plus de ses connaissances en biologie, une connaissance approfondie :

- du fonctionnement des programmes qui ont permi de générer les ressources,
- de la signification des paramètres de ces programmes,
- des caractéristiques des sources de données externes éventuellement utilisées par ces programmes.

Visualisation. Suit alors une phase de visualisation des ressources qui viennent d'être générées. L'expert cherche en effet à voir comment se positionnent les ressources sur la séquence à annoter, et utilise pour cela des outils standard de visualisation tels que ACEDB [6] ou Apollo [7]. Ces outils représentent la séquence d'ADN de manière linéaire, et les ressources sous forme de boîtes de couleur le long de la séquence. La figure 3.2 donne un exemple de ce type de représentation. Notons que ces outils de visualisation servent également de bases de données puisqu'ils permettent d'associer diverses informations à chaque type de ressource. Pour une ressource de type alignement de séquence par exemple, les informations essentielles stockées par ces outils sont :

- le nom de la molécule dont provient l'alignement (M),
- les positions de début et de fin de cet alignement sur la séquence à annoter,
- les positions de début et de fin de cet alignement sur M.

Pour visualiser les informations relatives à une ressource il suffit de cliquer sur la boîte de couleur correspondante.

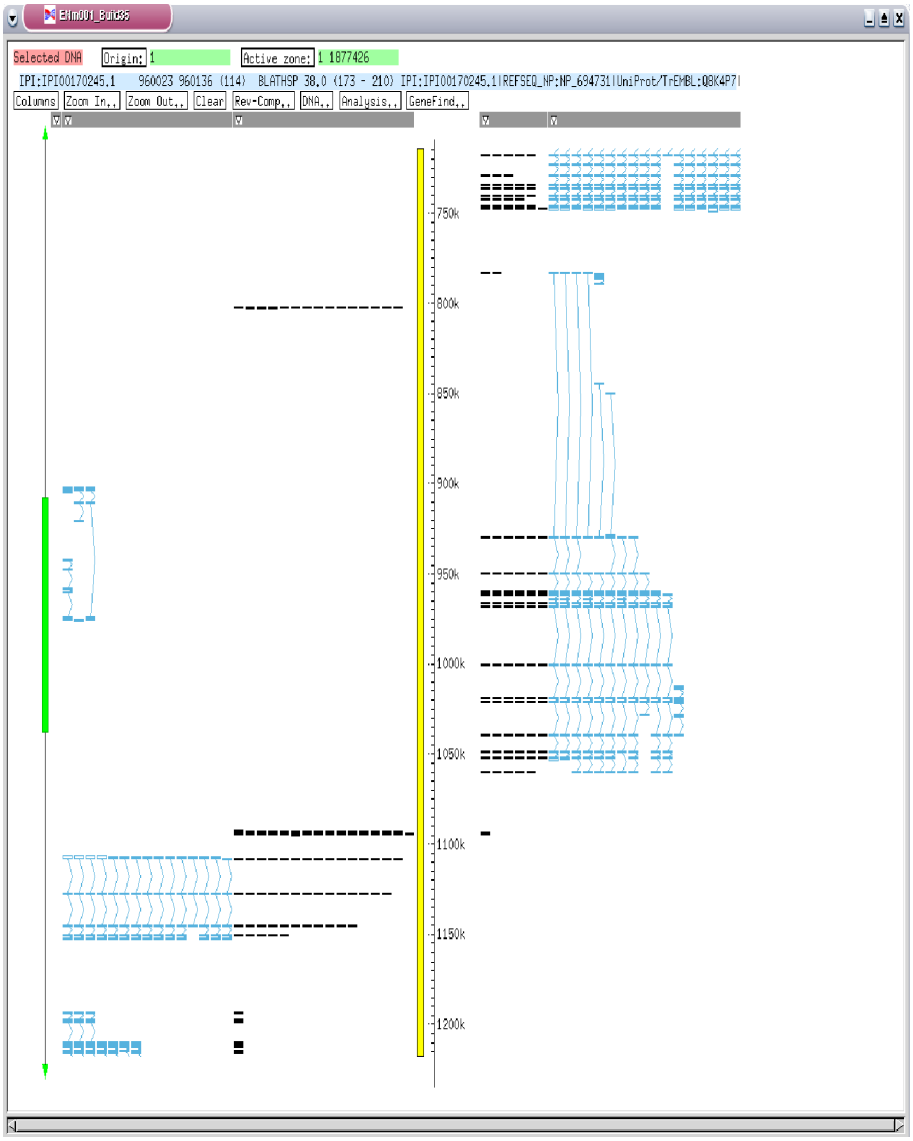


FIG. 3.2 – **Visualiser l'ADN et les ressources.** Cette figure est la capture d'écran d'une fenêtre ACEDB représentant une séquence d'ADN du génome humain ainsi que des ressources relatives à celle-ci : des alignements d'ADNc humains (en bleu, Genbank [35], voir annexe A), et des alignements de protéines de souris (en rouge, IPI [91], voir annexe B). La barre verticale centrale sépare les ressources relatives à la séquence d'ADN (à droite), des ressources relatives à la séquence complémentaire inverse de celle-ci (à gauche). Sa graduation indique la position où l'on se trouve sur la séquence, sur laquelle cet outil permet de se déplacer et d'effectuer des zooms.

Les résultats des programmes d'annotation *ab initio* représentent une ressource à faible valeur de confiance aux yeux des annotateurs en raison de leur tendance à sur-prédire.

Ainsi le travail des annotateurs porte essentiellement sur **l'analyse et la combinaison de ressources de type alignement de séquence**. Les alignements de séquences sont en général assez fiables, surtout dans le cas d'ADNc pleine longueur ou de protéines d'espèces proches, mais représentent des annotations partielles. En effet l'algorithme d'alignement utilisé est généralement de type local et heuristique, et ces alignements ne représentent donc que des **approximations** des exons de la séquence à annoter. De plus ces alignements dépendent très fortement des trois caractéristiques suivantes :

- le type de la séquence source,
- la distance phylogénétique entre la séquence source et la séquence à annoter (voir annexe B),
- les valeurs des paramètres utilisés par le programme d'alignement (voir annexe C pour le programme Blast [29]).

Blast [29] est le programme d'alignement local heuristique le plus utilisé par les annotateurs humains, et produit des alignements locaux appelés *HSP* pour High-scoring Segment Pair. Ainsi nous utiliserons le terme HSP pour désigner tout résultat de programme d'alignement local.

Démarche générale

En analysant le protocole d'annotation de l'expert, nous avons constaté qu'il visualise les ressources d'abord de manière *globale* puis de manière *locale*. La visualisation globale lui permet en effet de délimiter les différents gènes de la séquence à annoter sur la base du chevauchement global des ressources (figure 3.2). Puis il se focalise sur un premier gène pour l'analyser plus finement (figure 3.3). Du fait de l'orientation de la transcription de 5' en 3', l'annotateur parcourt généralement les gènes d'une séquence de 5' en 3'.

Très schématiquement, l'analyse d'un gène consiste en une **succession de comparaisons et de regroupements d'HSPs** (d'ADNc et de protéines), sur la base d'informations prouvant qu'ils **appartiennent à une même structure de transcrit**. Ces informations peuvent être de natures aussi diverses que le type et le nom de la molécule dont deux HSPs proviennent, leur position sur la séquence à annoter, ou encore les séquences qui les bordent ou qu'ils contiennent. Ainsi l'annotation de gènes experte est un processus semi-automatique dont on peut dire qu'il fonctionne par **renforcement d'hypothèse**.

Plus précisément la démarche de l'annotateur peut être vue comme l'utilisation d'un ensemble de règles heuristiques, qu'il combine de manière heuristique, et qu'il applique à des résultats de programmes d'alignements heuristiques entre la séquence à annoter et des séquences sources qu'il a choisies (type et espèce) de manière heuristique. Le concept d'heuristique est donc fondamental dans le protocole d'annotation de l'expert.

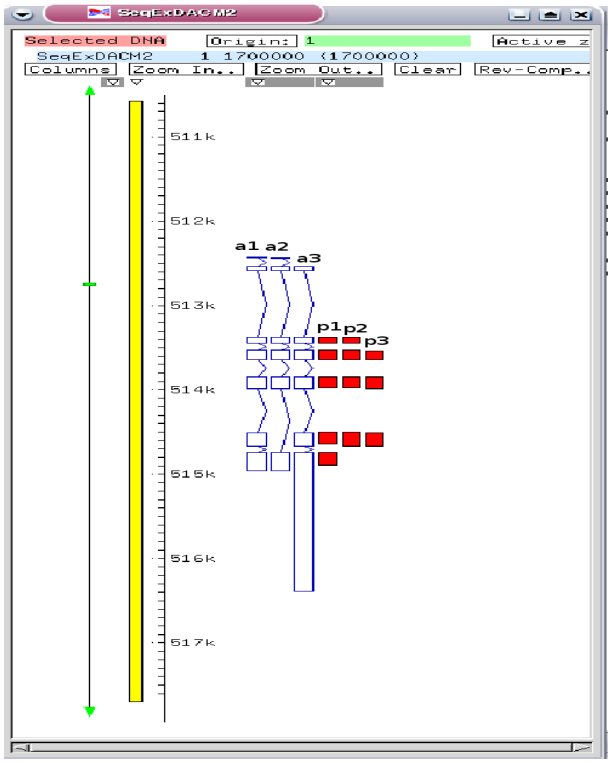


FIG. 3.3 – **Ressources relatives à un gène.** Cette figure est la capture d'écran d'une fenêtre ACEDB représentant les ressources relatives à un gène d'une séquence d'ADN génomique humain : les alignements a_1 à a_3 de trois ADNc humains en bleu, et les alignements p_1 à p_3 de trois protéines de souris en rouge. Chacun de ces alignements se compose de plusieurs HSPs représentés par des boîtes (a_1 se compose par exemple de sept HSPs), et éventuellement reliés entre eux (ici les alignements d'ADNc sont reliés entre eux, mais pas les alignements de protéines).

3.2 Nécessité d'un cadre d'expression pour l'expertise humaine

Etant donné que l'expert humain reste la référence en matière d'annotation, nous nous sommes intéressés à automatiser son protocole. Nous nous sommes donc interrogés sur la manière dont l'expert procède pour regrouper les HSPs initaux jusqu'aux structures de transcrits de la séquence à annoter. Cette étude nous a montré que formaliser et automatiser le protocole d'annotation de l'expert pose de nombreux problèmes, que nous allons tenter d'énumérer aussi exhaustivement que possible.

Evolution de l'analyse. L'ensemble des règles heuristiques combinées par l'expert, la manière dont il les combine, et le type et l'espèce dont les HSPs sont originaires et sur lesquels

il applique ces règles peuvent changer au cours du temps. Changer peut aussi bien signifier être modifié ponctuellement qu'être ajouté ou supprimé totalement. En effet certaines règles heuristiques dépendent de l'organisme étudié, comme par exemple le choix des ressources ou encore certains paramètres impliqués dans ces règles. De plus au cours de l'annotation d'une espèce donnée un annotateur n'a pas la même expertise après la première séquence annotée qu'après la centième. Il apprend sur cette espèce au fur et à mesure de ses annotations et lectures, et utilise donc des règles différentes au cours du temps. Le cadre formel choisi pour automatiser le protocole d'annotation de l'expert, doit ainsi permettre une **grande flexibilité** dans l'expression des règles heuristiques.

Cadre expressif. Un autre problème vient de ce que les règles heuristiques utilisées par l'expert ne sont pas toujours clairement enoncées par ce dernier. Pour les formaliser, il faut donc d'abord comprendre ces règles, puis les confronter les unes aux autres pour vérifier qu'elles ne sont ni contradictoires, ni trop redondantes entre elles, et enfin les présenter en retour à l'expert pour vérifier que leur formulation n'a pas entraîné d'erreurs. Un dialogue permanent et une interaction très forte sont donc nécessaires entre les représentants des deux disciplines. Le cadre formel développé doit donc permettre une expression aisée des différentes règles, la plus proche possible de leur énoncé initial. Ceci implique un cadre formel d'expression des règles qui soit à la fois **expressif, explicatif et intuitif**.

Hétérogénéité des sources et des règles. Le type (ADNc pleine longueur, ESTs, protéines) et l'espèce (mammifères, vertébrés, procaryotes, ..etc) des HSPs utilisés sont divers. Il faut donc un système capable de gérer des **données hétérogènes**. D'autre par les règles heuristiques peuvent concerner les phases de comparaison ou de regroupement d'HSPs, porter sur des propriétés structurelles ou de priorité des différents types d'HSPs, ou encore dépendre ou non d'un paramètre. Les règles heuristiques sont donc de différents types, ce qui implique la nécessité d'un cadre d'expression homogène pour les représenter. Le cadre développé doit ainsi être à la fois **générique et incrémental**.

En résumé un cadre formel permettant la combinaison de règles heuristiques sur les alignements de séquences, doit avoir les qualités suivantes :

- être extrêmement flexible,
- être expressif, explicatif et intuitif,
- être générique et incrémental.

Parce qu'il possède toutes ces qualités, nous pensons qu'un **langage d'annotation experte par mise en relation et regroupement d'HSPs**, est particulièrement approprié. Ce langage décrira le raisonnement de l'expert, et reposera sur une structuration de ce raisonnement.

3.3 DACM et langage

Afin de concevoir un cadre formel pour l'expression de l'expertise humaine dans le domaine de l'annotation des gènes, nous avons analysé le protocole d'annotation de l'expert du point de vue de la **méthode**. Nous décrivons donc dans un premier temps le résultat de cette analyse, i.e. la démarche de l'annotateur (sous-section 3.3.1), et dans un deuxième temps la façon dont nous avons choisi de l'automatiser (sous-section 3.3.2).

3.3.1 La démarche de l'annotateur.

Pour illustrer la démarche de l'annotateur, nous prenons l'exemple de l'alignement d'un ensemble d'ADNc humains (Genbank [35]) et de protéines de souris (IPI [91]) sur une séquence d'ADN humain. La capture d'écran d'une fenêtre ACEDB représentant les HSPs résultant de cet alignement est donnée en figure 3.3. Les HSPs d'ADNc sont représentés en bleu foncé et les HSPs de protéine en rouge. L'appartenance de deux HSPs à une même molécule est matérialisée par un lien physique pour les HSPs d'ADNc, et se déduit de la position verticale des HSPs pour les HSPs de protéine. On suppose que l'ensemble des HSPs provenant d'une même molécule (ADNc ou protéine) fait partie de la même structure de transcrit, et on nomme a_1 à a_3 les modèles de transcrits mono-ADNc (i.e. provenant du même ADNc), et p_1 à p_3 les modèles de transcrits mono-protéines (i.e. provenant de la même protéine) correspondant à ces structures (figure 3.3).

L'annotateur humain observe et compare les modèles de transcrits mono-molécules, type de ressource par type de ressource, de la plus fiable à la moins fiable. La fiabilité d'une ressource est fonction à la fois du type des molécules qui ont servi à la générer (ADNc pleine longueur, EST, protéine), et de la distance phylogénétique (voir annexe B) entre l'espèce dont ces molécules proviennent et l'espèce dont la séquence à annoter provient. Dans notre exemple, l'ADN à annoter provient de l'espèce humaine, par conséquent l'annotateur analyse d'abord les modèles de transcrits de type *ADNc humain*, puis les modèles de transcrit de type *protéine de souris*. Ces ressources sont analysées de 5' en 3', donc de haut en bas dans ACEDB. A chaque type de ressource l'annotateur associe un ensemble de relations qu'il juge pertinent de tester sur les HSPs de cette ressource. Prenons par exemple les modèles de transcrits mono-ADNc (ici a_1 à a_3), l'annotateur teste sur ceux-ci deux relations : le chevauchement[3] sur la séquence d'ADN, ou *chevauchement génomique*, et l'identité des introns chevauchants.

Comme a_1 et a_2 se chevauchent mais possèdent des introns chevauchants différents (un intron de a_2 chevauche deux introns de a_1), l'annotateur les sépare. Puis il compare a_3 aux représentants a_1 et a_2 des deux classes ainsi formées. a_3 chevauche a_1 et a_2, mais n'a d'introns chevauchants identiques qu'avec a_1, par conséquent l'annotateur met a_3 dans la même classe

[3]Deux modèles de transcrits se chevauchent si leurs étendues génomiques se chevauchent. L'étendue génomique d'un modèle de transcrit est l'intervalle allant de sa borne la plus en 5' à sa borne la plus en 3'.

que a_1. Ainsi une expertise consiste à diviser les modèles de transcrits mono-ADNc a_1 à a_3 en deux classes distinctes : l'une qui inclut a_1 et a_3, et l'autre qui inclut a_2. L'expert procède exactement de la même façon pour les modèles de transcrits mono-protéines en utilisant un autre ensemble de relations, toutefois cette action n'est pas détaillée ici.

Nous observons donc la **succession de deux actions** chez l'expert :

1. une mise en relation de modèles de transcrits,
2. un regroupement de ces modèles de transcrits en classes sur la base des relations précédamment établies entre eux.

De manière générale, nous pouvons affirmer que *l'expert établit des liens entre des ensembles d'alignements de séquences, dans le but de regrouper ceux qui représentent la même structure de transcrit*. Si l'on nomme *modèle de transcrit* ou *transmod*, un ensemble d'alignements de séquences qui représentent la même structure de transcrit, la démarche de l'expert se modélise particulièrement bien par des **multi-graphes colorés** où :

- les **sommets** sont des **transmods**,
- les **multiples couleurs** entre les sommets sont les multiples **relations** que l'expert juge utile d'établir entre ces transmods.

Sur notre exemple, le multi-graphe coloré des modèles de transcrits mono-ADNc a_1 à a_3 aurait trois sommets, un pour chacun des modèles de transcrits, et deux couleurs d'arêtes, les relations *chevauchement* et *introns_ chevauchants_ identiques*. Il est donné en figure 3.4.

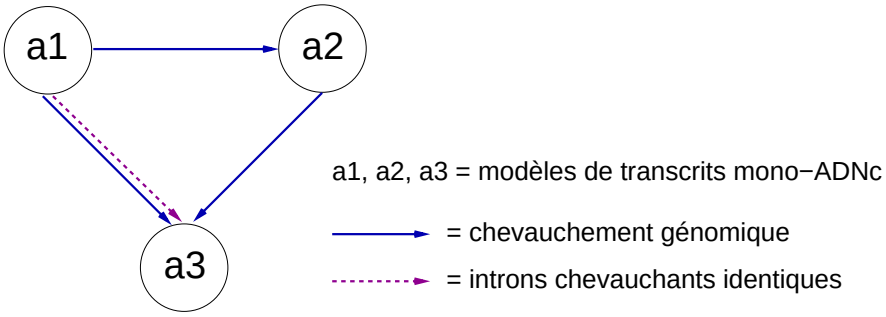


FIG. 3.4 – **Multi-graphe coloré.** Cette figure représente le multi-graphe coloré que l'on obtiendrait à partir des modèles de transcrits mono-ADNc a_1 à a_3 de notre exemple. Chaque tel modèle donne lieu a à un sommet du graphe, et chacune des deux relations «chevauchement génomique» et «introns chevauchants identiques» à une couleur d'arête du graphe. Une couleur d'arête n'est établie entre deux sommets que si les transmods représentés par ces sommets sont ordonnés et que la relation correspondant à cette couleur est vraie entre ces transmods. Ainsi la couleur d'arête «chevauchement génomique» est mise entre les sommets a_1 et a_2, a_1 et a_3, et a_2 et a_3, mais la couleur «introns chevauchants identiques» n'est mise qu'entre a_1 et a_3.

Il est alors nécessaire de modéliser le processus de regroupement faisant suite à la mise en relation effectuée par l'expert. A cette étape nous disposons d'un multi-graphe coloré dont les sommets sont des transmods, et dont les couleurs d'arêtes sont les relations établies par l'expert entre ces transmods. En fait, ce processus de *regroupement* se modélise par un mécanisme de **réduction** du graphe. Informellement *réduire un graphe* signifie définir des chemins dotés de certaines couleurs (d'arêtes et de sommets) dans ce graphe, et constituer une classe d'objets (de transmods) pour chacun de ces chemins.

Nous nous sommes rendu compte que le processus de mise en relation/réduction *apparaissait à différents étapes du protocole d'annotation de l'expert*. Puisque nous cherchons à automatiser l'ensemble de ce protocole, il est donc particulièrement commode de considérer que la réduction d'un multi-graphe coloré produit un nouveau multi-graphe coloré. En effet celui-ci, appelé *graphe réduit*, peut à son tour subir un processus de mise en relation/réduction. Pour illustrer cette notion, nous avons représenté le multi-graphe coloré obtenu par réduction du graphe de la figure 3.4 selon les couleurs d'arêtes *chevauchement* et *introns_ chevauchants_ identiques* (figure 3.5). Chaque sommet du graphe réduit corres-

FIG. 3.5 – **Réduction de multi-graphe coloré.** Cette figure représente le graphe résultant de la réduction du multi-graphe coloré des modèles de transcrits mono-ADNc de notre exemple. Comme cette réduction se fait sur la base de la présence des couleurs d'arêtes «chevauchement génomique» **et** «introns chevauchants identiques», les sommets de ce graphe sont au nombre de deux : $a_1 - a_3$ et a_2. Notons que chacun de ces sommets représente un regroupement expert de modèles de transcrits mono-ADNc.

pond à une classe d'objets (de transmods) déterminée par l'expert, à savoir ici la classe de a_1 et a_3 d'une part, et la classe de a_2 d'autre part. Il faut noter que dans le cas général, un même objet (transmod) peut faire partie de plusieurs classes différentes.

Notons également deux caractéristiques importantes des multi-graphes colorés que nous utilisons :

- ils sont **orientés**. La transcription est orientée de 5' en 3' le long de la molécule d'ADN (voir figure 2.1), de ce fait l'analyse des ressources et les relations entre les objets biologiques (exons, transcrits,...) se fait dans une orientation bien précise,
- les sommets portent des **couleurs**. Chaque sommet représente un transmod c'est-à-dire un ensemble d'alignements de séquences. Si les différents alignements d'un transmod sont originaires du même type de ressource r (ici ADNc ou protéine), alors le transmod

est dit de type r et le sommet du graphe qui lui est associé porte la couleur r.
Ainsi les multi-graphes colorés que nous utilisons sont appelés **DACMs** comme **Directed Acyclic Coloured Multigraphs**. Pour une définition plus formelle des DACMs et du processus de réduction, voir chapitre 4.

En résumé les deux étapes-clés du protocole d'annotation de l'expert peuvent chacune se voir comme une action portant sur un DACM :

1. la comparaison et la mise en relation de transmods : par l'ajout de couleurs d'arêtes sur le DACM dont les sommets sont ces transmods,
2. le regroupement de transmods selon certaines de ces relations : par la réduction de ce DACM selon certaines couleurs d'arêtes.

De plus, afin que cette succession d'étapes puisse avoir lieu de manière répétée, *chaque réduction de DACM donne lieu à un nouveau DACM.*

Toutes ces conclusions nous ont naturellement conduit à la conception d'un langage dédié à l'annotation de gènes experte : le langage DACMLang. Plus précisément, un programme en DACMLang est une suite de commandes, et une commande modélise une double-action de réduction/mise en relation. La suite de cette section est dédiée à la description informelle d'une commande de DACMLang, pour une formalisation complète voir chapitre 4.

3.3.2 Le langage DACMLang

Une commande de DACMLang suppose l'existence d'un DACM d et est de la forme :

```
Réduction de d en D selon certaines couleurs d'arêtes

:>

Ajout de couleurs d'arêtes sur D réduit de d
```

Réduction d'un DACM. La sous-commande de réduction d'un DACM d selon certaines couleurs d'arêtes correspond à l'action de regroupement des transmods effectuée par l'expert. Comme les transmods regroupés par l'expert le sont sur la base des relations précédemment établies entre eux, la sous-commande de réduction du DACM d se définit par une expression sur les couleurs d'arêtes de d. Plus précisément la sous-commande de réduction de d est constituée des cinq types d'éléments suivants :

- des identifiants de couleurs d'arêtes,
- des identifiants de couleurs de sommets,
- les opérateurs logiques `&&`, `||` et `,` sur les expressions sur les couleurs d'arêtes (note : `,` est équivalent à `||`),
- l'opérateur de composition `.` d'expressions sur les couleurs d'arêtes,

– l'opérateur d'extension + d'expressions sur les couleurs d'arêtes.

Pour illustrer la sous-commande de réduction du DACM d, nous reprenons l'exemple du DACM des modèles de transcrits mono-ADNc donné plus haut (figure 3.4). La sous-commande de réduction de ce DACM est la suivante :

```
([Adnc].(chevauchement && introns_chevauchants_identique)+) || ([Adnc])
```

car elle signifie : « chercher l'un des deux types de chemins suivants :
- des chemins qui commencent par un sommet de couleur ADNc (`[Adnc]`) et constitués de la plus longue (`+`) suite d'arêtes possédant à la fois (`&&`) les couleurs *chevauchement* et *introns_ chevauchants_ identiques*,
- des chemins réduits à un sommet de couleur ADNc (`[Adnc]`). »

Ajout de couleurs d'arêtes sur le DACM réduit. La sous-commande d'ajout de couleurs d'arêtes sur D réduit de d correspond à l'action de comparaison et de mise en relation de transmods effectuée par l'expert. Elle se compose d'une succession de filtres dont chacun représente l'ajout d'une couleur d'arête particulière sur D. Comme D est un DACM réduit de d, les sommets de D correspondent à des chemins de sommets de d, et chaque filtre de la sous-commande d'ajout de couleurs d'arêtes sur D est donc de la forme :

```
| condition sur deux chemins de d -> couleur d'arête de D
```

Un *transmod* est un ensemble d'alignements de séquences obtenus par comparaison de la séquence à annoter avec des molécules codantes épissées telles que des ADNc ou des protéines. Un transmod peut donc se voir comme un ensemble d'entiers : les positions des alignements qui le constituent sur la séquence à annoter. Ainsi, nous représentons un transmod par une suite d'intervalles ordonnés de positions de la séquence à annoter, qui représentent les ensembles maximaux de positions contigues où ces alignements ont eu lieu, et qui peuvent donc se voir comme des *modèles d'exons* de cette séquence (voir figure 3.6 pour un exemple de transmod, et chapitre 4 pour une définition formelle).

Ainsi un ensemble de transmods est un transmod (l'union d'un ensemble de positions est un ensemble de positions), et les sommets de D, qui sont des chemins de d, modélisent des transmods. La condition d'un filtre pour l'ajout de couleurs d'arêtes entre les sommets de D, peut donc se voir comme une expression sur les transmods modélisés par les sommets de D. Plus précisément cette condition s'exprime par des opérateurs de deux types :
- des opérateurs de **création** de transmod : portent sur deux transmods et produisent un transmod,
- des opérateurs **logiques** : portent sur deux transmods et produisent un booléen.

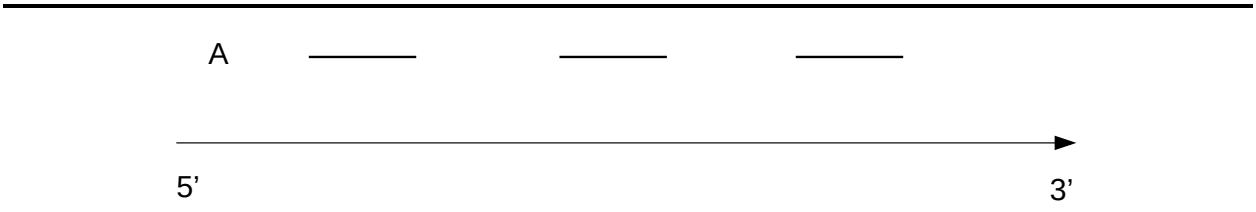


FIG. 3.6 – **Un transmod ou modèle de transcrit.** Cette figure représente une séquence d'ADN génomique (flèche de 5' en 3'), ainsi qu'un transmod A de cette séquence. Ce transmod est matérialisé par trois traits horizontaux, dont chacun représente un ensemble maximal de positions contigues de ce transmod, ou intervalle, et peut se voir comme un modèle d'exon de A.

Les opérateurs de DACMLang de ces deux catégories sont listés et décrits dans le tableau 3.1, et les moins intuitifs d'entre eux décrits dans la suite du texte.

Il existe trois types de *chevauchement* pour les transmods :
- le chevauchement *faible* (opérateur `>>-`) : correspond au chevauchement des étendues génomiques de deux transmods,
- le chevauchement *fort* (opérateur `>>+`) : correspond à l'existence d'une position commune entre les intervalles de deux transmods,
- le chevauchement *interne* (opérateur `>>=`) : correspond au fait qu'aucun des intervalles du deuxième transmod n'est totalement inclus dans un intervalle du complément[4] du premier transmod (prédicat non symétrique).

Ces trois types de chevauchement sont illustrés en figure 3.7 page 65.

L'*identité des introns chevauchants* de deux transmods (opérateur `><`) n'a d'intérêt que pour des transmods chevauchants (dans les autres cas, cette propriété est vrai), et est illustrée sur un exemple en figure 3.8.

Pour illustrer la sous-commande d'ajout de couleurs d'arêtes sur le DACM D réduit de d, nous supposons avoir effectué la sous-commande de réduction de d, et donc disposer des sommets de D. De plus nous supposons vouloir ajouter la couleur d'arête *chevauchement* entre deux sommets de D dont les transmods sont ordonnés et d'étendue génomique chevauchante. En DACMLang cela s'écrit :

```
5' >>- 3' -> chevauchement
```

Les DACMs et le langage DACMlang permettent de modéliser la partie centrale et essentielle de l'annotation experte : la façon dont les experts passent des HSPs aux modèles de

[4]Le complément d'un transmod correspond aux intervalles complémentaires de ce transmod qui sont inclus dans son étendue génomique, et représente donc l'ensemble des modèles d'introns de ce transmod.

Opérateurs de création de transmod	Opérateurs logiques
. `<>` : transmod vide, . `5'` : transmod le plus en amont parmi deux, . `5'[C]` : transmod le plus en amont parmi deux et de couleur dans `C`, . `3'` : transmod le plus en aval parmi deux, . `3'[C]` : transmod le plus en aval parmi deux et de couleur dans `C`,	. `<*` : un transmod se termine strictement avant un autre, . `<=*` : un transmod se termine avant un autre, . `>*` : un transmod se termine strictement après un autre, . `>=*` : un transmod se termine après un autre, . `==` : égalité de deux transmods, . `>>-` : chevauchement faible de deux transmods, . `>>+` : chevauchement fort de deux transmods, . `>>=` : chevauchement interne de deux transmods, . `><` : identité des introns chevauchants de deux transmods, . `&&` : et, . `||` : ou, . `!` : non.

TAB. 3.1 – **Opérateurs pour l'ajout de couleurs d'arêtes sur le DACM réduit.**

transcrits de la séquence à annoter. Toutefois les DACMs ne permettent pas (pour l'instant !) de modéliser toutes les étapes du protocole d'annotation de l'expert, comme par exemple la recherche du CDS, et l'annotation experte par DACM doit donc s'intégrer dans un environnement plus complet. Dans la suite nous décrivons un exemple de tel environnement : *Exogean* comme *Expert on gene annotation*, décrit en figure 6.1 page 126.

Dans la suite nous décrivons les différentes étapes d'Exogean : d'abord l'«annotation experte par DACM» (section 3.4), puis les autres étapes d'annotation experte, globalement désignées par le terme *annotation experte annexe* (section 3.5).

3.4 Annotation experte par DACM

L'annotation experte par DACM d'Exogean construit les structures de transcrits des gènes de la séquence à annoter, à partir d'HSPs d'ADNc de la même espèce et d'HSPs de protéines d'une autre espèce. Pour cela, elle utilise trois étapes dont chacune repose sur la construction et la réduction d'un DACM, appelés DACM_1, DACM_2 et DACM_3 (voir figure 6.1

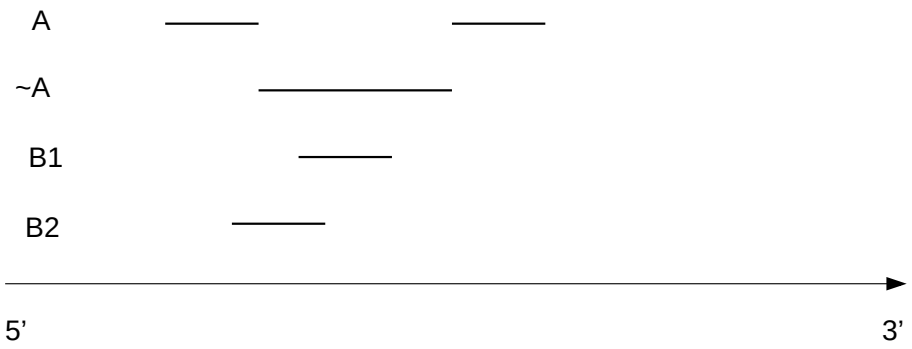


FIG. 3.7 – **Chevauchement de transmods.** Cette figure représente quatre transmods : A, i.e. le transmod issu du complément de A, B_1 et B_2. Les trois transmods A, B_1 et B_2 ont des étendues génomiques chevauchantes : ils se chevauchent de façon faible. Comme les intervalles de A n'ont pas de position commune avec les intervalles de B_1 mais ont des positions communes avec les intervalles de B_2, A et B_2 se chevauchent de façon forte, mais pas A et B_1. Enfin il existe un intervalle de B_1 totalement inclus dans un intervalle de , ce qui n'est pas le cas pour B_2 : A et B_2 se chevauchent de façon interne, mais pas A et B_1.

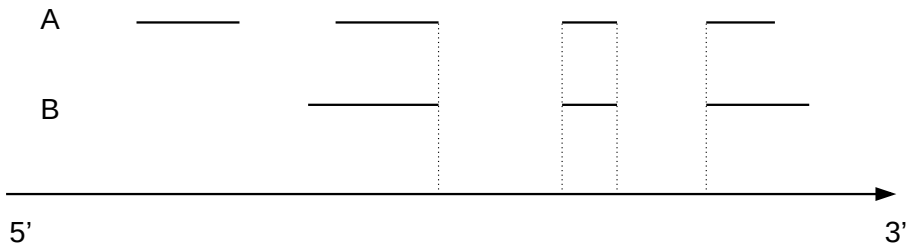


FIG. 3.8 – **Identité des introns chevauchants.** Cette figure représente deux transmods A et B d'une séquence d'ADN génomique. Les traits verticaux pointillés matérialisent la position des introns sur la séquence d'ADN. Le transmod A possède 3 introns et le transmod B deux introns. Les deux introns les plus en 3' de A et B se chevauchent respectivement et sont identiques. Aet B vérifient donc la relation d'identité des introns chevauchants.

page 126). Pour un i donné de $[1:3]$, $\text{DACM}_{(i+1)}$ est issu de la réduction de DACM_i, et modélise ainsi des modèles de transcrits «plus complexes» que DACM_i (voir section 3.3). Dans cette section, nous décrivons et donnons les raisons biologiques de la construction et de la réduction de ces trois DACM successivement. Il faut noter que la transcription de l'ADN en ARN est un processus orienté du 5' vers le 3' de l'ADN (voir figure 2.1). Ainsi, les arêtes colorées de nos DACMs, qui correspondent aux relations utilisées par l'expert entre les modèles de transcrits de ce DACM, sont également **orientées**. Nous utiliserons

indifféremment les termes *modèle de transcrit* et *transmod*.

3.4.1 $DACM_1$

La première étape de l'annotation experte agit sur des HSPs d'ADNc de la même espèce et de protéines d'une autre espèce (*modèles de transcrits de niveau* 1). Rappelons qu'un HSP est le résultat d'un alignement heuristique local et sans indels entre la séquence à annoter et la séquence d'une molécule codante épissée (ADNc ou protéine, voir figure 3.10 page 80).

Cette première étape consiste en la comparaison et le regroupement d'HSPs provenant d'une même molécule (ADNc ou protéine), et dont l'empreinte sur la séquence à annoter témoigne d'une même structure de transcrit. Les objets biologiques résultant de ce regroupement sont appelés *modèles de transcrits mono-molécules*, ou encore *modèles de transcrits mono-ADNc et mono-protéines* si l'on distingue le type de la molécule d'origine (*modèles de transcrits de niveau* 2).

Justification. Considérons le cas simple d'une séquence d'ADN génomique S ne contenant qu'un seul gène. Si l'épissage de ce gène produit l'ARNm a, et que a correspond à l'un des ADNc comparés à S, alors les HSPs provenant de cet ADNc forment sur S une structure de transcrit biologiquement valide. La situation est à peu près identique dans le cas où S est comparée à la protéine d'un gène orthologue du gène qu'elle contient (donc une protéine d'une autre espèce) : les HSPs de cette protéine forment le plus souvent sur S une structure de transcrit biologiquement valide[5].

Or la séquence à annoter ne contient pas un seul gène mais plusieurs, voire l'ensemble des gènes d'une espèce si l'on cherche à annoter un génome entier. Or à l'échelle du génome, des *duplications* de gènes peuvent être observées qui compliquent beaucoup la situation idéale décrite pour une séquence ne comportant qu'un gène (voir annexe B). En effet si l'on considère systématiquement tous les HSPs d'une même molécule épissée comme faisant partie de la même structure de transcrit, il est possible que l'on considère plusieurs gènes paralogues (appartenant à la même espèce et issus d'un même gène ancestral par duplication, voir annexe B) comme faisant partie de la même structure de transcrit.

Description des relations. Pour savoir si des HSPs originaires d'une même molécule épissée appartiennent à la même structure de transcrit, les experts utilisent trois relations sur les HSPs :

- l'appartenance de deux HSPs à une même molécule : *même_molécule*,
- le fait que deux HSPs provenant d'une même molécule et ordonnées sur la séquence à annoter sont séparés par une distance génomique de 5' en 3' inférieure à la taille maximale d'un intron : *taille_int_max*,

[5] Cette structure de transcrit peut ne pas être biologiquement valide, dans le cas où certains exons du gènes ne sont pas conservés entre les deux espèces, ou si la protéine source contient un domaine répété, cependant nous considérons ces cas comme mineurs.

– le fait que deux HSPs provenant d'une même molécule M et ordonnés sur la séquence à annoter sont également ordonnés sur M : *ordre_ gx_ molécule*.

Illustration des relations. La figure 3.11 page 81 illustre la relation *taille_ int_ max*, et la figure 3.12 page 82 la relation *ordre_ gx_ molécule*.

Réduction en DACMLang. Dans le langage DACMLang, $DACM_1$ est supposé être construit à partir des HSPs d'ADNc et de protéines initiaux ainsi que des connaissances que l'expert possède sur ces objets : chaque HSP donne lieu à un sommet de $DACM_1$ et chacune des trois relations précédentes à une couleur d'arête. Nous voulons réduire $DACM_1$ selon les trois couleurs d'arêtes à la fois (`&&`) et par les plus longs chemins d'arêtes (`+`). Ainsi la sous-commande de réduction de $DACM_1$ s'écrit en DACMLang :

```
(même_molécule && taille_int_max && ordre_gx_molécule )+
```

Le DACM obtenu par réduction de $DACM_1$ est appelé $DACM_2$.

A ce stade $DACM_2$ possède des sommets, mais ceux-ci ne sont reliés entre eux par aucune couleur d'arête. Les couleurs d'arêtes de $DACM_2$ correspondent aux relations que l'expert juge pertinent d'établir entre les modèles de transcrits mono-molécules.

3.4.2 $DACM_2$

La deuxième étape de l'annotation experte agit sur des modèles de transcrits mono-molécules. Elle consiste en la comparaison et le regroupement de modèles de transcrits mono-molécules de même type (ADNc ou protéine), qui représentent la même structure de transcrit de la séquence à annoter. Les objets biologiques résultant de ce regroupement sont appelés *modèles de transcrit multi-molécules de même type*, ou *modèles de transcrits multi-ADNc et multi-protéines* si l'on distingue le type de la molécule d'origine (*modèles de transcrits de niveau* 3).

Justification. Il arrive fréquemment que les modèles de transcrits mono-molécules de même type soient redondants et/ou complémentaires. Cette étape a donc le double but de supprimer la redondance de certaines ressources, et de permettre à d'autres de se compléter. Eliminer la redondance des ressources permet de ne pas annoter plusieurs fois le même transcrit partiellement, alors qu'exploiter la complémentarité des ressources permet d'annoter les transcrits les plus complets possibles.

Le regroupement des modèles de transcrits mono-molécules de même type s'effectue séparément pour chaque type de ressource. En effet les ADNc de la même espèce et les protéines d'une autre espèce ne sont ni aussi informatives ni aussi fiables. Un alignement d'ADNc de la même espèce que la séquence à annoter indique en effet quasiment toujours des bornes d'épissage exactes, ce qui n'est pas le cas d'un alignement de protéine d'une autre espèce.

Regrouper les ressources de type ADNc et protéine à ce stade provoquerait une perte d'information concernant les bornes d'épissage, et empêcherait en particulier de savoir si un modèle de transcrit mono-ADNc en *étend* un autre. Par extension on entend le fait de représenter la même structure de transcrit alternative du gène, mais en apportant des nucléotides ou des exons supplémentaires en 3'.

De plus les alignements d'ADNc indiquent de façon précise les différents transcrits alternatifs des gènes de la séquence à annoter, ce qui n'est pas le cas des protéines d'une autre espèce. Ces deux constatations nous conduisent à des règles de regroupement plus fines pour les ADNc de la même espèce que pour les protéines d'une autre espèce.

Description des relations. Pour savoir si deux modèles de transcrits mono-molécules de même type appartiennent à la même structure de transcrit, l'expert utilise trois relations portant sur les modèles de transcrits mono-molécules :

- le fait que deux modèles de transcrits mono-ADNc chevauchants et ordonnés ont leurs introns chevauchants identiques et que la fin du deuxième est en 3' de la fin du premier : *extension*,
- le fait que deux modèles de transcrits mono-ADNc chevauchants et ordonnés ont leurs introns chevauchants identiques et que la fin du deuxième est en 5' de la fin du premier : *inclusion*,
- le chevauchement de l'étendue génomique de deux modèles de transcrits mono-protéines : *chevauchement*.

Notons que ces trois relations ne concernent que des modèles de transcrits mono-molécules de même type ordonnés sur la séquence à annoter.

Rappelons que les alignements d'ADNc de la même espèce indiquent des bornes d'épissage **exactes**. Cela jusitifie la nécessité de regrouper des modèles de transcrits mono-ADNc chevauchants dont les introns chevauchants sont **identiques**. Les relations *extension* et *inclusion* sont utilisés par l'algorithme *ClusterMerge* du programme ESTGenes [66] pour inférer l'ensemble minimal des transcrits non redondants compatibles avec la structure épissée d'un ensemble d'alignements d'ESTs sur un génome.

Les alignements de protéines d'une autre espèce ne peuvent pas indiquer les différents transcrits alternatifs des gènes de la séquence à annoter de façon assez fiable. En effet contrairement aux alignements d'ADNc de la même espèce, ces ressources ne donnent pas les bornes précises des exons des transcrits. Ainsi nous considérons des modèles de transcrits mono-protéines d'étendue génomique chevauchante comme faisant partie de la même structure de transcrit. Rappelons que la majeure partie des protéines isssues de bases de données sont obtenues par traduction du CDS annoté d'un ADNc pleine longueur. Il suffit donc que deux ADNc pleine longueur aient un CDS identique pour qu'une même protéine soit présente en deux exemplaires dans une base de données. La redondance des alignements protéiques sur la séquence à annoter peut également s'expliquer par l'existence de protéines paralogues proches dans les bases de données (voir annexe B).

Illustration des relations. La figure 3.13 page 83 illustre les relations *extension* et *inclusion*, et la figure 3.14 page 83 la relation *chevauchement*.

Ajout de couleurs d'arêtes en DACMLang. Dans le langage DACMLang l'ajout de couleurs d'arêtes entre les sommets de $DACM_2$ est spécifiée par la sous-commande suivante :

```
| 5'[Prot] >>- 3'[Prot] -> chevauchement          (1)

| (5'[Adnc] >>- 3'[Adnc]) &&
  (5'[Adnc] >< 3'[Adnc])  &&
  ((3'[Adnc]) >=* (5'[Adnc])) -> extension        (2)

| (5'[Adnc] >>- 3'[Adnc]) &&
  (5'[Adnc] >< 3'[Adnc])  &&
  ((3'[Adnc]) <* (5'[Adnc])) -> inclusion         (3)
```

La règle (1) permet d'établir la couleur *chevauchement* entre deux sommets protéiques de $DACM_2$. Cette couleur est ajoutée entre deux sommets protéiques de $DACM_2$ dont les transmods sont ordonnés (`5'` et `3'`) et ont des étendues génomiques chevauchantes (`>>-`).

Les règles (2) et (3) permettent d'établir les couleurs *extension* et *inclusion* entre deux sommets ADNc de $DACM_2$. Le test qui précède la flèche (`->`) est la conjonction de trois conditions dont les deux premières sont identiques pour les deux règles. La première condition teste si les transmods associés à deux sommets ADNc de $DACM_2$ sont à la fois ordonnés et d'étendue génomique chevauchante, et la deuxième teste si ces transmods ont leurs introns chevauchants identiques. La troisième condition (qui distingue les deux règles) teste si la fin du transmod le plus en 3' est après (`>=*`, règle (2)) ou strictement avant (`<*`, règle (3)) la fin du transmod le plus en 5'.

L'ajout de couleurs d'arêtes entre les sommets de $DACM_2$ correspond à la phase experte de comparaison et de mise en relation des modèles de transcrits mono-molécules de même type. Elle est suivie de la réduction de $DACM_2$, qui correspond à la phase experte d'appariement des transcrits mono-molécules de même type.

Réduction en DACMLang. Comme nous voulons réduire $DACM_2$ différemment selon le type de ses sommets, la sous-commande de réduction de $DACM_2$ est constituée de deux expressions alternatives séparées par le signe `||`. Nous voulons réduire :

- le sous-graphe protéique de $DACM_2$ selon la couleur *chevauchement* et par les plus longs chemins d'arêtes (`+`),
- le sous-graphe ADNc de $DACM_2$ selon la couleur *extension* et par les plus longs chemins d'arêtes (`+`).

Ainsi la sous-commande de réduction de $DACM_2$ s'écrit en DACMLang :

```
[Prot]chevauchement+ || [Adnc]extension+
```

Le DACM obtenu par réduction de $DACM_2$ est appelé $DACM_3$.

A ce stade $DACM_3$ possède des sommets, mais ceux-ci ne sont reliés entre eux par aucune couleur d'arête. Les couleurs d'arêtes de $DACM_3$ correspondent aux relations que l'expert juge pertinent d'établir entre les modèles de transcrits multi-molécules de même type.

3.4.3 $DACM_3$

La troisième et dernière étape de l'annotation experte agit sur des modèles de transcrits multi-molécules de même type. Elle consiste en la comparaison et le regroupement de modèles de transcrit multi-molécules de même type redondants qui représentent la même structure de transcrit de la séquence à annoter. Plus précisément cette étape compare et relie un même modèle de transcrit multi-ADNc à plusieurs modèles de transcrits multi-protéines. Les objets biologiques résultant de ce regroupement sont appelés *modèles de transcrits multi-molécules multi-types* (*modèles de transcrits de niveau* 4). Ce sont les stuctures de transcrits alternatives les plus complètes et les moins redondantes de la séquence à annoter, et sont celles produites en sortie d'Exogean.

Justification. Comme celle qui la précède, cette étape permet d'éliminer la redondance et d'exploiter la complémentarité des ressources. Elle pemet donc d'annoter les transcrits d'une séquence de façon à la fois la plus concise et la plus complète possible. Cette étape est particulière car elle réalise la **fusion** des ressources ADNc et protéine. En effet, outre le fait que les modèles de transcrits multi-molécules d'un type de ressource donné (ADNc ou protéine) peuvent étendre les modèles de transcrits multi-molécules de l'autre type par l'intermédiaire d'un ou de plusieurs exons en 5' et/ou en 3', les ressources de type ADNc et protéine sont tout à fait complémentaires (voir tableau 3.2).

Source	Avantages
ADNc	. Bornes d'épissage . Transcrits alternatifs . Exons contigus
Protéine	. Exons codants . Phase des exons . Bornes du CDS

TAB. 3.2 – **Apport des différentes sources de données.**

Description des relations. Pour savoir si un modèle de transcrit multi-ADNc et un modèle de transcrit multi-protéine appartiennent à la même structure de transcrit, l'expert utilise deux relations portant sur un couple (modèle de transcrit multi-ADNc, modèle de transcrit multi-protéine) :

- le chevauchement de l'étendue génomique : *chevauchement*,
- le fait qu'aucun modèle d'exon du deuxième (le modèle de transcrit multi-protéine) ne soit totalement inclus dans un modèle d'intron du premier (le modèle de transcrit multi-ADNc) : *epissage_ compatible*.

Le tableau 3.2 explique pourquoi les arêtes colorées de $DACM_3$ relient un sommet ADNc à plusieurs sommets protéiques et non l'inverse. En effet à ce stade les sommets ADNc indiquent chacun un transcrit alternatif différent et ne peuvent donc plus se compléter l'un l'autre (ils représentent des classes exclusives). En revanche les sommets protéiques représentent des modèles de transcrits plus approximatifs et peuvent donc encore se compléter entre eux ou compléter des modèles de transcrits multi-ADNc. Ce tableau montre aussi que lorsqu'un ADNc indique des exons contigus sur la séquence à annoter, la confiance est grande dans les bornes d'épissage. C'est la raison pour laquelle la seule contrainte pour lier un modèle de transcrit multi-ADNc à un modèle de transcrit multi-protéine, est de s'assurer qu'aucun HSP du deuxième n'est totalement inscrit dans un intron du premier. En effet si l'on reliait deux tels objets on ajouterait un exon protéique entre deux exons contigus ADNc et cela violerait notre règle de confiance dans les bornes d'épissage d'un ADNc.

Illustration des relations utilisées. La figure 3.15 page 84 illustre la relation d'épissage compatible. La relation de chevauchement de l'étendue génomique à été illustrée précédemment.

Ajout de couleurs d'arêtes en DACMLang. Dans le langage DACMLang l'ajout de couleurs d'arêtes entre les sommets de $DACM_3$ est spécifiée par la sous-commande suivante :

```
| 5'[Adnc] >>- 3'[Prot] -> chevauchement                    (1)
| 3'[Adnc] >>- 5'[Prot] -> chevauchement                    (2)

| (5'[Prot]) >>= (3'[Adnc]) -> epissage_compatible          (3)
| (3'[Prot]) >>= (5'[Adnc]) -> epissage_compatible          (4)
```

Les règles `(1)` et `(2)` permettent d'établir la couleur *chevauchement* entre un sommet ADNc et un sommet protéique. Cette couleur est ajoutée entre un sommet ADNc et un sommet protéique dès lors que leurs transmods ont des étendues génomiques chevauchantes (`>>-`), et ceci quel que soit leur ordre (d'où les deux règles).

Les règles `(3)` et `(4)` permettent d'établir la couleur *epissage_ compatible* entre un sommet ADNc et un sommet protéique. Cette couleur est ajoutée entre un sommet ADNc et un sommet protéique dès lors que leurs transmods se chevauchent de façon interne (`>>=`), et

ceci quel que soit leur ordre (d'où les deux règles).

L'ajout de couleurs d'arêtes entre les sommets de $DACM_3$ correspond à la phase experte de comparaison et de mise en relation des modèles de transcrits multi-molécules. Elle est suivie de la réduction de $DACM_3$, qui correspond à la phase experte d'appariement des transcrits multi-molécules.

Réduction en DACMLang. Nous voulons réduire $DACM_3$ selon les couleurs *chevauchement* et *epissage_compatible* à la fois (`&&`). D'autre part nous ne voulons pas perdre les sommets ADNc qui ne sont liés à aucun sommet protéique. Ainsi la sous-commande de réduction de $DACM_3$ s'écrit en DACMLang :

```
(chevauchement && epissage_compatible) || [Adnc]
```

Les trois étapes de l'annotation experte par DACM, à savoir la réduction des $DACM_1$, $DACM_2$ et $DACM_3$, sont illustrées sur un même exemple biologique en figure 3.16 page 85.

3.5 Annotation experte annexe

L'annotation experte annexe désigne les étapes d'Exogean qui utilisent les règles heuristiques de l'expert, mais n'ont pas (encore !) été modélisées par des DACMs (figure 6.1 page 126). L'annotation experte annexe désigne donc les étapes suivantes :

- la recherche du CDS (étape 3 d'Exogean, sous-section 3.5.1),
- le pré-traitement des HSPs (étape 1 d'Exogean) et le filtre transcrit/pseudo-transcrit (étape 4 d'Exogean), regroupés sous le terme *filtrage des données* (sous-section 3.5.2).

Du fait de l'orientation de la transcription de 5' en 3' de la molécule d'ADN, et comme dans le cas de l'annotation experte par DACM, les étapes de l'annotation experte annexe analysent les données de 5' en 3' de la séquence d'ADN à annoter.

3.5.1 Recherche du CDS

La *recherche du CDS* suit l'annotation experte par DACMs (figure 6.1 page 126) et agit donc sur des modèles de transcrits multi-molécules multi-types. Pour un tel modèle, elle consiste en la définition de la partie codante de l'ARNm du transcrit correspondant. Comme un modèle de transcrit multi-molécule multi-type est un ensemble de modèles de transcrits multi-molécule de même type, qui sont eux-mêmes des ensembles de modèles de transcrits mono-molécules, qui sont eux-mêmes des ensembles d'HSPs, un modèle de transcrit multi-molécule multi-type est une collection d'HSPs d'ADNc et de protéines. Ainsi la recherche du CDS s'effectue sur une collection d'HSPs supposés appartenir à la même structure de transcrit.· Déterminer le CDS nécessite donc dans un premier temps de déterminer précisément la structure du transcrit, autrement dit de regrouper ces HSPs en exons.

Déterminer les exons du transcrit. Pour déterminer les exons d'un modèle de transcrit multi-molécule multi-type, nous déterminons dans un premier temps ses modèles d'exons sur la base du chevauchement des HSPs d'ADNc et de protéines qui le constituent. Ces modèles d'exons sont appelés *clusp* (cluster d'HSPs) et sont définis comme des ensembles maximaux d'HSPs chevauchants (figure 3.17 page 86). Un clusp peut ainsi être vu comme un mot génomique dont la borne 5' correspond à la borne 5' de(s) HSP(s) les plus en 5' du clusp, et dont la borne 3' correspond à la borne 3' de(s) HSP(s) les plus en 3' du clusp. Cependant les bornes 5' et 3' d'un clusp n'ont a priori pas de sens biologique, contrairement à celles des exons.

Ce qui différencie un exon d'un clusp est le fait qu'il comporte des signaux interprétés par la machinerie cellulaire (signaux d'epissage ou de début/fin de transcription) au niveau de ses extrémités 5' et 3'. Ainsi pour déterminer l'exon qui correspond à un clusp nous procédons à une étape de recherche de signaux en 5' et en 3' du clusp.

Les signaux (5' et 3') de l'exon correspondant à un clusp peuvent se situer soit au niveau des bornes de l'un des HSPs du clusp, soit au voisinage des bornes du clusp. Les signaux recherchés sont de deux types :

- les signaux *externes* du transcrit : ils constituent les bornes 5' et 3' du transcrit,
- les signaux *internes* du transcrit : ils constituent les signaux d'épissage du transcrit.

La recherche du signal externe 5' du transcrit dépend du type (ADNc ou protéine) des molécules des HSPs les plus en 5' du clusp initial :

- Si ces HSPs proviennent d'un ADNc alors la borne externe 5' du transcrit est donnée par leur borne 5',
- Si ces HSPs proviennent d'une protéine alors un codon ATG en phase avec l'extrémité 5' de ces HSPs est recherché dans la direction 5' du clusp.

La recherche du signal externe 3' du transcrit se fait exactement de la même façon, mais en remplaçant 5' par 3', initial par terminal, et ATG par Stop.

La recherche des signaux internes du transcrit équivaut à celle des bornes de ses introns et agit donc sur une paire de clusps contigus du transcrit. A l'image de la recherche des signaux externes du transcrit, la recherche des signaux internes dépend du type de molécules, ADNc ou protéines, des HSPs de chaque paire de clusps (c_1, c_2) inspectée :

- si c_1 et c_2 contiennent des HSPs h_1 et h_2 provenant d'un même ADNc, alors la borne 3' de h_1 et la borne 5' de h_2 sont prises comme bornes de l'intron formé par c_1 et c_2,
- sinon
  - si c_1 et c_2 contiennent des HSPs h_1 et h_2 provenant d'une protéine, alors deux étapes ont lieu :
    1. constitution d'un ensemble de couples de signaux (donneur,accepteur) : on recherche des signaux donneurs d'épissage à partir de la borne 3' de h_1 et dans la direction 3', et des signaux accepteurs d'épissage à partir de la borne 5' de h_2 et dans la direction 5', sur une certaine distance génomique. Si h_1 et h_2 proviennent de la **même** protéine P alors cette distance est fonction du nombre d'acides aminés manquants entre h_1 et h_2 dans P, sinon elle est donnée par défaut. Cette

recherche produit un ensemble de couples de signaux (donneur,accepteur) potentiels ;

2. choix de la meilleure paire de signaux (donneur,accepteur) : parmi tous les couples de signaux (donneur,accepteur) ainsi constitués seuls ceux qui préservent la phase indiquée par les protéines[6] sont analysés. Préserver la phase signifie ici à la fois induire un ajout (ou un retrait) de nucleotides multiple de 3, et ne pas introduire de codon stop en phase. Pour sélectionner le meilleur couple (donneur,accepteur), la stratégie est différente selon que h_1 et h_2 proviennent ou non d'une même protéine P. Si tel est le cas, alors on sélectionne le couple (donneur,accepteur) qui permet de compléter le nombre d'acides aminés manquants entre h_1 et h_2 dans P, sinon on sélectionne celui qui maximise la somme (score donneur par matrice de poids de signal donneur + score accepteur par matrice de poids de signal accepteur).

– sinon : on recherche des signaux donneurs d'épissage à partir de la borne 3' de h_1 et dans la direction 3', et des signaux accepteurs d'épissage à partir de la borne 5' de h_2 et dans la direction 5', sur une distance génomique donnée par défaut. Parmi tous les couples de signaux (donneur,accepteur) ainsi constitués, on retient celui qui maximise la somme (score donneur par matrice de poids de signal donneur + score accepteur par matrice de poids de signal accepteur).

Avoir identifié les signaux externes et internes du transcrit signifie avoir reconstitué précisément la structure du transcrit. Comme le CDS est la partie traduite de protéine de l'ARNm du transcrit, il est nécessaire de déterminer l'ARNm du transcrit. Comme les modèles de transcrits multi-molécules multi-types dont nous disposons ne représentent pas nécessairement des structures de transcrits *complètes*, la concaténation de leurs exons ne constitue pas toujours l'ARNm complet du transcrit, mais une séquence de nucléotides que nous nommons *miniseq* ([54], figure 3.18 page 86). Il s'agit alors de trouver un CDS dans cette miniseq.

Trouver un CDS dans la miniseq. Le CDS d'un transcrit est habituellement défini comme une sous-séquence de l'ARNm du transcrit, de longueur multiple de 3, ne contenant pas de codon stop en phase, débutant par un codon ATG (acide aminé Méthionine) et suivie par un codon stop. Or ce que nous appelons transcrit est en fait ici une collection d'HSPs (voir annotation experte par DACM section 3.4), qui révèlent la présence d'une structure de transcrit en cette position de la séquence, mais pas nécessairement dans sa totalité. Si ces HSPs représentent une structure de transcrit incomplète, alors la miniseq représente un ARNm partiel et le CDS recherché dans cette miniseq peut être incomplet. Pour cette raison nous devons étendre la définition classique du CDS.

Un CDS est ici défini comme une sous-séquence de la miniseq de longueur multiple de 3 et ne contenant pas de codon stop en phase, pouvant débuter soit par un codon ATG (Met) soit au début (Deb) de la miniseq, et pouvant se terminer soit par un codon Stop (Stop) soit à la fin (Fin) de la miniseq. Nous définissons donc quatre types de CDS :

[6] Cette phase est supposée être la même pour les différentes protéines de c_1 et c_2.

- MetStop : CDS complet en 5' et en 3',
- MetFin : CDS complet en 5' et incomplet en 3',
- DebStop : CDS incomplet en 5' et complet en 3',
- DebFin : CDS incomplet en 5' et en 3'.

La recherche du CDS dans la miniseq se fait différemment selon si le transcrit contient ou non des HSPs de protéines. En effet un alignement de protéine sur une séquence d'ADN génomique permet à la fois de localiser un exon codant mais aussi de connaître sa phase. Si un transcrit contient des HSPs de protéines, ces derniers indiquent chacun une phase. Si cette phase est la même pour tous, alors on considère que cette phase est celle du CDS recherché, sinon on ne tient pas compte de la phase des HSPs de protéines.

Nous obtenons ainsi la procédure suivante :

- Constitution de l'ensemble des CDS :
  - S'il existe un HSP de protéine et qu'il indique la même phase p que tous les autres HSPs de protéines, alors on recherche tous les CDS (tels que définis plus haut) en phase p,
  - Sinon on recherche tous les CDS en phases 1, 2 et 3 ;
- Sélection du CDS le plus probable : parmi l'ensemble des CDS précédemment trouvés, on sélectionne le plus long[7].

Chaque transcrit obtenu par DACM est ainsi associé à un CDS et donc à une protéine, éventuellement partielle.

En raison d'artéfacts potentiels dans les données à différentes étapes du protocole d'annotation, des étapes de *filtrage des données* sont nécessaires, en particulier avant l'annotation par DACM (prétraitement des HSPs) et après la recherche du CDS (filtre transcrit/pseudo-transcrit).

3.5.2 Filtrage des données

Avant l'annotation experte par DACMs et après la recherche du CDS, ont lieu deux étapes de filtrage des données : le pré-traitement des HSPs et le filtre transcrit/pseudo-transcrit.

Pré-traitement des HSPs. Le pré-traitement des HSPs est la première étape d'Exogean (figure 6.1 page 126), qui prend en entrée des HSPs d'ADNc de la même espèce et de protéines d'une autre espèce, et élimine de cet ensemble les artéfacts d'alignement, ou plus généralement les données impropres à la construction de modèles de transcrits codants·

Lorsqu'une séquence σ d'ADN génomique est comparée à une séquence λ d'ADNc de la même espèce ou de protéine d'une autre espèce, les séquences se situant aux extrémités des exons sont souvent conservées différemment de la partie centrale de ceux-ci (plus ou moins que cette dernière selon les cas). Ainsi les programmes d'alignement produisent souvent des HSPs dont les bornes ne représentent pas les bornes *exactes* des exons. Le pré-traitement s'assure que les HSPs provenant de la même molécule λ, d'ADNc ou de protéine, peuvent

[7] Cette sélection pourrait se faire selon d'autres critères que la longueur, par exemple selon le nombre d'exons, ou même selon une combinaison de différents critères (voir perspectives en section 7.2).

servir de «briques de base» à la construction d'une structure de transcrit sur σ. Pour cela le pré-traitement doit parfois *rogner*, *fusionner* ou *éliminer* certains HSPs. Cette procédure reflète les décisions prises par l'humain lorqu'il évalue l'ensemble des alignements d'ADNc et de protéines sur une région génomique donnée.

Le pré-traitement utilise des règles différentes selon que les HSPs proviennent d'ADNc de la même espèce, ou de protéines d'une autre espèce.

Pré-traitement des HSPs d'ADNc. Le pré-traitement des HSPs d'ADNc agit sur des paires d'HSPs contiguës et ordonnées sur σ, provenant du même ADNc. Soit h_1 et h_2 deux tels HSPs, k un entier positif et $\delta_{n_{1,2}}$ le nombre de nucléotides séparant h_1 et h_2 sur σ (distance sur σ entre la fin de h_1 et le début de h_2). Le pré-traitement agit sur des HSPs h_1, h_2 vérifiant l'une des conditions suivantes :

- la distance génomique entre h_1 et h_2 est très petite ($\delta_{n_{1,2}} \leq k$) : h_1 et h_2 sont fusionnés en un seul et même HSP d'ADNc car cela indique que l'une des séquences (celle à annoter ou celle de l'ADNc) est de mauvaise qualité (en général $k = 3$ *nucleotides*),
- les bornes de l'intron entre h_1 et h_2 ne sont pas correctes (généralement GT/AG mais des signaux non canoniques sont également tolérés) : h_1 et h_2 sont éliminés car des alignements d'ADNc de la même espèce doivent donner des bornes d'épissage correctes. Le cas contraire indique soit un mauvais étiquetage de l'ADNc soit une contamination génomique (surtout si l'ADNc est de type EST).

Pré-traitement des HSPs de protéines. Le pré-traitement des HSPs de protéines consiste en deux étapes :

- une étape de *rognage* des HSPs, qui analyse les HSPs une à une, et compare leur mot génomique et protéique afin de rogner les extrémités 5' et 3' *mal alignées* de ces HSPs,
- une étape de *fusion/élimination* des HSPs, qui analyse les HSPs par paires (h_1, h_2) contiguës ordonnées sur σ de même protéine ν, et qui se fonde sur la position relative de h_1 et h_2 sur σ et ν, afin de repérer des problèmes d'alignement et de les rectifier, soit en fusionnant h_1 avec h_2, soit en éliminant h_2.

Rognage des HSPs de protéines. Les HSPs de protéines sont obtenus par un programme d'alignement local heuristique et représentent donc des exons codants approximatifs. Le rognage des HSPs de protéines vise à éliminer les extrémités des HSPs de protéines qui s'alignent mal, et qui sont donc potentiellement situées dans les introns d'un transcrit. Un ensemble d'HSP étant une relation entre deux séquences, tout HSP h peut se modéliser comme un couple de mots (ω, ρ) dont l'un est sur l'alphabet de la première séquence et l'autre sur l'alphabet de la deuxième séquence. Soit $h = (\omega, \rho)$, $\omega \in \Sigma_{nt}^*$, $\rho \in \Sigma_{aa}^*$ un HSP de protéine, le *rognage* de h consiste en trois étapes (figure 3.19 page 87) :

- la traduction de ω en acides aminés,
- la comparaison en terme de matchs (appariements) et de mismatches (misappariements) d'acides aminés, du mot protéique ainsi formé au mot protéique ρ. Cela produit un mot sur l'alphabet $\{\|, \times\}$ des matchs et des mismatches de taille $\mid \rho \mid$,

– le parcours de ce mot, d'abord de 5' en 3' puis de 3' en 5', par fenêtre glissante de taille fixe W (ici 5), dans le but de repérer les préfixes et les suffixes maximaux qui correspondent à des extrémités 5' et 3' de h mal alignées et donc à éliminer de h (d'où le terme de *rognage*). Pour cela il faut définir les propriétés d'une extrémité bien alignée, autrement dit définir les propriétés d'un mot de taille W sur l'alphabet $\{\|, \times\}$ correspondant à une extrémité bien alignée. Ici ces propriétés sont les suivantes (mais elles peuvent être modifiées) :
  – débuter par un match $\|$,
  – contenir au maximum deux mismatches $\times$.

Fusion et élimination des HSPs de protéines. L'étape de *fusion et élimination des HSPs de protéines* agit sur une paire d'HSPs (h_1, h_2) contiguës ordonnées sur σ de même protéine ν, et inspecte leur position relative sur σ et ν à l'aide de deux variables :
– $\delta_{n_{1,2}}$: nombre de nucléotides séparant h_1 et h_2 sur σ,
– $\delta_{a_{1,2}}$: nombre d'acides aminés séparant h_1 et h_2 sur ν.
L'idée sous-jacente à l'étape de *fusion/élimination* des HSPs est qu'un ensemble d'HSPs de même protéine ν appartenant à une autre espèce que celle de σ, et se situant sur σ à proximité les uns des autres, ont de fortes chances de représenter un transcrit de σ dont la traduction du CDS forme tout ou partie d'une protéine de σ orthologue à ν (voir annexe B). Si tel est le cas, les HSPs devraient être séparés sur σ d'une distance assez grande ($\delta_{n_{1,2}} \gg 1$), correspondant à la taille d'un intron, mais séparées sur ν d'une très courte distance ($\delta_{a_{1,2}} \simeq 0$) (en effet la protéine ν ne comporte pas d'introns). Ainsi un couple $(\delta_{n_{1,2}}, \delta_{a_{1,2}})$ vérifiant $\delta_{n_{1,2}} \simeq 3 \times \delta_{a_{1,2}}$ peut signifier, si $\delta_{n_{1,2}}$ est suffisamment petit, que h_1 et h_2 font en réalité partie d'un même exon codant, dont la partie centrale est peu conservée entre les espèces de σ et de ν et ne se serait pas alignée. Plus précisément, si k est un paramètre entier, neuf conditions sur $\delta_{n_{1,2}}$ et $\delta_{a_{1,2}}$ (tableau 3.3) conduisent à la fusion de h_1 et h_2 ou à l'élimination de h_2..

Pour plus de détails sur le pré-traitement des HSPs de protéines ainsi qu'une formalisation de cette étape, voir [60].

Après le pré-traitement des HSPs a lieu l'annotation experte par DACMs, suivie de la recherche du CDS des transcrits ainsi obtenus. Parmi les différents transcrits alternatifs des gènes de σ dotés de leur CDS, certains ne codent pas pour des protéines fonctionnelles et peuvent représenter des pseudogènes : le *filtre transcrit/pseudo-transcrit* permet de faire cette discrimination.

Filtre transcrit/pseudo-transcrit. Le *filtre transcrit/pseudo-transcrit* est la quatrième et dernière étape d'Exogean (figure 6.1 page 126), qui sert à classifier les modèles de transcrits obtenus par annotation experte et dotés d'un CDS dans l'une des deux classes suivantes :
– *modèle de transcrit final*, codant pour une protéine fonctionnelle,
– *modèle de pseudo-transcrit*, ne codant pas pour une protéine fonctionnelle.
Rappelons la structure idéale d'un transcrit eucaryote (cf section 3.1) :
– une grande étendue génomique,
– des exons courts,

- des introns longs,
- un grand nombre d'exons,
- un CDS long,
- un grand nombre d'exons codants, dans l'absolu et relativement au nombre d'exons UTR ou non codants,
- un CDS débutant par un codon ATG et se terminant par un codon Stop.

Nous recherchons donc des transcrits de ce type et définissons pour cela les cinq paramètres suivants (entiers) :

- *sizecdsnomet* : taille minimale d'un CDS ne débutant pas par une méthionine (en nucléotides),
- *sizecdswithmet* : taille minimale d'un CDS débutant par une méthionine (en nucléotides),
- *propcdsmonoex* : pourcentage minimal entre la longueur du CDS et la longueur du transcrit pour un CDS monoexonique,
- *sizecdsmonoex* : taille minimale d'un CDS monoexonique (en nucléotides),
- *sizecdsdiex* : taille minimale d'un CDS diexonique (en nucléotides).

Ainsi pour un modèle de transcrit t de cds c, de longueur totale d'exons en nucléotides $lgextot$, de nombre d'exons codants $nbexincds$ et de longueur totale d'exons codants en nucléotides $lgexcds$, t est rangé dans la catégorie *modèle de pseudo-transcrit* si l'une des conditions suivantes est vérifiée :

- c ne commence pas par un ATG et $lgexcds < sizecdsnomet$,
- c commence par un ATG et $lgexcds < sizecdswithmet$,
- $nbexincds = 1$ et $lgexcds < \frac{propcdsmonoex}{100} \times lgextot$,
- $nbexincds = 1$ et $lgexcds < sizecdsmonoex$,
- $nbexincds = 2$ et $lgexcds < sizecdsdiex$,
- t n'est constitué que par des HSPs de protéines et en ce cas soit $lgexcds$ n'est pas multiple de 3, soit c comporte un stop en phase.

Dans tous les autres cas, t est classé dans la catégorie *modèle de transcrit final*.

Le présent chapitre décrit l'annotation de gènes réalisée par les experts humains, et la manière dont nous avons choisi de l'automatiser. Pour combiner les heuristiques utilisées par les experts humains nous utilisons en effet plusieurs constructions et réductions de multigraphes colorés : c'est le programme Exogean. Etant donné le caractère récurrent de cette double action, nous avons choisi de développer un cadre formel pour son expression : le langage DACMlang.

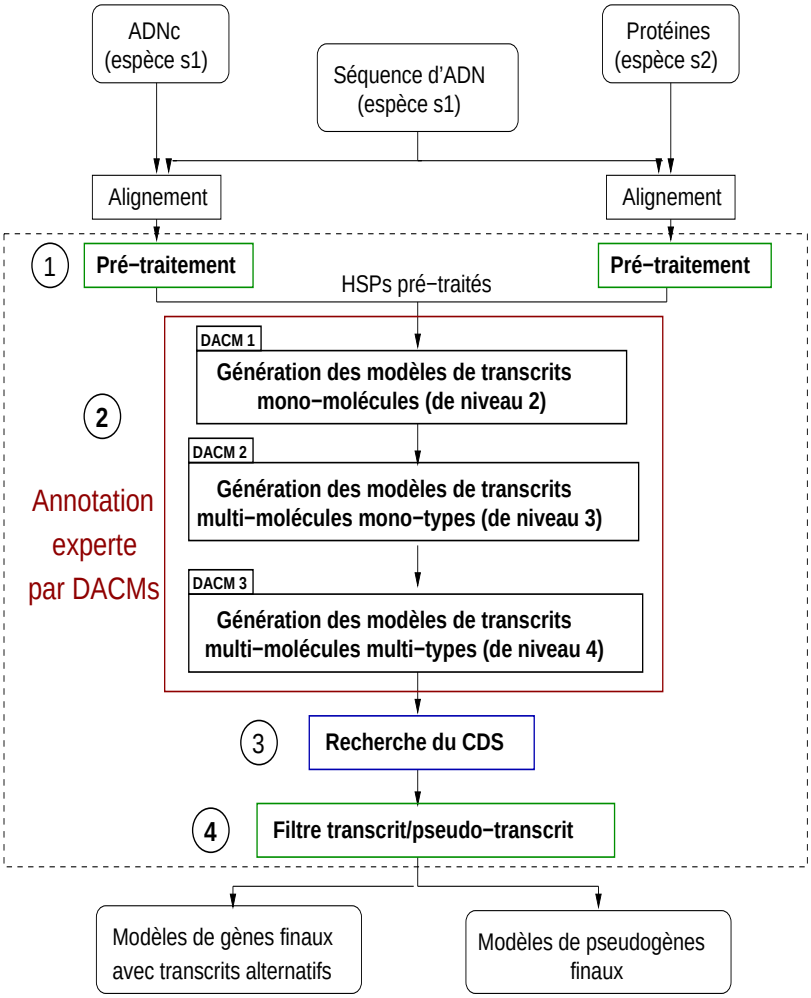
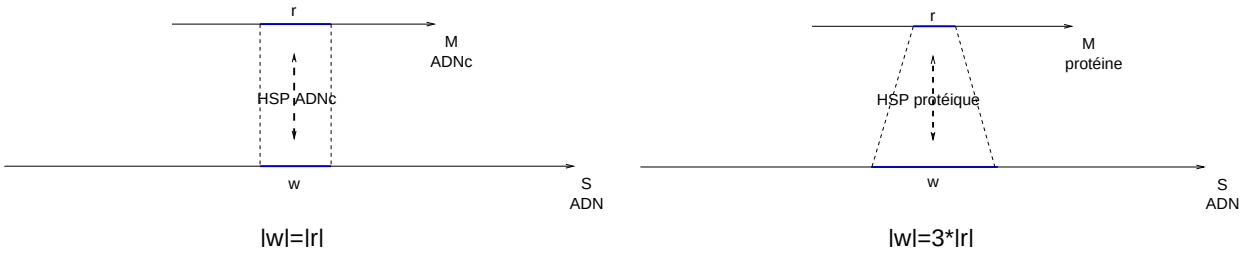


FIG. 3.9 – **Les différentes étapes d'Exogean.** Exogean prend en entrée la séquence à annoter ainsi que le résultat (HSPs) de l'alignement entre cette séquence et d'une part des séquence d'ADNc de la même espèce, et d'autre part des séquences de protéines d'une autre espèce. Exogean se compose ensuite de quatre étapes : (1) le **pré-traitement** des HSPs d'ADNc et de protéines ; (2) l'**annotation experte par DACM** qui produit les structures des transcrits alternatifs des gènes de la séquence à annoter à partir des HSPs pré-traités. Cette étape se compose de trois sous-étapes dont chacune produit des modèles de transcrits de «plus grande complexité» par réduction d'un DACM ; (3) la **recherche du CDS** (CoDing Sequence) qui définit les bornes de la partie de chaque transcrit qui codera les acides aminés de la protéine correspondante ; (4) le **filtre transcrit/pseudo-transcrit** qui élimine les pseudogènes de ces structures.



(a) HSP d'ADNc (b) HSP de protéine

FIG. 3.10 – **Un HSP d'ADNc (a) et un HSP de protéine (b).** Un HSP peut se voir comme une relation entre la séquence à annoter et la séquence d'une molécule codante épissée. Ainsi, un HSP peut se représenter comme un couple (mot génomique, mot de molécule codante épissée). Comme les alignements locaux utilisés ici sont sans indels, le mot génomique d'un l'HSP d'ADNc est de même taille que son mot d'ADNc, et le mot génomique d'un HSP de protéine est de taille trois fois supérieure à celle de son mot de protéine.

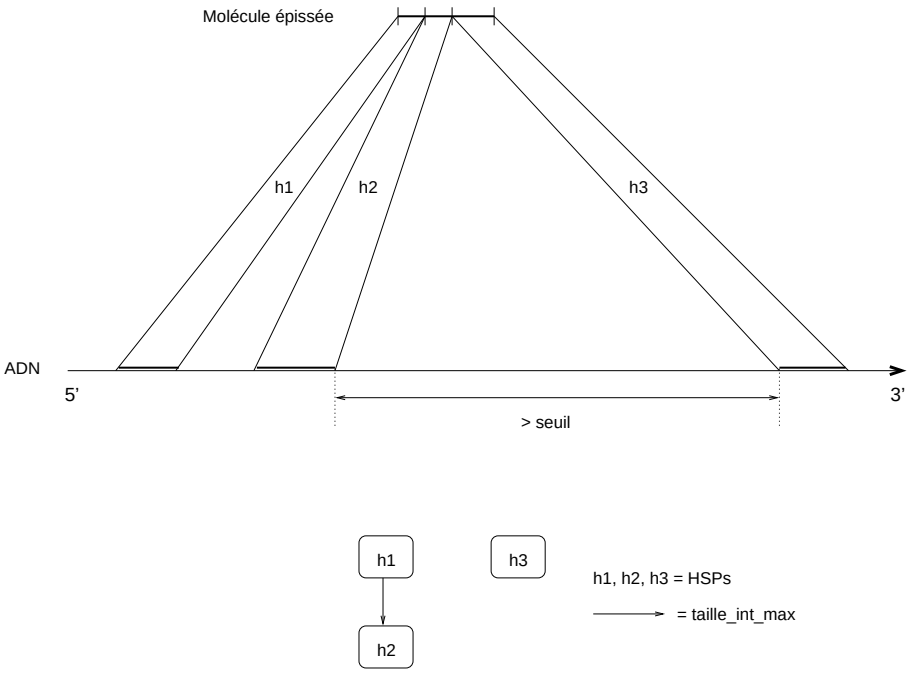


FIG. 3.11 – **Relation de la taille maximale d'intron.** Cette figure représente trois HSPs h_1, h_2 et h_3, d'une même molécule épissée (ADNc ou protéine). Sous ces HSPs est représenté le DACM de la relation $taille\_int\_max$ correspondant : ses sommets sont les différents HSPs. Quant à l'ajout de la couleur d'arête $taille\_int\_max$, elle s'effectue par comparaison des HSPs ordonnés : la distance génomique entre h_1 et h_2 est inférieure à la taille maximale d'intron la couleur $taille\_int\_max$ est donc ajoutée entre les sommets correspondants, en revanche la distance génomique entre h_2 et h_3 n'est pas inférieure à la taille maximale d'intron, la couleur $taille\_int\_max$ n'est donc pas ajoutée entre les sommets correspondants.

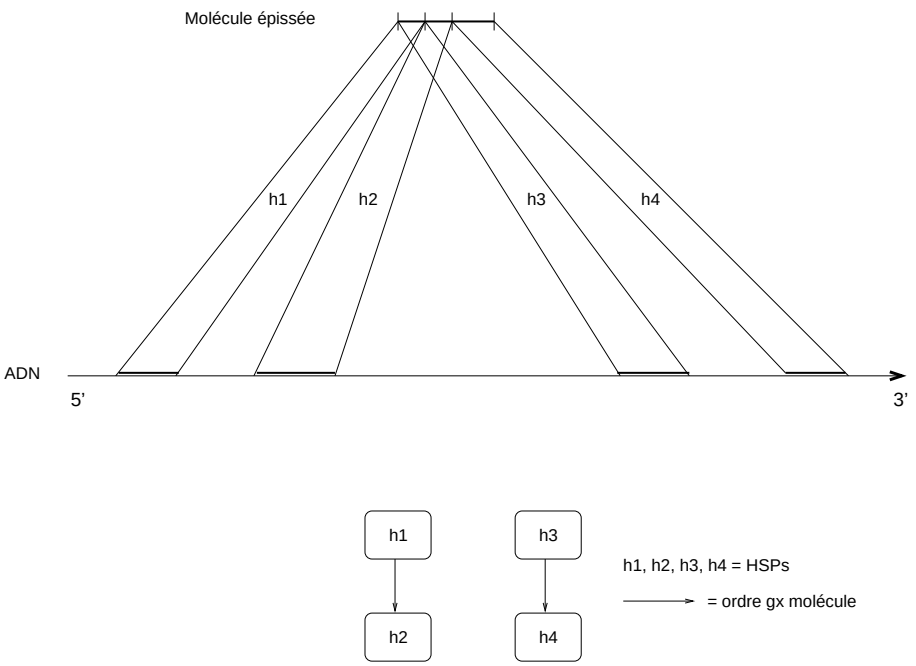


FIG. 3.12 – **Relation de l'ordre génomique molécule.** Cette figure représente quatre HSPs h_1, h_2, h_3 et h_4, d'une même molécule épissée (ADNc ou protéine). Sous ces HSPs est représenté le DACM de la relation *ordre_gx_molécule* correspondant : ses sommets sont les différents HSPs. Quant à l'ajout de la couleur d'arête *ordre_gx_molécule*, elle s'effectue par comparaison des HSPs ordonnés : h_1 et h_2 d'une part et h_3 et h_4 d'autre part sont ordonnés sur la molécule épissée, la couleur *ordre_gx_molécule* est donc ajoutée entre les sommets correspondants, en revanche h_2 et h_3 ne sont pas ordonnés sur la molécule épissée, la couleur *ordre_gx_molécule* n'est donc pas ajoutée entre les sommets correspondants.

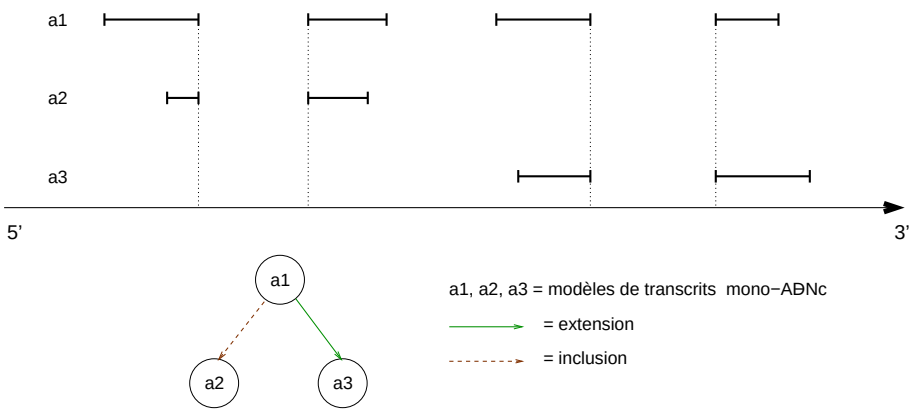


FIG. 3.13 – **Relations d'extension et d'inclusion.** Sur cette figure sont représentés trois modèles de transcrits mono-ADNc a_1, a_2 et a_3 d'une séquence d'ADN qui se chevauchent sur celle-ci. Au dessous de ces modèles est indiqué le DACM des relations *extension* et *inclusion* correspondant. Chaque modèle de transcrit donne lieu à un sommet différent du DACM, quant aux couleurs d'arêtes *extension* et *inclusion*, elles s'obtiennent par comparaison des modèles de transcrits chevauchants ordonnés, c'est-à-dire a_1 et a_2 et a_1 et a_3. Les introns chevauchants de a_1 et a_2 d'une part et de a_1 et a_3 d'autre part sont identiques. Comme la fin de a_2 est en 5' de celle de a_1 et que la fin de a_3 est en 3' de celle de a_1, la couleur *inclusion* est ajoutée entre les sommets a_1 et a_2, et la couleur *extension* est ajoutée entre les sommets a_1 et a_3.

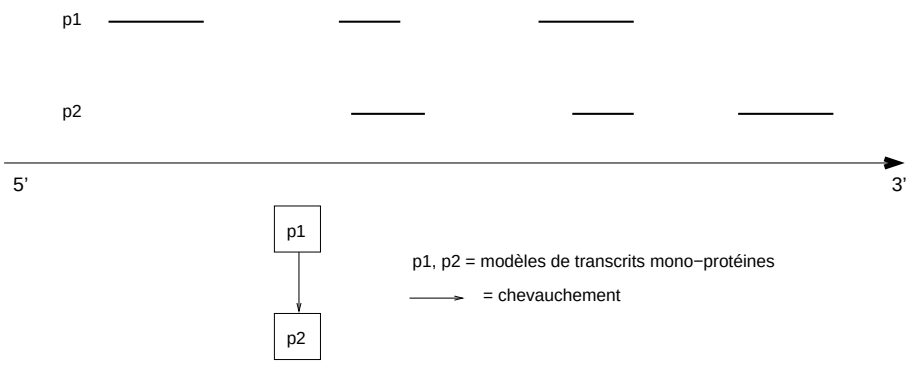


FIG. 3.14 – **Relation de chevauchement.** Sur cette figure sont représentés deux modèles de transcrits mono-protéines, p_1 et p_2, d'une séquence d'ADN, chevauchants sur celle-ci. Au dessous de ces modèles est indiqué le DACM de la relation *chevauchement* correspondant. Chaque modèle de transcrit donne lieu à un sommet différent du DACM et les couleurs entre ces sommets s'obtiennent par comparaison des modèles de transcrits ordonnés correspondants. Comme p_1 et p_2 sont ordonnés et chevauchants, la couleur *chevauchement* est ajoutée entre les sommets qui les représentent.

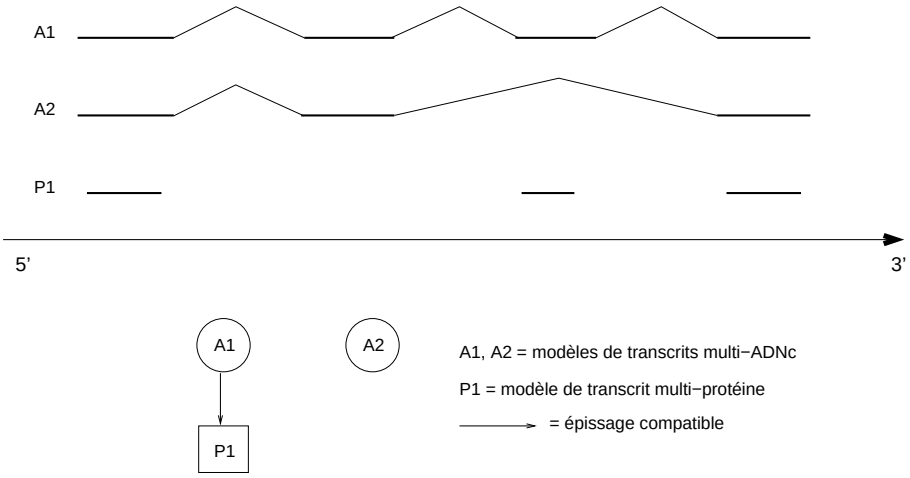


FIG. 3.15 – **Relation de l'épissage compatible.** Cette figure représente trois modèles de transcrits multi-molécules de même type d'une séquence d'ADN, chevauchants sur celle-ci. Les deux premiers, A_1 et A_2, sont de type ADNc et sont représentés avec leur modèles d'introns (sous forme de traits cassés reliant deux modèles d'exons), le troisième, P_1, est de type protéine. Aucun modèle d'exon de P_1 n'est totalement inclus dans un modèle d'intron de A_1, en revanche un modèle d'exon de P_1 est totalement inclus dans un modèle d'intron de A_2. Pour cette raison dans le DACM dont les sommets sont les modèles de transcrits multi-molécules de même type A_1, A_2 et P_1, la couleur *epissage_ compatible* est ajoutée entre les sommets A_1 et P_1 mais pas entre les sommets A_2 et P_1.

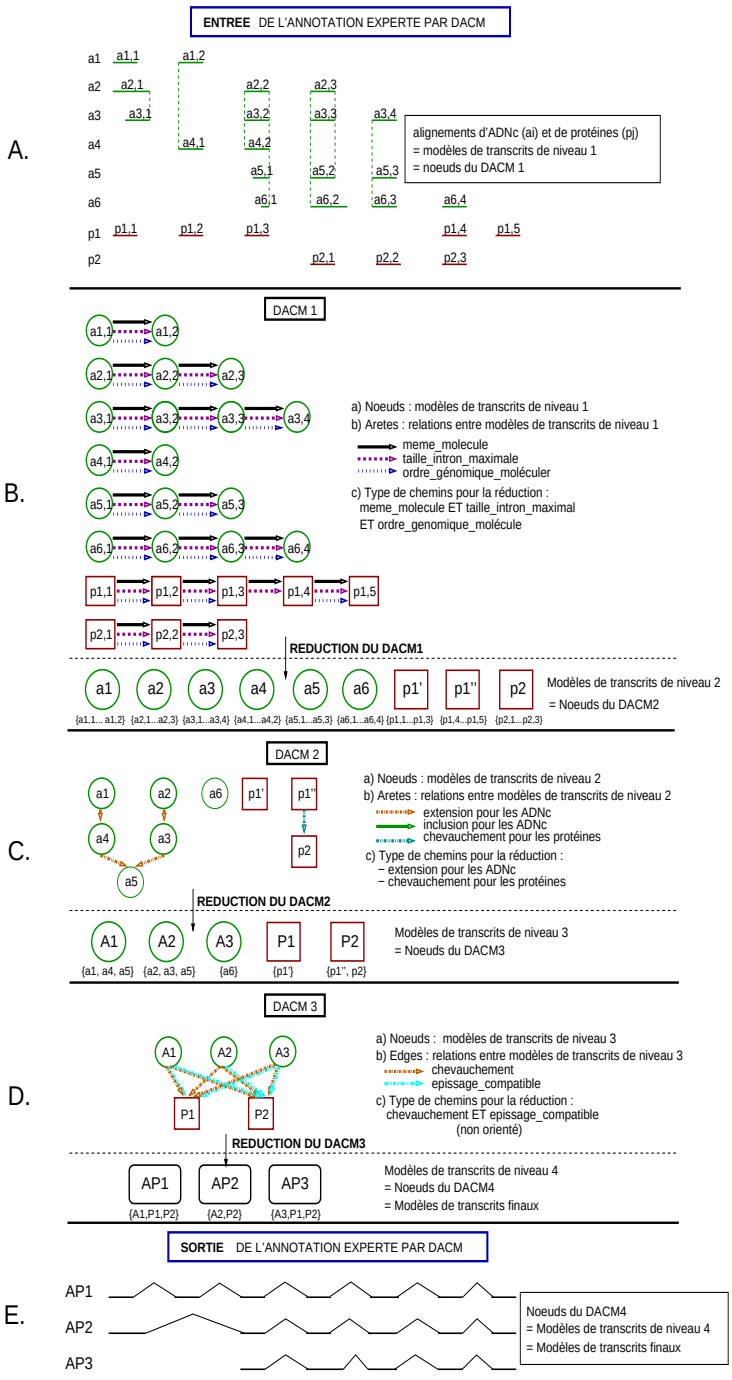


FIG. 3.16 – **Annotation experte par DACMs sur un exemple.** Ce schéma se compose de 5 parties : A. représente une entrée de l'annotation experte par DACM, à savoir un ensemble d'HSPs d'ADNc et de protéines sur la séquence d'ADN à annoter ; B., C. et D. représentent la construction et la réduction de $DACM_1$, $DACM_2$ et $DACM_3$ respectivement sur cet exemple ; E. représente la sortie de l'annotation experte par DACM, à savoir des modèles de transcrits multi-molécules multi-types sur la séquence à annoter.

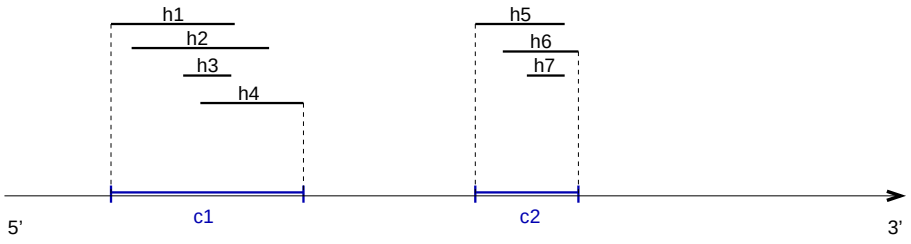


FIG. 3.17 – **Clusps d'un ensemble d'HSPs.** Cette figure représente sept HSPs h_1 à h_7 d'une séquence d'ADN, matérialisés par des traits pleins horizontaux au-dessus de celle-ci. On souhaite regrouper ces HSPs en ensembles maximaux d'HSPs chevauchants sur cette séquence, aussi appelés *clusps*. Ici on obtient donc deux clusps, c_1 et c_2, matérialisés par des traits bleus horizontaux. Les extrémités 5' et 3' de chaque clusp sont les bornes 5' et 3' des HSPs les plus en 5' et 3' du clusp. Les traits pointillés verticaux positionnent ces extrémités sur la séquence d'ADN.

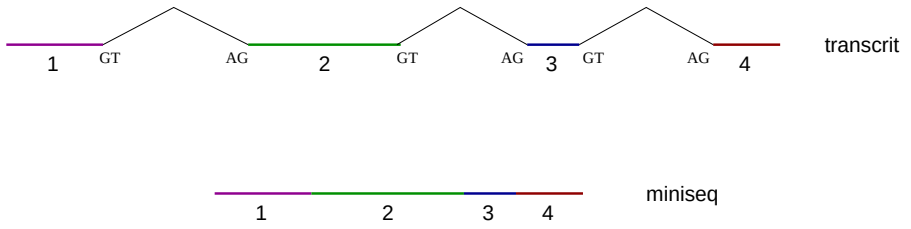


FIG. 3.18 – **Miniseq d'un transcrit.** Cette figure représente un modèle de transcrit multi-molécule multi-type dont les bornes internes ont été trouvées (bornes d'épissage, le plus souvent GT/AG). La miniseq du transcrit s'obtient par simple concaténation des exons successifs du transcrit. Cette miniseq représente tout ou partie de l'ARNm du transcrit, et contient donc tout ou partie du CDS du transcrit.

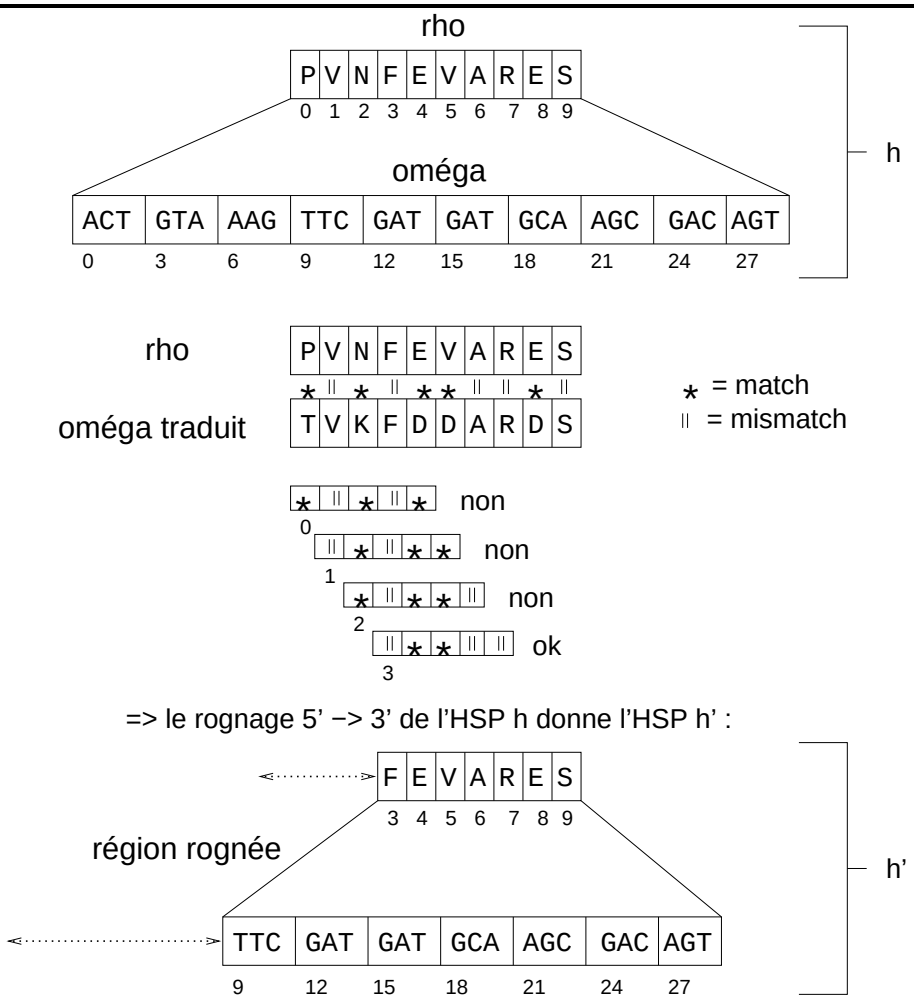
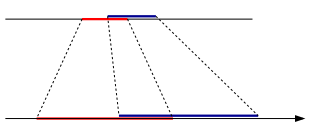
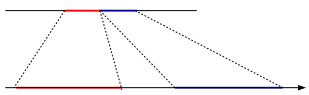
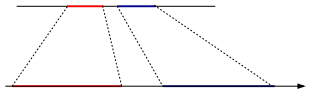
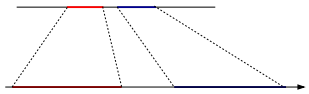
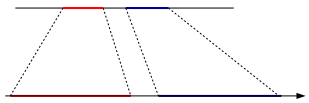
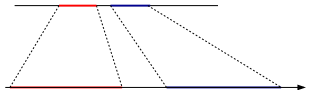
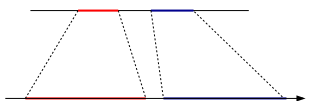
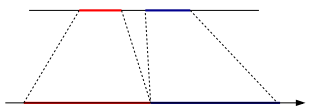
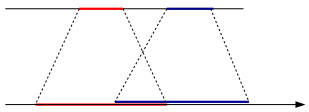


FIG. 3.19 – **Rognage 5'-> 3' d'un HSP.** Cette figure représente un HSP de protéine h de mot génomique oméga et de mot protéique rho. Le mot génomique oméga est traduit en acides aminés et le mot protéique ainsi obtenu est comparé au mot protéique rho de h en terme de match et de mismatch d'acide aminé. Le mot des matchs ($\|$) et des mismatches ($\times$) ainsi formé est parcouru de 5' en 3' par fenêtres de taille 5. Pour chaque sous-mot de taille 5, on teste s'il représente une extrémité 5' bien alignée de l'HSP (condition : débuter par un match et contenir au maximum deux mismatches). Les trois premières fenêtres ne représentent pas des extrémités 5' bien alignées de l'HSP, en revanche la quatrième en représente une. Pour cette raison le rognage 5'->3' de h produit l'HSP h' qui correspond à l'HSP h privé de ses trois premiers acides aminés (et donc de ses neuf premiers nucléotides).

Condition	Représentation	Action	Raison
$\begin{cases} \delta_{a_{1,2}} \leq -2 \\ \delta_{n_{1,2}} \leq -1 \end{cases}$		Fusion de h_1 et h_2	Répétitions dans les deux protéines (la protéine annotée sur σ, et la protéine orthologue ν).
$\begin{cases} \delta_{a_{1,2}} = -1 \\ 1 \leq \delta_{n_{1,2}} \leq 11 \end{cases}$		Fusion de h_1 et h_2	Insertion d'acides aminés dans la protéine annotée sur σ et frameshift si $\neg(\delta_{n_{1,2}} \equiv 0 \mod 3)$.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = 3 \times \delta_{a_{1,2}} \end{cases}$		Fusion de h_1 et h_2	Entre h_1 et h_2 se trouve une région codante peu conservée entre les deux espèces.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = 3 \times \delta_{a_{1,2}} + x \\ x \equiv 0 \mod 3 \\ 3 \leq x \leq k-3 \end{cases}$		Fusion de h_1 et h_2	Entre h_1 et h_2 se trouve une région codante peu conservée entre les deux espèces et un surplus d'acides aminés dans la protéine annotée sur σ.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = 3 \times \delta_{a_{1,2}} + x \\ \neg(x \equiv 0 \mod 3) \\ -2 \leq x \leq 5 \end{cases}$		Fusion de h_1 et h_2	Frameshift et petite différence dans le nombre d'acides aminés entre les deux protéines.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = 3 \times \delta_{a_{1,2}} + x \\ \neg(x \equiv 0 \mod 3) \\ 7 \leq x \leq k-1 \end{cases}$		Fusion de h_1 et h_2	Entre h_1 et h_2 se trouve une région codante peu conservée entre les deux espèces et frameshift et insertion d'acides aminés dans la protéine annotée sur σ.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = 3\delta_{a_{1,2}} + x \\ \neg(x \equiv 0 \mod 3) \\ x \leq -4 \end{cases}$		Fusion de h_1 et h_2	Entre h_1 et h_2 se trouve une région codante peu conservée entre les deux espèces et un frameshift et une perte d'acides aminés dans la protéine annotée sur σ.
$\begin{cases} \delta_{a_{1,2}} \geq 0 \\ \delta_{n_{1,2}} = -1 \end{cases}$		Fusion de h_1 et h_2	Perte d'acides aminés dans la protéine annotée sur σ.
$\begin{cases} \delta_{a_{1,2}} \geq -1 \\ \delta_{n_{1,2}} \leq -2 \end{cases}$		Elimination de h_2	Répétition sur σ.

TAB. 3.3 – **Pré-traitement des HSPs de protéines.** Les figures de la colonne «Représentation» représentent deux HSPs h_1 (en rouge) et h_2 (en bleu), originaires de la même séquence d'ADN génomique σ (en bas), et de la même séquence de protéine ν (en haut), et dans la configuration donnée par la colonne «Condition». La colonne «Action» explique comment le pré-traitement agit dans une telle situation, et la colonne «Raison» la raison biologique d'une telle action.

Chapitre 4

Un cadre formel pour une annotation de gènes experte

4.1 Introduction

L'annotation experte consiste essentiellement en l'application de règles heuristiques sur des ensembles d'alignements de séquences. Plus précisément elle comporte une double action particulièrement importante : la mise en relation d'ensembles d'alignements de séquences représentant la même structure de transcrit, et leur regroupement en un même objet sur la base de ces relations.

Pour modéliser cette double action, nous utilisons des multi-graphes orientés acycliques colorés, ou *DACM*, dont :

- les sommets sont des ensembles d'alignements de séquences,
- les couleurs d'arêtes multiples entre les sommets sont des relations entre ces ensembles d'alignements de séquences.

En effet, en terme de DACM la mise en relation d'ensemble alignements de séquences réalisée par l'expert correspond à un *ajout de couleurs d'arêtes* sur le DACM dont les sommets sont ces ensembles d'alignements de séquences, et le regroupement d'ensembles d'alignements de séquences sur la base des relations précédemment établies par un mécanisme de *réduction* de ce DACM.

La double action de mise en relation/réduction est en réalité récurrente dans le protocole d'annotation de l'expert humain. Automatiser celui-ci par le biais de DACMs nécessite donc de pouvoir enchaîner la double action automatique d'ajout de couleurs d'arêtes sur un DACM et de réduction de ce DACM. Une manière d'y parvenir est de considérer que la réduction d'un DACM produit un nouveau DACM, qui pourra à son tour subir une étape d'ajout de couleurs d'arêtes et de réduction.

Comme les règles heuristiques des étapes de mise en relation et de réduction utilisées

par l'expert sont susceptibles d'évoluer au cours du temps, nous pensons nécessaire de développer un cadre générique et incrémental pour l'annotation experte par mise en relation et regroupement d'alignements de séquences. Ce cadre prend ici la forme d'un langage, nommé DACMLang, dont l'objet d'une commande est de transformer un DACM en un autre.

La suite de ce chapitre est dédiée à la définition de la sémantique du langage DACMLang, fondée sur la sémantique dénotationnelle [105]. Dans les sections 4.2 et 4.3, nous définissons les objets et concepts nécessaires à la mise en place de cette sémantique, et dans la section 4.4 nous la définissons formellement.

4.2 Transmod, DACM et réduction

Les alignements de séquences manipulés par l'expert résultent de la comparaison de la séquence à annoter avec des séquences épissées codantes telles que des ADNc ou des protéines. L'empreinte de ces alignements sur la séquence à annoter constituent donc des structures approximatives de transcrits non épissées (avec introns), que nous nommons *modèles de transcrits* ou *transmods*. Une structure de transcrit est une succession d'exons, c'est-à-dire un ensemble de mots disjoints de la séquence à annoter. Comme un mot d'une séquence peut se voir comme l'ensemble des positions de ce mot sur cette séquence, un transmod peut être considéré comme l'ensemble des positions des mots correspondant à ses différents modèles d'exons, autrement dit comme un ensemble de positions. En notant N la taille maximale des séquences à annoter, et $\mathcal{N}$ l'intervalle $[0 : N]$, on obtient la définition suivante :

Définition 4.2.1
Un modèle de transcrit, ou transmod, est un ensemble d'entiers de $\mathcal{N}$. Le type transmod, noté T, est défini par :

$$T = 2^{\mathcal{N}}$$

En pratique un transmod t se décrit comme une union d'intervalles ordonnés, disjoints et maximaux, qui correspondent biologiquement à ses différents modèles d'exons :

$$t = \bigcup_{i=1}^{n} [a_i : b_i]$$

avec $1 + b_i < a_{i+1},\ i \in [1 : n-1]$,
où l'on note $[a : b]$ l'ensemble $\{\ x \in \mathbb{N} \mid a \leq x \leq b\ \}$.
Le transmod vide est l'ensemble de positions vide $\emptyset$.
La condition $1 + b_i < a_{i+1},\ i \in [1 : n-1]$ dans l'écriture d'un transmod comme union d'intervalles nous permet d'écrire cet ensemble sous une forme unique. Dans tout ce qui suit, à chaque fois qu'un transmod est écrit sous forme d'union d'intervalles, cette condition est supposée vérifiée et l'écriture est donc unique.

Les transmods correspondent aux objets biologiques mis en relation puis regroupés par l'expert. Ce sont aussi les sommets de nos DACMs.

Définition 4.2.2
Un DACM (Directed Acyclic Coloured Multigraph) est un quintuplet $G = \langle V, C_v, E, C_e, f_v \rangle$ *où :*

- V *est un ensemble de sommets inclus dans* T *(i.e. un sommet est un transmod),*
- C_v *est un ensemble de couleurs de sommets :* $\{adnc,\ prot,\ \perp\}$,
- C_e *est un ensemble de couleurs d'arêtes,*
- $E \subseteq V \times C_e \times V$ *est un ensemble d'arêtes orientées colorées,*
- $f_v : V \mapsto C_v$ *est la fonction de couleur de sommet.*

Il faut noter que le sommet d'un DACM est un transmod. La couleur d'un sommet permet de connaître le type des molécules dont les alignements d'un transmod proviennent. Ainsi un sommet porte la couleur :

- *adnc* si le transmod associé est constitué uniquement d'alignements d'ADNc,
- *prot* si le transmod associé est constitué uniquement d'alignements de protéines,
- $\perp$ si le transmod associé est constitué à la fois d'alignements d'ADNc et de protéines.

Le tableau 4.1 liste les principales caractéristiques d'un DACM en les expliquant par des éléments du protocole d'annotation de l'expert. L'ensemble des DACMs est noté $\mathbb{D}$.

Caractéristique d'un DACM	**Raison**
une arête peut posséder plusieurs couleurs[1]	l'expert établit plusieurs relations entre deux transmods.
un sommet possède une couleur	l'expert juge un transmod en fonction du type des molécules dont les alignements sous-jacents proviennent.
les arêtes sont orientées	l'orientation de la transcription de 5' en 3' le long de la molécule d'ADN conduit l'expert à inspecter les alignements de séquences dans un certain ordre, et donc à établir entre ceux-ci des relations orientées.

TAB. 4.1 – **Caractéristiques d'un DACM.**

L'expert cherche à regrouper des transmods qui représentent la même structure de transcrit en s'appuyant sur les relations qu'il a précédemment établies entre eux. Pour cette raison

[1] Ou, de manière équivalente, il peut exister plusieurs arêtes de couleur différente entre deux mêmes sommets.

il peut lui être utile de savoir quelles relations il a établi entre deux transmods, ce qui en terme de DACM correspond à déterminer l'ensemble des couleurs d'arêtes présentes entre deux sommets.

Définition 4.2.3
La fonction $f_e : V \times V \mapsto 2^{C_e}$ *détermine l'ensemble des couleurs d'arêtes présentes entre deux sommets. Elle est définie par :*

$$\forall (v_1, v_2) \ \in V^2, \ f_e(v_1, v_2) = \{c \ \in C_e \mid (v_1, c, v_2) \ \in E\}$$

Les regroupements de transmods effectués par l'expert se fondent sur les relations précédemment établies entre ces transmods. Ces regroupements de transmods correspondent donc à des ensembles de sommets reliés entre eux par des couleurs d'arêtes, autrement dit à des *chemins* du DACM.

Définition 4.2.4
Soit $d = \langle V, C_v, E, C_e, f_v \rangle$ *un DACM, un chemin* p *de* d *est tel que :*

$$\begin{cases} p = v_1.v_2.\cdots.v_m, \ v_i \in V, \\ \forall \ i \in [1 : m-1], \ \exists c_i \in C_e, \ (v_i, c_i, v_{i+1}) \in E \end{cases}$$

Un chemin est donc une suite de sommets, et donc un mot de V^*. Rappelons que X^* dénote le monoïde libre sur X [16]. Un chemin ne contenant aucun sommet est le chemin vide noté ϵ en référence au mot vide.

Dans toute la suite nous adoptons les notations suivantes :
- $d = \langle V, C_v, E, C_e, f_v \rangle$ est un DACM,
- $P \subset V^*$ est l'ensemble des chemins de d,
- $P^+ = P - \{\epsilon\}$ est l'ensemble des chemins non vides de d.

Définition 4.2.5
La fonction last de dernier élément d'un chemin est telle que :
- $last : P^+ \mapsto V$
- $\forall p = v_1.\cdots.v_m \ \in P^+, \ last(p) = v_m$

Un regroupement de transmods du DACM d est un chemin de d auxquels on peut associer un transmod.

Proposition 4.2.1
Soit $p = v_1.v_2.\cdots.v_m$ *un chemin de* d, $\bigcup_{i=1}^{m} v_i$ *est un transmod.*

Preuve :
$\forall \ i \in [1 : m-1], \ v_i \in V$
$\Rightarrow \forall \ i \in [1 : m-1], \ v_i \in T$
$\Rightarrow \forall \ i \in [1 : m-1], \ v_i \subset \mathcal{N}$
$\Rightarrow \bigcup_{i=1}^{m} v_i \ \subset \ \mathcal{N}$
$\Rightarrow \bigcup_{i=1}^{m} v_i \ \in \ T.$

Définition 4.2.6
Soit $p = v_1.v_2.\cdots.v_m$ un chemin de d, $\bigcup_{i=1}^{m} v_i$ est le transmod associé à p.

Le transmod associé à un chemin est l'union des transmods des sommets de ce chemin, il représente donc un objet biologique «plus complexe» que ces derniers. Notons que le transmod associé au chemin vide ϵ est naturellement le transmod vide $\emptyset$. Nous définissons la fonction de transmod associé à un chemin.

Définition 4.2.7
La fonction Tr de transmod associé à un chemin est définie par :
- $Tr : P \mapsto T$
- $\forall p \in P,\ p = v_1.\cdots.v_m\ :\ Tr(p) = \bigcup_{i=1}^{m} v_i$

Soit $Tset$ un ensemble de transmods et d le DACM dont les sommets sont les éléments de $Tset$. La double action de mise en relation et de regroupement des éléments de $Tset$ effectuée par l'expert se modélise par l'ajout de couleurs d'arêtes sur d suivie de la définition d'un ensemble de chemins de d, qui deviennent de nouveaux transmods. Comme cette double action est récurrente on la modélise par une fonction qui prend en entrée un DACM et rend en sortie un autre DACM. En effet celui-ci pourra à son tour subir un ajout de couleurs d'arêtes et une réduction. Nous définissons donc la notion de *DACM réduit d'un autre DACM selon un ensemble de chemins.*

Définition 4.2.8
Soit $\rho = \{p_1, \cdots, p_l\}$ un ensemble de chemins de d, le DACM $D = \langle \mathcal{V}, \mathcal{C}_v, \mathcal{E}, \mathcal{C}_e, \mathrm{F}_v \rangle$ réduit de d selon ρ possède les propriétés suivantes :
- $\mathcal{V} = \{\mathsf{V}_1, \mathsf{V}_2, \cdots, \mathsf{V}_l\}$ *où* $\mathsf{V}_i = Tr(p_i) \in T$
- $\mathcal{C}_v = C_v = \{adnc, prot, \perp\}$
- $\mathrm{F}_v : \mathcal{V} \mapsto \mathcal{C}_v\ /\ \forall \mathsf{V} \in \mathcal{V},\ \mathsf{V} = Tr(p),\ p = v_1.\cdots.v_m\ v_i \in V\ :$
  $\mathrm{F}_v(\mathsf{V}) = \begin{cases} c\ si\ f_v(v_i) = c,\ i \in [1:m] \\ \perp\ \ sinon \end{cases}$

Les sommets du DACM réduit D correpondent à des chemins de d, et donc à des transmods «plus complexes» que ceux de d. L'étape suivante est, du point de vue de l'expert de mettre en relation ces nouveaux transmods, et du point de vue des DACMs d'ajouter des couleurs d'arêtes sur D. Comme cette étape implique la comparaison de transmods, nous avons besoin de définir certaines fonctions sur T.

4.3 Fonctions sur les transmods

Un transmod est un ensemble de positions bornées de la séquence à annoter. Les fonctions de début et de fin donnent les positions extrêmes d'un tel ensemble.

Définition 4.3.1
Le début est une fonction deb, définie par :

- $deb : T \mapsto \mathbb{N}$
- $\forall A \in T,\ deb(A) = min(A)$

Définition 4.3.2
La fin est une fonction fin, définie par :
- $fin : T \mapsto \mathbb{N}$
- $\forall A \in T,\ fin(A) = max(A)$

Pour un transmod A, l'intervalle $[deb(A) : fin(A)]$ est appelée *étendue génomique* de A.

Pour savoir si deux transmods appartiennent au même gène ou représentent le même transcrit biologique, il peut être utile de connaître la position relative des étendues génomiques de ces transmods, ou de leurs modèles d'exons. Ces relations entre intervalles de positions rappellent les relations définies par Allen entre des intervalles de temps. En effet le système d'Allen représente le temps par des intervalles bornés, et a défini 13 relations entre deux intervalles t et s. Ces 13 relations sont en fait 7 relations de base et leur inverse. Elles sont indiquées en figure 4.1.

La notion de «chevauchement» entre deux intervalles que nous utilisons par la suite correspond à l'ensemble des relations de base définies par Allen exceptée la relation «avant». Nous définissons ici trois types de chevauchement différents entre deux transmods :
- le chevauchement faible : équivaut au chevauchement des étendues génomiques des deux transmods,
- le chevauchement fort : équivaut à l'existence d'un exon e_1 dans le premier transmod et à l'existence d'un exon e_2 dans le deuxième transmod tels que e_1 et e_2 se chevauchent,
- le chevauchement interne : équivaut à l'absence d'exons du deuxième transmod strictement inclus dans un intron du premier transmod sur l'étendue génomique de ce dernier.

Définition 4.3.3
Le chevauchement faible est un prédicat $\triangledown$, défini par :
- $\triangledown : T \times T \mapsto \mathbb{B}$
- $\forall (A, B) \in T^2,\ \triangledown(A, B) \Leftrightarrow ([deb(A) : fin(A)] \cap [deb(B) : fin(B)] \neq \emptyset)$

Définition 4.3.4
Le chevauchement fort est un prédicat $\blacktriangledown$, défini par :
- $\blacktriangledown : T \times T \mapsto \mathbb{B}$
- $\forall (A, B) \in T^2,\ \blacktriangledown(A, B) \Leftrightarrow (A \cap B \neq \emptyset)$

Définition 4.3.5
Le chevauchement interne est un prédicat $\Diamond$, défini par :
- $\Diamond : T \times T \mapsto \mathbb{B}$
- $\forall (A, B) \in T^2,\ A = \cup_{i=1}^{n} [d_i^A : f_i^A],\ R(B, A) = \cup_{j=1}^{p} [d_j^B : f_j^B]$:
  $\Diamond(A, B) \Leftrightarrow (\nexists j \in [1 : p],\ \forall\ i \in [1 : n],\ [d_j^B : f_j^B] \cap [d_i^A : f_i^A] = \emptyset)$

nom	anglais	notation	Schéma	définition
égal	equal	t = s	|-t-| |-s-|	$(t^- = s^-) \wedge (t^+ = s^+)$
avant	before	t < s	|-t-| |-s-|	$t^+ < s^-$
recouvrement	overlap	t o s	|-t-| |-s-|	$(t^- < s^-) \wedge (t^+ > s^-) \wedge (t^+ < s^+)$
rencontre	meets	t m s	|-t-| |-s-|	$t^+ = s^-$
pendant	during	t d s	|-t-| |---s---|	$(t^- > s^-) \wedge (t^+ < s^+)$
départ	starts	t s s	|-t-| |---s---|	$(t^- = s^-) \wedge (t^+ < s^+)$
fin	finishes	t f s	|-t-| |---s---|	$(t^- > s^-) \wedge (t^+ = s^+)$

FIG. 4.1 – **Sept relations de base entre deux intervalles t et s.** $t-$ et $s-$ désignent la borne inférieure des intervalles t et s respectivement, et $t+$ et $s+$ la borne supérieure de ces intervalles. Cette figure est issue de [11]

Les transmods sont des ensembles d'alignements entre la séquence à annoter et des molécules sources. Or il existe des types de molécules sources qui produisent des alignements particulièrement précis au niveau des bornes exon-intron (les ADNc de la même espèce que la séquence à annoter par exemple). Pour savoir si deux transmods de ce type représentent le même transcrit biologique, il est nécessaire de tester l'*identité de leurs introns chevauchants*. Cette fonction nécessite celle des fonctions de complément et de restriction.

Les intervalles d'un transmod représentent les modèles d'exons d'un modèle de transcrit. Pour en obtenir les modèles d'introns, on utilise la fonction de *complément*.

Définition 4.3.6
Le complément est une fonction $\complement$*, définie par :*

– $\complement : T \mapsto T$
– $\forall A \in T,\ \complement(A) = (\mathcal{N} - A) \cap [deb(A) : fin(A)]$

Pour ne conserver d'un transmod que les positions incluses dans l'étendue génomique d'un deuxième, on utilise la fonction de *restriction*.

Définition 4.3.7
La restriction est une fonction R, définie par :
– $R : T \times T \mapsto T$
– $\forall (A, B) \in T^2,\ R(A, B) = A \cap [deb(B) : fin(B)]$

Définition 4.3.8
L'identité des introns chevauchants est une fonction $\bowtie$*, définie par :*
– $\bowtie : T \times T \mapsto \mathbb{B}$
– $\forall (A, B) \in T^2,\ \bowtie (A, B) \ \Leftrightarrow\ R(\complement(A), B) \ ==\ R(\complement(B), A)$

Si les transmods A et B ne se chevauchent pas faiblement, $\bowtie (A, B)$ est vrai.

L'ajout de couleurs d'arêtes a lieu entre les sommets d'un DACM D réduit d'un DACM $d = \langle V, C_v, E, C_e, f_v \rangle$. Comme les sommets de D sont issus de chemins de sommets de d, nous avons besoin d'«étendre» certaines des fonctions précédemment définies sur T, à V^* :

1. les fonctions deb, fin ont pour signature $f : T \mapsto \mathbb{N}$. Leur extension à V^* est définie par :
   – $f^* : V^* \mapsto \mathbb{N}$
   – $\forall p \in V^*,\ f^*(p) = f(Tr(p))$

2. les fonctions $\triangledown$, $\blacktriangledown$, $\Diamond$ et $\bowtie$ ont pour signature $f : T \times T \mapsto \mathbb{B}$. Leur extension à V^* est définie par :
   – $f^* : V^* \times V^* \mapsto \mathbb{B}$
   – $\forall p_1, p_2 \in V^*,\ f^*(p_1, p_2) = f(Tr(p_1), Tr(p_2))$

4.4 Le langage DACMLang

Une commande de DACMLang prend en entrée un DACM d et rend en sortie un DACM D réduit de d et sur lequel des couleurs d'arêtes ont été ajoutées. Notons que la réduction de d se fait selon un ensemble de chemins défini par une expression sur les couleurs d'arêtes de d. Une commande de DACMLang est de la forme :

```
expression sur les couleurs d'arêtes de d      /* définit un ensemble rho
                                                  final de chemins de d */

:>                                             /* soit D réduit de d selon
                                                  rho final */

liste de filtres condition -> couleur          /* ajoute couleur entre deux
                                                  sommets de D qui vérifient
                                                  condition */
```

donc de la forme :

```
exp :> lf
```

Plus généralement un programme en DACMLang est une suite ordonnée de commandes :

```
  exp1 :> lf1          /* entrée : DACM d1 ; sortie : DACM d2 */
; exp2 :> lf2          /* entrée : DACM d2 ; sortie : DACM d3 */
;     ...
; expn :> lfn          /* entrée : DACM dn ; sortie : DACM dn+1 */
```

où :

- exp_i est une expression sur les couleurs d'arêtes du DACM d_i,
- lf_i est une liste de filtres *condition* -> *couleur* où :
  - *condition* porte sur deux sommets du DACM d_{i+1},
  - *couleur* est une couleur d'arête du DACM d_{i+1}.

La figure 4.2 illustre un tel programme.

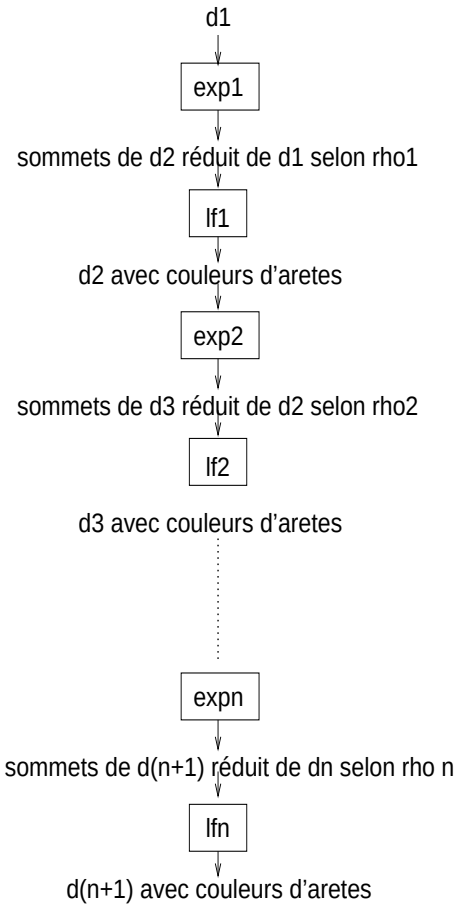


FIG. 4.2 – **Un programme en DACMLang.**

Pour `c` et `C` des identifiants de couleurs d'arêtes, $\mathtt{E}_c$ un identifiant d'ensemble de couleurs d'arêtes, et `x` un identifiant de couleur de sommet, la syntaxe précise de DACMLang est la suivante :

```
cmd_i ∈ CMD, i∈[1 : n]
exp, exp_1, exp_2 ∈ EXP
filter_j ∈ FILTER, j∈[1 : p]
logic, logic_1, logic_2 ∈ LOGIC
tm, tm_1, tm_2 ∈ TRANSMOD

PROG ::= cmd_1 ;   ···   ; cmd_n

CMD ::= exp :> | filter_1 | ··· | filter_p

EXP ::= c
       |[x]
       |exp_1 && exp_2
       |exp_1 || exp_2
       |exp_1 . exp_2
       |exp +

FILTER ::= logic -> C

LOGIC ::= tm_1 <* tm_2
        | tm_1 =<* tm_2
        | tm_1 >* tm_2
        | tm_1 >=* tm_2
        | tm_1 == tm_2
        | tm_1 >>- tm_2
        | tm_1 >>+ tm_2
        | tm_1 >>= tm_2
        | tm_1 >< tm_2
        | logic_1 && logic_2
        | logic_1 || logic_2
        | !logic

TRANSMOD ::= <>
           |5'
           |5'[E_c]
           |3'
           |3'[E_c]
```

Dans toute la suite nous utiliserons la λ-notation suivante :

$$\lambda\ x \longrightarrow\ exp$$

représentant une fonction qui à une valeur donnée x associe la valeur exp.

Sémantique de EXP.

- `EXP` représente une expression sur les couleurs d'arêtes d'un DACM d,

- `EXP` prend en entrée un DACM $d = \langle V, C_v, E, C_e, f_v \rangle$ ainsi que l'ensemble des chemins réduits à un sommet de d : $\{v\}_{v \in V}$, et rend en sortie un ensemble de chemins de d,

- la sémantique de `EXP` a pour signature :

$$\mathcal{E}[\![\ . \ ]\!] \ : \ \mathbb{D} \rightarrow 2^{V^*} \rightarrow 2^{V^*}$$

- la syntaxe de `EXP` comporte les six règles de sémantique suivante :

1. $\mathcal{E}[\![\texttt{c}]\!]_\rho = \lambda \ d \ \rho \ \longrightarrow \ \{ \ p.v \mid p \in \rho, \ v \ \in \ V, \ c \ \in f_e(last(p), v) \ \}$

2. $\mathcal{E}[\![\texttt{[x]}]\!]_\rho = \lambda \ d \ \rho \ \longrightarrow \ \{p \ \in \rho \mid f_v(last(p)) = x\}$

3. $\mathcal{E}[\![exp_1 \ \texttt{\&\&} \ exp_2]\!] = \lambda \ d \ \rho \ \longrightarrow \ (\mathcal{E}[\![exp_1]\!] \ d \ \rho) \cap (\mathcal{E}[\![exp_2]\!] \ d \ \rho)$

4. $\mathcal{E}[\![exp_1 \ \texttt{||} \ exp_2]\!] = \lambda \ d \ \rho \ \longrightarrow \ (\mathcal{E}[\![exp_1]\!] \ d \ \rho) \cup (\mathcal{E}[\![exp_2]\!] \ d \ \rho)$

5. $\mathcal{E}[\![exp_1 \ . \ exp_2]\!] = \lambda \ d \ \rho \ \longrightarrow \ \mathcal{E}[\![exp_2]\!] \ d \ (\mathcal{E}[\![exp_1]\!] \ d \ \rho)$

6. $\mathcal{E}[\![exp\texttt{+}]\!] = \lambda \ d \ \rho \ \longrightarrow \ \mu \ (\ f_{d,exp} \ )$

   où :
   - $\mu : (2^{V^*} \mapsto 2^{V^*}) \mapsto 2^{V^*}$ un opérateur de point fixe sur les ensembles de chemins,

   - $f_{d,exp} : 2^{V^*} \mapsto 2^{V^*}$ une fonction sur les ensembles de chemins définie pour un ensemble de chemins ρ par :

     $$f_{d,exp}(\rho) = \{\epsilon\} \ \boxtimes \ (\mathcal{E}[\![exp]\!] \ d \ \rho)$$

     avec :
     - $\boxtimes : 2^{V^*} \times 2^{V^*} \mapsto 2^{V^*}$ est la fonction de remplacement par préfixe. $\boxtimes$ prend en entrée deux ensembles de chemins ρ_1 et ρ_2 et renvoit l'ensemble des éléments de ρ_1 qui ne sont préfixes d'aucun élément de ρ_2 augmenté de l'ensemble des éléments de ρ_2 dont des éléments de ρ_1 sont préfixes, et dont on a éliminé les chemins qui s'écrivent comme des sous-mots d'autres chemins de l'ensemble. Plus formellement $\boxtimes$ est définie pour deux ensembles de chemins ρ_1, ρ_2 par :

       $$\rho_1 \boxtimes \rho_2 = \circledast \ (\ \rho_1 - \{p_1 \ \in \rho_1 \mid \exists \ p_2 \ \in \rho_2, \ \exists \ \omega \ \in V^*, \ p_2 = p_1 \cdot \omega\} \\ \bigcup \{p_2 \ \in \rho_2 \mid \exists \ p_1 \ \in \rho_1, \ \exists \ \omega \ \in V^*, \ p_2 = p_1 \cdot \omega\} \ )$$

– $\circledast : 2^{V^*} \mapsto 2^{V^*}$ est la fonction de réduction d'ensemble de chemins. $\circledast$ prend en entrée un ensemble de chemins ρ et renvoit l'ensemble ρ privé des éléments qui sont des sous-mots d'au moins un autre élement de ρ. Plus formellement $\circledast$ est définie pour un ensemble de chemin ρ par :

$$\circledast(\rho) = \rho - \{p' \in \rho \mid \exists p \in \rho,\ \exists\, \omega_1, \omega_2 \in V^*,\ p = \omega_1.p'.\omega_2\}.$$

Sémantique de TRANSMOD.

– `TRANSMOD` représente un sommet d'un nouveau DACM D issu de la réduction d'un DACM d dont l'ensemble de sommets est noté V,

– `TRANSMOD` prend en entrée deux chemins de d, et rend en sortie un sommet de D,

– la sémantique de `TRANSMOD` a pour signature :

$$\mathcal{T}[\![\ .\ ]\!] \ :\ V^* \to V^* \to V^*$$

– la syntaxe de `TRANSMOD` comporte les cinq règles de sémantique suivante :

1. $\mathcal{T}[\![\texttt{<>}]\!] = \lambda\ a\ b\ \longrightarrow\ \epsilon$

2. $\mathcal{T}[\![\texttt{5'}]\!] = \lambda\ a\ b\ \longrightarrow\ (si\ deb^*(a) \leq deb^*(b)\ alors\ a\ sinon\ b)$

3. $\mathcal{T}[\![\texttt{5'[C]}]\!] = \lambda\ a\ b\ \longrightarrow$
   cas :
   $|f_v(a)\ \in\ C\ \wedge\ deb^*(a) \leq deb^*(b)\ \Rightarrow\ a$
   $|f_v(b)\ \in\ C\ \wedge\ deb^*(b) < deb^*(a)\ \Rightarrow\ b$
   $|vrai \Rightarrow\ \epsilon$

4. $\mathcal{T}[\![\texttt{3'}]\!] = \lambda\ a\ b\ \longrightarrow\ (si\ deb^*(a)\ >\ deb^*(b)\ alors\ a\ sinon\ b)$

5. $\mathcal{T}[\![\texttt{3'[C]}]\!] = \lambda\ a\ b\ \longrightarrow$
   cas :
   $|f_v(a)\ \in\ C\ \wedge\ deb^*(a) > deb^*(b)\ \Rightarrow\ a$
   $|f_v(b)\ \in\ C\ \wedge\ deb^*(b) \geq deb^*(a)\ \Rightarrow\ b$
   $|vrai\ \Rightarrow\ \epsilon$

Sémantique de LOGIC.

– `LOGIC` représente une condition logique sur deux sommets d'un DACM D obtenu par réduction d'un DACM d dont l'ensemble de sommets est noté V,

– `LOGIC` prend en entrée deux chemins de d et rend en sortie un booléen,

– la sémantique de `LOGIC` a pour signature :

$$\mathcal{L}[\![\ .\ ]\!]\ :\ V^* \to V^* \to \mathbb{B}$$

– la syntaxe de `LOGIC` comporte les douze règles de sémantique suivante :

1. $\mathcal{L}[\![s_1\ \texttt{<*}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ (fin^*(a)\ <\ fin^*(b))$
2. $\mathcal{L}[\![s_1\ \texttt{=<*}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ (fin^*(a) \leq fin^*(b))$
3. $\mathcal{L}[\![s_1\ \texttt{>*}\ s_2]\!] = \neg\ (\mathcal{L}[\![s_1 \leq *\ s_2]\!])$
4. $\mathcal{L}[\![s_1\ \texttt{>=*}\ s_2]\!] = \neg\ (\mathcal{L}[\![s_1\ < *\ s_2]\!])$
5. $\mathcal{L}[\![s_1\ \texttt{==}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ (a == b)$
6. $\mathcal{L}[\![s_1\ \texttt{>>-}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ \triangledown^*(a,b)$
7. $\mathcal{L}[\![s_1\ \texttt{>>+}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ \blacktriangledown^*(a,b)$
8. $\mathcal{L}[\![s_1\ \texttt{>>=}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ \Diamond^*(a,b)$
9. $\mathcal{L}[\![s_1\ \texttt{><}\ s_2]\!] = \lambda\ a\ b\ \longrightarrow\ \bowtie^*(a,b)$
10. $\mathcal{L}[\![l_1\ \texttt{\&\&}\ l_2]\!] = \lambda\ a\ b\ \longrightarrow\ (\mathcal{L}[\![l_1]\!]\ a\ b)\ \&\&\ (\mathcal{L}[\![l_2]\!]\ a\ b)$
11. $\mathcal{L}[\![l_1\ \texttt{||}\ l_2]\!] = \lambda\ a\ b\ \longrightarrow\ (\mathcal{L}[\![l_1]\!]\ a\ b)\ ||\ (\mathcal{L}[\![l_2]\!]\ a\ b)$
12. $\mathcal{L}[\![\texttt{!}\ l\ ]\!] = \lambda\ a\ b\ \longrightarrow\ \neg(\mathcal{L}[\![l\ ]\!]\ a\ b)$

Sémantique de FILTER.

– `FILTER` représente un filtre pour l'ajout d'une couleur d'arête sur un DACM D obtenu par réduction d'un DACM d dont l'ensemble de sommets est noté V,

– `FILTER` prend en entrée un ensemble de chemins de d et une couleur d'arête de D et rend en sortie un ensemble d'arêtes colorées de D,

– la sémantique de `FILTER` a pour signature :

$$\mathcal{F}[\![\ .\ ]\!]\ :\ 2^{V^*} \to \mathcal{C}_e \to 2^{V^* \times \mathcal{C}_e \times V^*}$$

– la syntaxe de `FILTER` comporte une seule règle dont la sémantique est la suivante :

$$\mathcal{F}[\![logic \to colour]\!] = \lambda\ \rho\ c\ \longrightarrow\ \{(a,b,c)\ |\ (\mathcal{L}[\![logic]\!]\ a\ b),\ a,b \in \rho\}$$

Sémantique de CMD.

- `CMD` représente une commande de DACMLang,

- `CMD` prend en entrée un DACM d et rend en sortie un DACM D réduit de d et sur lequel des couleurs d'arêtes ont été ajoutées,

- la sémantique de `CMD` a pour signature :

$$\mathcal{C}\ [\![\ .\ ]\!]\ :\ \mathbb{D} \rightarrow \mathbb{D}$$

- la syntaxe de `CMD` comporte une seule règle dont la sémantique est la suivante :
  $\mathcal{C}\ [\![\ exp\ \texttt{:>}\ \texttt{|}f_1\ \cdots\ \texttt{|}f_p\ ]\!] = \lambda\ \langle V, C_v, E, C_e, f_v\rangle\ \longrightarrow\ \langle \mathcal{V}, \mathcal{C}_v, \mathcal{E}, \mathcal{C}_e, \mathrm{F}_v\rangle$

  avec :
  $$\begin{cases} f_i\ =\ cond_i\ \texttt{->}\ c_i \\ \\ \rho = (\mathcal{E}[\![exp]\!]\ \langle V, C_v, E, C_e, f_v\rangle\ \{v\}_{v\in V}) - \{\epsilon\} = \{p_1, \cdots, p_l\},\ p_i{\in}P \\ \\ \mathcal{V} = \{Tr(p_1), \cdots, Tr(p_l)\} \\ \\ \mathcal{C}_v = C_v \\ \\ \mathcal{E} = \bigcup_{i=1}^{p}\ (\mathcal{F}[\![f_i]\!]\ \mathcal{V}\ c_i) \\ \\ \mathcal{C}_e = \bigcup_{i=1}^{p} c_i \\ \\ \mathrm{F}_v : \mathcal{V}{\mapsto}C_v\ /\ \forall \mathsf{V} \in \mathcal{V},\ V = Tr(p),\ p = v_1.\cdots.v_m,\ v_i{\in}V\ : \\ \mathrm{F}_v(\mathsf{V}) = \begin{cases} c\ si\ f_v(v_i) = c,\ i{\in}[1:m] \\ \bot\ \ sinon \end{cases} \end{cases}$$

Sémantique de PROG.

- `PROG` représente un programme en DACMLang, c'est-à-dire une suite de n commandes,

- `PROG` prend en entrée un DACM d_1 et rend en sortie un DACM d_{n+1} issu de l'action en cascade des n commandes du programme à partir du DACM d_1 (voir figure 4.2 page 98),

- la sémantique de `PROG` a pour signature :

$$\mathcal{P}[\![\ .\ ]\!]\ :\ \mathbb{D} \rightarrow \mathbb{D}$$

- la syntaxe de `PROG` comporte une seule règle dont la sémantique est la suivante :
  $\mathcal{P}\ [\![\ cmd\ \ ;\ \ cmds\ ]\!]\ =\ \mathcal{P}\ [\![\ cmds\ ]\!]\ \circ\ \mathcal{P}\ [\![\ cmd\ ]\!]$
  $\mathcal{P}\ [\![\ cmd\ ]\!]\ =\ \lambda\ d\ \longrightarrow\ \mathcal{C}\ [\![\ cmd\ ]\!]\ d$

ou, de manière équivalente :
$\mathcal{P}[\![\ cmd_1\ ;\ cmd_2\ ;\ \cdots\ ;\ cmd_n\ ]\!] = \lambda\, d_1 \longrightarrow$
$\mathcal{C}[\![\ cmd_n\ ]\!](\ \cdots(\mathcal{C}[\![\ cmd_2\ ]\!]\ (\ \mathcal{C}[\![\ cmd_1\ ]\!]\ d_1\ )\ )\ \cdots\ )$

Les ensembles utilisés dans nos définitions sont tous des domaines au sens de la théorie des domaines [78]. Les opérations que nous avons utilisées dans nos définitions sont donc toutes continues. Ainsi les fonctions sémantiques $\mathcal{E},\ \mathcal{T},\ \mathcal{L},\ \mathcal{F},\ \mathcal{C}, \mathcal{P}$ sont bien définies.

Dans le chapitre précédent, nous avons décrit la démarche de l'annotateur expert. Il en est ressorti une action fondamentale car récurrente : la mise en relation et le regroupement d'alignements de séquence. Dans ce chapitre nous avons formalisé cette annotation de gènes par mise en relation et regroupement d'alignements de séquences. Ce cadre formel prend la forme d'un langage dédié, DACMLang, dont le but est de permettre à chaque expert en annotation de gènes, de générér son propre programme d'annotation. Plus précisément un programme écrit en DACMLang équivaut au coeur d'un programme d'annotation tel qu'Exogean (cf chapitre 3), autrement dit à une *annotation experte par DACM*.

Dans le chapitre suivant, nous décrivons l'application de cette formalisation : les résultats du programme Exogean sur des séquences d'ADN génomique.

Chapitre 5

Résultats et discussion

Nous présentons ici deux types de résultats : les performances d'Exogean évaluées dans le cadre du concours EGASP'05 sur des régions du génome humain, et l'influence des ressources utilisées par Exogean sur ses performances sur le chromosome 22 humain.

5.1 Performances d'Exogean au concours EGASP'05

Exogean a été évalué dans le cadre du concours EGASP'05 (Encode Genome Annotation Assessment Project [65]), les 6 et 7 mai 2005 au Sanger Institute à Hinxton (U.K.). EGASP'05 est lié au projet ENCODE (ENCyclopedia Of DNA Elements), lancé en septembre 2003 par l'institut national américain de recherche sur le génome humain (NHGRI), avec pour but d'identifier tous les éléments fonctionnels présents dans le génome humain. La phase pilote de ce projet s'attache à analyser celui-ci particulièrement finement sur une petite échelle : 1% du génome humain, donc environ 30 mégabases (Mb) d'ADN, réparti sur 44 régions appelées «régions ENCODE» [2, 52].

Le concours EGASP'05. Le concours EGASP'05 [75, 65] est lié au projet GENCODE, sous-projet d'ENCODE, dont le but est d'identifier l'ensemble des *gènes* codant pour des protéines contenus dans les régions ENCODE [3]. Plus précisément ce travail est réalisé par une équipe d'annotateurs experts : l'équipe «Havana» du Sanger Institute [10]. Les gènes identifiés par cette équipe seront appelés «gènes Havana».

Le concours EGASP'05 a été organisé avec un double but : d'une part évaluer dans quelle mesure les programmes d'annotation actuels parviennent à identifier les gènes codant pour des protéines annotés manuellement, et d'autre part déterminer nos connaissances sur le contenu en gènes du génome humain. Les prédictions ont donc été évaluées sur leur capacité à la fois à détecter les gènes Havana, et à prédire de *nouveaux* transcrits, non identifiés par Havana.

De façon pratique, les gènes Havana de 13 régions ENCODE parmi les 44 ont été révélées aux participants avant le concours, afin de servir de *régions d'«apprentissage»*, et l'évaluation n'a porté que sur les 31 régions restantes, qui représentent environ 21 Mb du génome humain.

Ouvert à tous, le concours EGASP'05 a rassemblé 18 équipes d'annotation à travers le monde, dont chacune pouvait décider de participer par l'intermédiaire d'un ou plusieurs programmes d'annotation de gènes. On dénombre ainsi un total de 26 programmes d'annotation de gènes, divisés en 7 catégories (tableau 5.1 page 118). Exogean utilise des ADNc humains et des protéines de souris pour annoter les gènes humains, il fait donc partie de la catégorie 3.

Evaluation des performances. Pour évaluer les prédictions d'un programme en prenant comme référence les annotations Havana, il est nécessaire de définir les trois nombres suivants (figure 5.1) :

- VP = nombre de vrais positifs = prédictions coïncidant avec une annotation,
- FN = nombre de faux négatifs = annotations ne coïncidant avec aucune prédiction,
- FP = nombre de faux positifs = prédictions ne coïncidant avec aucune annotation.

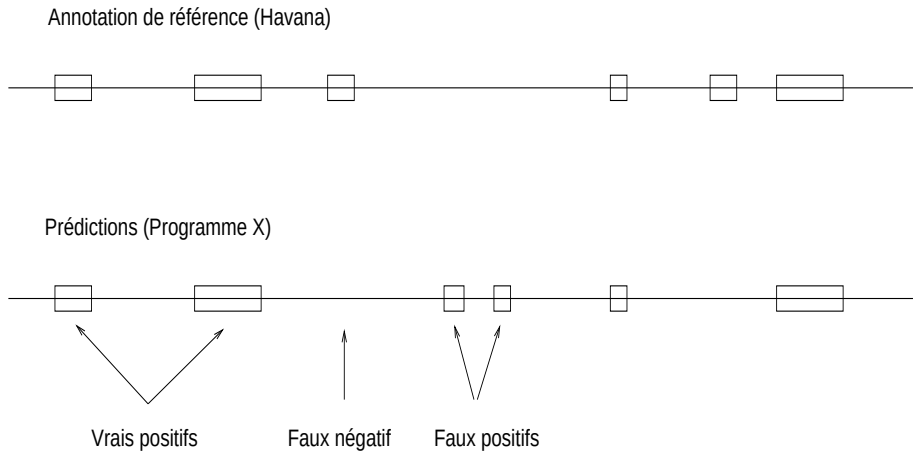


FIG. 5.1 – **Définitions utiles au calcul des performances.**

En effet les deux mesures standard communément utilisées pour l'évaluation sont la sensibilité et la spécificité, qui se calculent en fonction des trois nombres précédents de la manière suivante :

- la *sensibilité* représente le pourcentage d'annotations identifiées par les prédictions :

$$Sn = \frac{VP}{VP + FN}$$

- la *spécificité* représente le pourcentage de prédictions qui identifient une annotation :

$$Sp = \frac{VP}{VP + FP}$$

Ces formules montrent qu'il est particulièrement difficile pour un programme d'avoir à la fois une sensibilité et une spécificité élevées. En effet si sa sensibilité est élevée, cela signifie qu'il prédit beaucoup, et donc a plus de chance de prédire des objets faux (faux positifs FP), ce qui diminuera sa spécificité.

L'évaluation de la sensibilité et de la spécificité s'appliquent à quatre types d'objets différents :
- le nucléotide,
- l'exon (= succession de nucléotides contigus),
- le transcrit (= succession d'exons contigus),
- le gène (= succession de transcrits chevauchants).

Notons que les performances en transcrit et en gène sont nouvelles par rapport aux évaluations précédentes [45, 77, 33, 121]. Elles sont particulièrement importantes puisqu'elles permettent de prendre en compte l'agencement des exons en transcrits et le fait qu'un même gène puisse donner lieu à plusieurs transcrits alternatifs.

Trois remarques importantes doivent être ajoutées au sujet de l'évaluation :
- chaque objet évalué n'est considéré que s'il fait *partie du CDS d'un transcrit*,
- l'évaluation des objets de type x est toujours précédée par l'élimination de la redondance dans les objets de type x, à la fois dans les annotations et dans les prédictions,
- pour chaque type d'objet, les performances sont calculées en terme de *chevauchement* (performances dites *en prédictions chevauchantes*), et de *correspondance exacte* (performances dites *en prédictions exactes*).

Les performances en prédictions exactes des différents programmes dans les quatres objets nucléotide (N), exon (E), transcrit (T) et gène (G), ainsi que les nombres d'exons par transcrit (ExT) et de transcrits par gène (TrG), ont été calculés par le programme Eval [89], et sont indiquées dans le tableau 5.2 page 119. D'autre part le tableau 5.3 page 120 donne les performances d'Exogean de façon plus détailée.

5.1.1 Performances d'Exogean en prédictions chevauchantes

Les spécificités et sensibilités d'Exogean aux quatre niveaux nucléotide, exon, transcrit et gène codants, sont de plus de 94% et 82% respectivement, ce qui représente un taux de faux positif de 6%, et un taux de faux négatif de 18%. Comme les performances en gènes représentent une mesure particulièrement significative [42], nous avons analysé les faux positifs et faux négatifs en terme de *gène*.

Les gènes faux positifs. 8 gènes sur 219 prédits par Exogean (moins de 4%) ne chevauchent aucun gène Havana et sont donc considérés comme des gènes faux positifs (gènes FP). L'inspection manuelle de ces gènes FP permet de les diviser en quatre classes, représentées dans le tableau 5.4 page 120.

Nous constatons que 2 gènes sur 8 s'expriment sous forme d'ARN et représentent soit des CDS putatifs, soit des ARNs fonctionnels non codants connus. Par conséquent *seuls* 6 *gènes sur* 219 *sont réellement faux positifs.*

Les gènes faux négatifs. Sur les 296 gènes annotés par Havana, 53 (18%) ne sont chevauchés par aucun gène Exogean et sont donc considérés comme des gènes faux négatifs (gènes FN). De même que pour les gènes FP nous avons tenté de diagnostiquer la cause de ces gènes, ce qui nous a permi de les diviser en 8 classes, représentées dans le tableau 5.5 page 120.

Ainsi, les gènes FN sont majoritairement dus (33 cas sur 53) à des situations où plusieurs gènes Havana sont organisés en *cluster*. Comme ces gènes sont souvent similaires, une protéine homologue à l'un a de grandes chances d'être aussi homologue à l'autre, ce qui engendre l'alignement de cette protéine sur chacun de ces gènes. Si cette protéine est bien annotée et que les gènes du cluster sont très similaires entre eux, les alignements (HSPs) ainsi obtenus vont présenter des «retours en arrière» dans la protéine. Cela permettra à Exogean de «couper» ces alignements en autant de gènes que présents dans le cluster, grâce à la règle *ordre_genomique_molecule* du DACM_1 (section 3.4.1). Toutefois il peut arriver que cette protéine ne présente pas ou peu de retours en arrière : en ce cas Exogean englobera des alignements qui correspondent à plusieurs gènes de ce cluster dans un seul et même gène Exogean. Parmi les gènes Havana chevauchés par ce seul gène Exogean, un seul représentera un gène VP, et tous les autres seront considérés comme des gènes FN. Depuis la compétition nous nous sommes penchés sur le problème de la prédiction des gènes en cluster, et avons produit une nouvelle version d'Exogean capable de prédire correctement la majorité d'entre eux (voir section 5.1.3 pour plus de détails).

Les gènes FN sont également dus (8 gènes sur 53) à la situation où la longueur du CDS d'un transcrit est petite. En effet, Exogean élimine totalement les transcrits dont le CDS est inférieur à 300 nucléotides (section 3.5.2), car il les considère comme des pseudo-transcrits. Pour qu'Exogean prédise ces transcrits, il suffit d'abaisser ce seuil, ce que nous avons fait, et ce qui a permi d'améliorer les performances d'Exogean (voir section 5.1.3).

Enfin les six autres causes de gènes FN, plus mineures (12 gènes sur 53), sont liées à des paramètres trop stringents d'Exogean. Une nouvelle version d'Exogean prend en compte ce problème et obtient des performances en gènes encore meilleures (voir section 5.1.3).

Il faut toutefois noter que même s'il est important pour un programme de localiser des exons, transcrits et gènes de manière chevauchante, la principale difficulté réside dans l'identification de ces objets de manière *exacte*.

5.1.2 Performances d'Exogean en prédictions exactes

Un transcrit Exogean est dit *exact* s'il existe un transcrit Havana dont toutes les bornes du CDS, à savoir le début, les bornes d'épissage, et la fin, sont identiques à celles de son CDS. Un programme d'annotation de gène fiable a une forte spécificité en gène exact. Pour cette raison nous avons toujours cherché à augmenter ce chiffre, tout en négligeant le moins possible son pendant : la sensibilité en gène exact. De fait, la spécificité d'Exogean en gène exact est particulièrement élévée (80.82%), et devance de beaucoup celles des autres programmes (68% pour le deuxième) (figure 5.2). Nous nous sommes donc interrogés si cette forte spécificité en gène exact avait lieu au prix d'une faible sensibilité en gène exact. En réalité Exogean fait partie d'un groupe de sept programmes dont la sensibilité en gène exact se distingue par rapport à celle des autres programmes (entre 63% et 73% contre moins de 50%). Cela montre que la forte spécificité d'Exogean en gène exact ne se paye pas au prix d'une faible sensibilité en gène exact. De plus, si l'on utilise la mesure standard qui consiste à calculer la moyenne de la spécificité et de la sensibilité en gène exact [45], Exogean arrive en tête des 20 programmes d'annotation de gènes participants (figure 5.2).

Comme les annotateurs humains de l'équipe Havana, Exogean prédit *plusieurs transcrits alternatifs par gène*. Plus précisément, Exogean et Havana identifient en moyenne 2.34 et 2.19 transcrits par gène respectivement (tableau 5.2 page 119). Bien qu'Exogean arrive en deuxième position concernant la sensibilité en transcrits exacts, la spécificité d'Exogean en transcrits exacts est *très inférieure* à sa spécificité en gène exact (tableau 5.3 page 120). Pour l'expliquer nous avons tenté de savoir s'il existait une catégorie de transcrits Havana mieux identifiés par Exogean.

Effectivement parmi les 267 transcrits prédits de façon exacte par Exogean, 266 correspondent à des transcrits Havana dont le CDS est *complet*, c'est-à-dire commençant par une méthionine et terminant par un codon stop. Ce résultat a deux conséquences importantes : la première est le fait qu'Exogean prédit mieux les transcrits Havana dont le CDS est complet, et la deuxième qu'une partie des transcrits complets prédits par Exogean et qui chevauchent des transcrits Havana incomplets, sont probablement corrects. En effet sur 23 gènes Havana dont tous les transcrits sont incomplets, 11 sont chevauchés par un transcrit d'Exogean complet. Il n'est donc pas exclus qu'*Exogean puisse annoter complètement et correctement des transcrits qu'Havana n'a annoté que partiellement*. Dans une telle situation les transcrits d'Exogean et d'Havana devraient présenter des différences essentiellement au niveau de leurs extrémités. Pour le vérifier nous avons calculé les performances d'Exogean en exon [89] séparément pour chaque type d'exon : initial, interne, terminal (figure 5.5). Ce calcul montre que les exons internes d'Havana sont mieux identifiés de façon exacte par Exogean que ne le sont les exons initiaux et terminaux d'Havana. Cela n'est pas le cas en terme d'exons chevauchants.

Un facteur important pouvant expliquer qu'Exogean identifie mieux de manière exacte les transcrits Havana dont le CDS est complet, est le fait qu'Exogean utilise comme res-

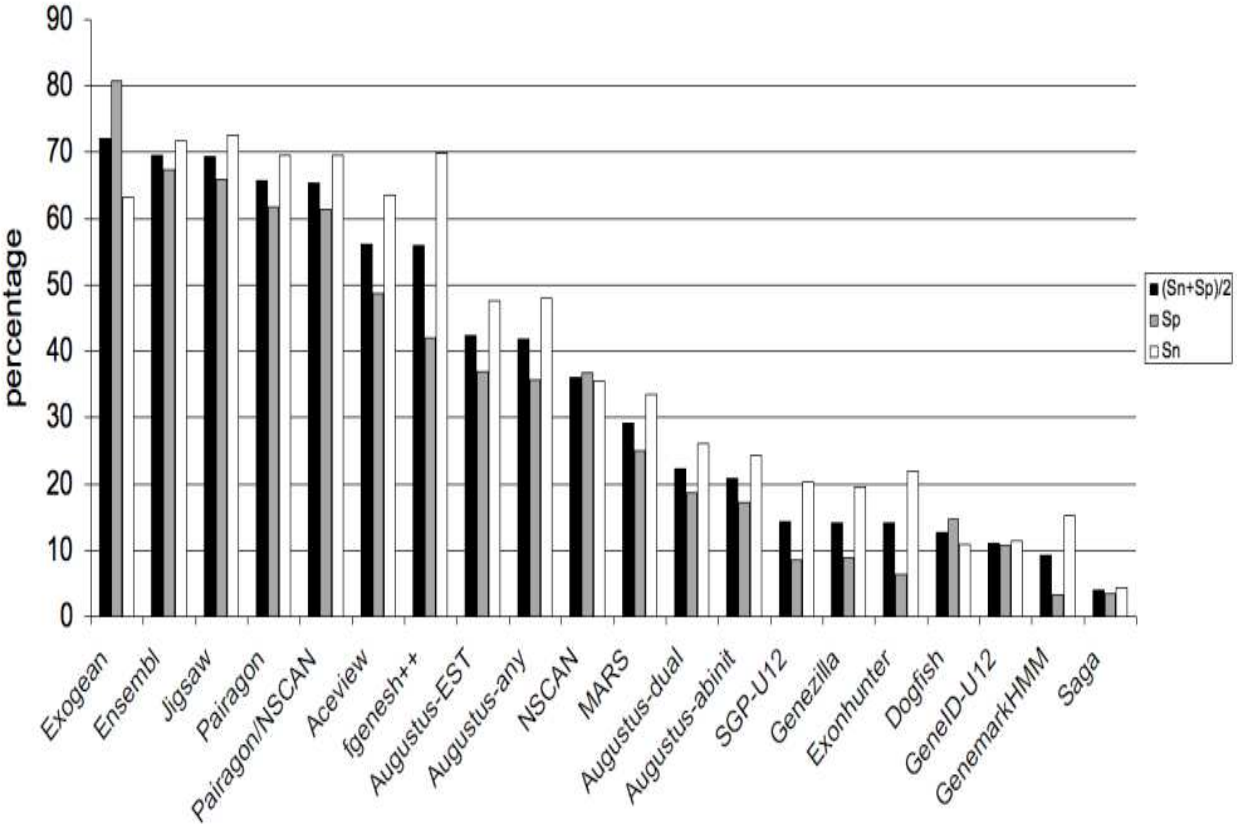


FIG. 5.2 – **Performances en gène exact des programmes participants à EGASP'05.**

sources des alignements avec des ADNc pleine longueur humains et avec des protéines de souris, mais pas avec des ESTs humains comme le fait Havana. Ainsi un transcrit annoté par Havana sur la base d'ESTs humains seulement, sera prédit par Exogean sur la base de protéines de souris uniquement. Selon la conservation avec la protéine orthologue de souris (voir annexe B), ce gène sera prédit par Exogean de façon plus ou moins complète, voire même pas prédit du tout. Une des solutions serait de prendre en compte des alignements avec des ESTs, ou encore des résultats de comparaison génomique-génomique (voir perspectives chapitre 7).

Enfin Exogean est le programme qui prédit le plus d'exons par transcrits en moyenne : 9.8 contre 8.28 pour Havana et moins de 8.6 pour les autres programmes (tableau 5.2 page 119). Cela s'explique par le fait qu'Exogean prédit moins de transcrits contenant peu d'exons

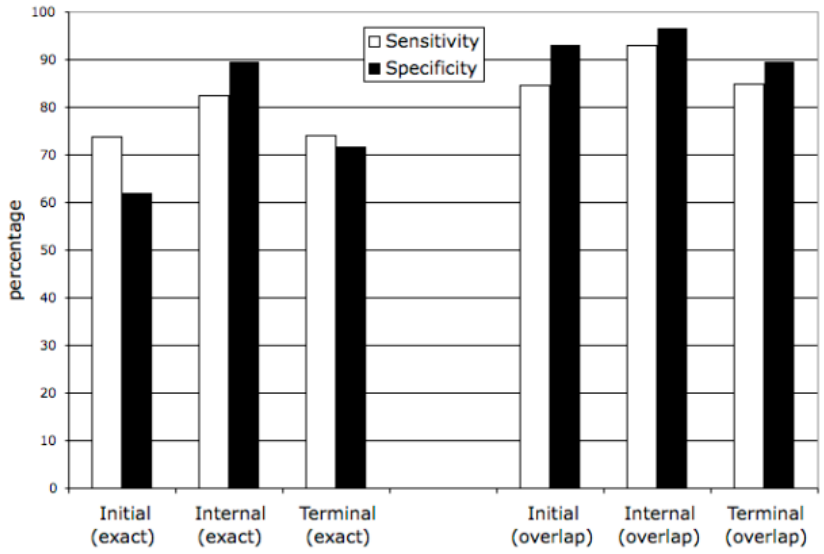


FIG. 5.3 – **Performances d'Exogean en exon selon la catégorie de l'exon.**

qu'Havana (figure 5.4). Cependant Havana et Exogean prédisent un nombre très similaire de transcrits contenant beaucoup d'exons (plus de 9), ce qui s'accompagne pour Exogean d'une plus grande sensibilité en prédictions exactes sur cette catégorie de transcrits (figure 5.5). Là encore les ressources utilisées expliquent ces observations : des transcrits contenant beaucoup d'exons ont plus de chance d'être prédits grâce à des ADNc pleine longueur que grâce à des ADNc de type EST, puisque ces derniers sont en général plus courts que les premiers (voir annexe A).

5.1.3 Une version améliorée d'Exogean

La compétition EGASP'05 nous a permi d'identifier deux principaux problèmes d'Exogean : les difficultés pour prédire des gènes paralogues en cluster, et une mauvaise définition du CDS. Depuis EGASP'05, nous nous sommes donc penchés sur ces problèmes et avons implanté de nouvelles règles pour les résoudre. Ainsi la nouvelle version d'Exogean atteint une sensibilité en gène exact de 72.64%, contre 63.18% au moment du concours, tout en gardant une spécificité en gène exacte à peu près stable (79.30% contre 80.82% au moment du concours). Cette nouvelle version d'Exogean a progressé dans la mesure standard de la moyenne entre la sensibilité et la spécificité en gène exact puisqu'elle est aujourd'hui de 76%, contre 72% au moment du concours. Les améliorations apportées à Exogean ont donc permi

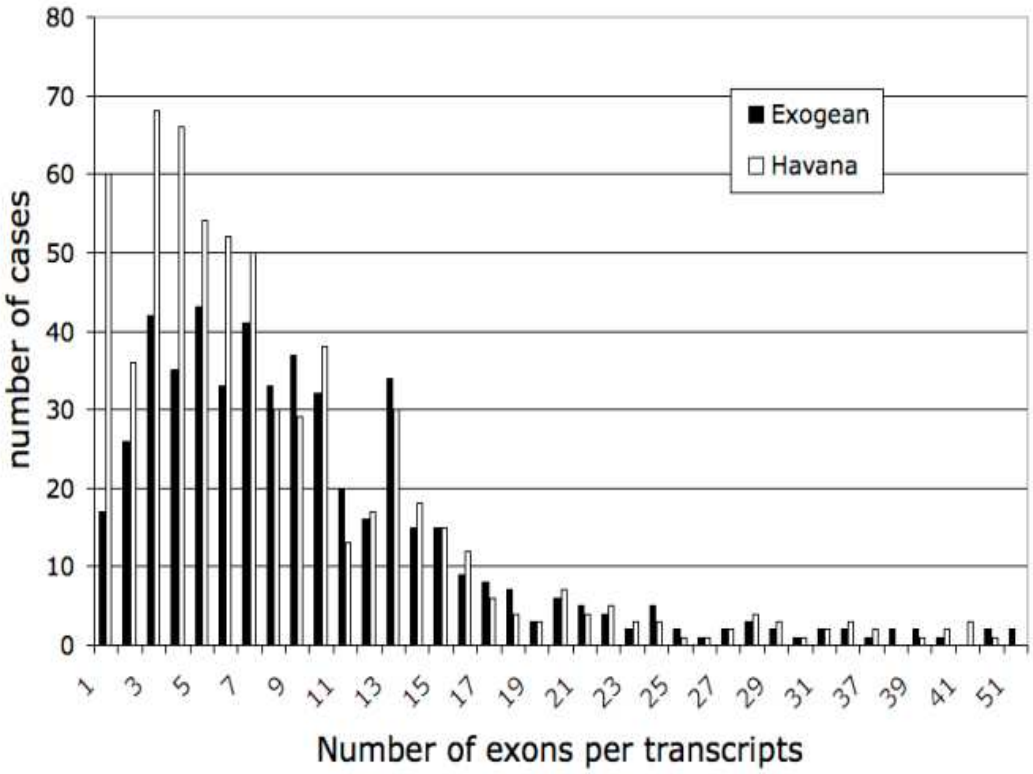


FIG. 5.4 – **Distribution du nombre d'exons dans les transcrits Havana et Exogean.**

une *progression de* 4% des performances en gène exact d'Exogean, chiffre qui plaçait déjà ce programme en tête des 20 programmes d'annotation de gènes participants.

5.1.4 Discussion

Les programmes d'annotation de gènes ont déjà été évaluées par le passé [45, 77, 33, 121, 65], cependant ces évaluations ont toujours souffert d'une faible quantité d'annotations de référence auxquelles confronter les prédictions des programmes. En ce sens, EGASP'05 représente une *opportunité unique pour évaluer rigoureusement les programmes d'annotation de gènes*. Une mesure standard communément utilisée est la suivante : un gène exact est un gène dont au moins un transcrit codant est identifié de manière totalement exacte. Comme nous l'avons vu, cette mesure place Exogean en tête des vingt programmes évalués

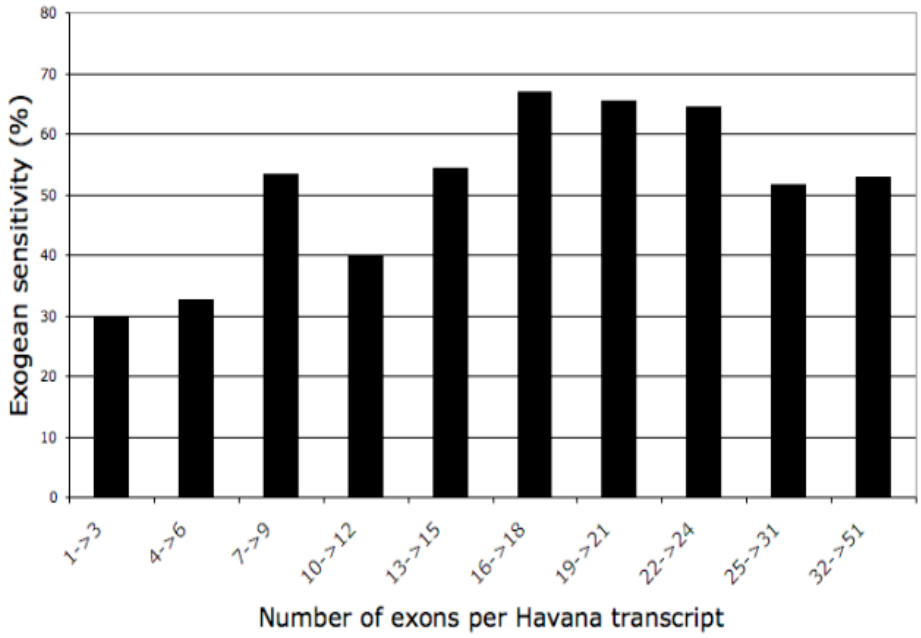


FIG. 5.5 – **Sensibilité d'Exogean en transcrit exact selon le nombre d'exons par transcrit.**

lors du concours EGASP'05. En particulier la spécificité d'Exogean selon cette mesure est la meilleure de loin (12% de plus que le deuxième meilleur). Ce résultat reflète notre volonté de concevoir un programme hautement spécifique, en négligeant toutefois le moins possible la sensibilité. Comme 8 gènes sur 10 prédits par Exogean sont exacts, Exogean pourrait être utilisé pour accélérer le protocole d'annotation de l'humain, d'autant plus qu'il utilise pour annoter les gènes des règles similaires à celles utilisées par l'expert. Pour aider les experts dans cette tâche Exogean génère, en plus des positions des transcrits et de leur séquence, des informations sur chaque gène et transcrit prédit. Ces informations résument la structure des gènes et des transcrits, les ressources utilisées pour les annoter, les problèmes et conflits rencontrés, ainsi que les solutions adoptées. Les experts humains peuvent partir de ces informations et utiliser d'autres règles, ressources ou expériences pour corriger ou confirmer ces prédictions.

Exogean est très spécifique, cependant sa spécificité pourrait être améliorée de plusieurs manières. Tout d'abord les annotations générées par Exogean pour le concours EGASP'05 se fondent uniquement sur deux sources de données dont on a réalisé un alignement sur la

séquence à annoter : des séquences d'ADNc humains et des protéines prédites à partir du génome de souris. Il n'est par conséquent pas surprenant que des programmes qui utilisent d'autres ressources (ESTs humains, ADNc d'autres espèces, ADN génomique conservé), identifient davantage de gènes qu'Exogean. D'ailleurs nous concevons actuellement de nouvelles règles permettant d'intégrer ces ressources.

Les programmes d'annotation de gènes utilisent généralement des modèles statistiques pour capturer les propriétés des gènes et les annoter dans l'ADN génomique. Ces modèles statistiques peuvent être utilisés seuls ou en combinaison avec des ressources expérimentales. Exogean n'utilise pas cette stratégie puisqu'il s'appuie uniquement sur des règles heuristiques symboliques déduites de l'expertise humaine. Ainsi Exogean ne nécessite pas d'apprentissage sur un ensemble de gènes connus.

Exogean ne peut annoter un transcrit d'une séquence d'ADN que s'il dispose d'alignements d'ADNc de la même espèce ou de protéines d'une autre espèce relatives à ce transcrit, autrement dit de *ressources*· Nous avons donc cherché à évaluer l'impact de ces ressources sur les performances d'Exogean. Pour cela, nous avons utilisé le chromosome 22 humain [62] (assemblage 34 du NCBI) comme séquence d'ADN à annoter, et comme référence les annotations manuelles générées par Vega (VErtebrate Genome Annotation [31]).

5.2 Influence des ressources sur les performances d'Exogean

Nous avons évalué l'influence sur les performances d'Exogean des deux types d'entrée suivants :

- la quantité d'alignements d'ADNc de la même espèce (ici l'humain) et de protéines d'une autre espèce (ici la souris) (section 5.2.1),
- la distance d'évolution entre l'espèce des protéines et l'espèce de la séquence à annoter (ici l'humain) (section 5.2.2).

Dans les deux cas nous avons calculé les performances d'Exogean en terme de spécificité et de sensibilité en *transcrits chevauchants*.

5.2.1 Influence de la quantité d'alignement

Afin de connaître l'influence indépendante de chaque type de ressource (ADNc de la même espèce et protéines d'une autre espèce) sur les performances d'Exogean, nous avons réalisé l'expérience suivante pour chaque type de ressource : nous avons sélectionné de manière aléatoire des quantités de la totalité des alignements allant de 10% à 100% par pas de 10%, et mesuré les performances d'Exogean en transcrits chevauchants dans ces 10 conditions. Les résultats obtenus sont donnés en figure 5.6.

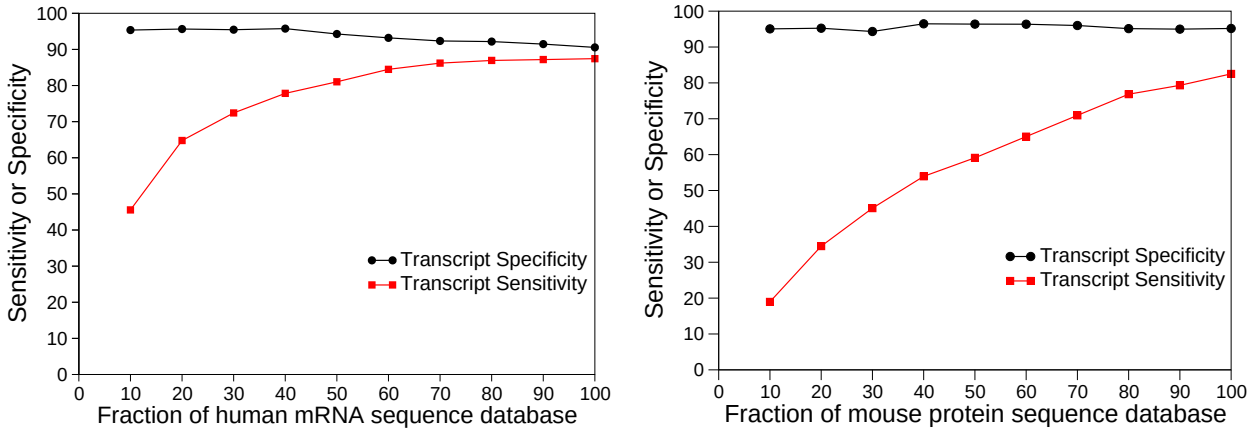


FIG. 5.6 – **Influence de la quantité d'alignement sur les performances d'Exogean.**

Nous constatons que dans le cas où la quantité d'alignements augmente, quelle que soit sa nature, la spécificité d'Exogean en transcrits chevauchants reste à peu près constante. Exogean est donc *particulièrement robuste au bruit* présent dans les séquences biologiques de grande taille. Il faut noter que cette robustesse n'a pas lieu au détriment de la sensibilité en transcrits chevauchants, puisque cette dernière augmente avec la quantité d'alignements. Notons toutefois une différence entre les deux types de ressources quant à l'évolution de la sensibilité en transcrits chevauchants : pour les ADNc de la même espèce, la courbe atteint un plateau pour une quantité d'alignements de $70 - 80\%$, alors que pour les protéines d'une autre espèce elle croît de façon à peu près linéaire jusqu'à 100%. Ajouter des alignements de protéines de souris permet donc toujours d'améliorer les performances d'Exogean en transcrits chevauchants.

5.2.2 Influence de la distance d'évolution

Dans le cas où Exogean est utilisé uniquement avec des alignements de protéines, il est prévisible que la distance d'évolution entre l'espèce des protéines utilisées et l'espèce de la séquence à annoter ait une influence sur les performances. Pour évaluer cette influence, nous avons réalisé un alignement entre d'une part le chromosome 22 humain, et d'autre par les protéines des cinq espèces suivantes :

- la souris,
- le rat,
- le poulet,
- le tetraodon,
- le poisson zèbre.

Nous avons ensuite mesuré les performances d'Exogean en transcrits chevauchants pour des alignements de protéines correspondant aux neuf espèces ou combinaisons d'espèces suivantes :

- souris,
- rat,
- poulet,
- tetraodon,
- poisson zèbre,
- souris + poulet
- poulet + tetraodon
- tetraodon + poisson zèbre,
- toutes.

Les performances d'Exogean en transcrits chevauchants dans ces neuf conditions, sont données en figure 5.7.

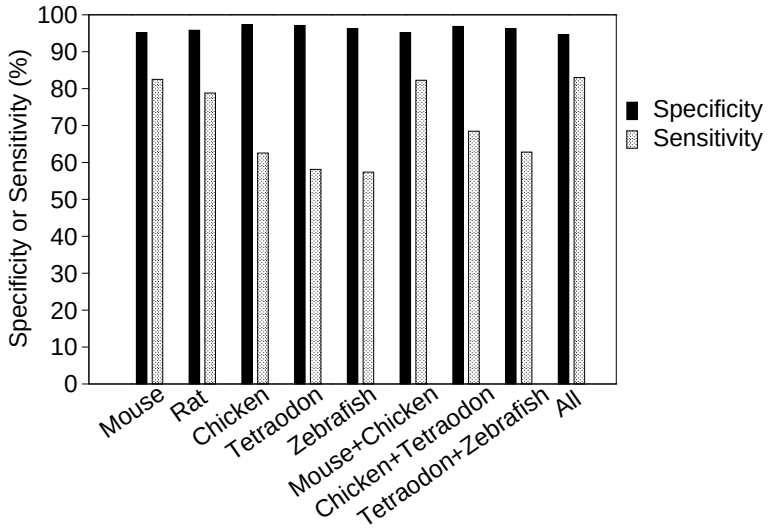


FIG. 5.7 – **Influence de la distance d'évolution sur les performances d'Exogean.**

Comme pour l'expérience précédente, nous constatons que quelque soit l'espèce ou la combinaison d'espèces utilisée, la spécificité d'Exogean reste à peu près constante. Ceci n'est pas surprenant étant donné que l'alignement a été calibré sur les protéines de souris, et que la souris est l'espèce la plus proche de l'humain. Ainsi il est peut probable d'introduire des alignements artéfactuels à mesure que la distance d'évolution augmente.

En revanche la sensibilité d'Exogean diminue quand la distance d'évolution augmente. Notons toutefois que les gènes manqués par Exogean (gènes FN) ne sont en général pas les mêmes pour les différentes espèces ou combinaisons d'espèces, puisque la combinaison poulet+tetraodon permet par exemple d'obtenir une meilleure sensibilité que l'espèce poulet seule. Notons que la combinaison des 5 espèces ne permet pas d'obtenir une meilleure sensibilité que l'espèce souris seule.

Le présent chapitre décrit et discute de manière détaillée les résultats obtenus par le programme Exogean sur des séquences d'ADN du génome humain. Nous avons d'abord décrit les performances d'Exogean au concours EGASP'05, puis l'influence des ressources prises en entrée d'Exogean sur ses performances. Toutes ces évaluations ont été faites en prenant comme référence les annotations de gènes générées par les experts humains.

Ainsi les conclusions de ce chapitre sont les suivantes :
- Exogean est le meilleur programme dans la mesure standard suivante : l'identification exacte d'au moins un transcrit par gène,
- Exogean est le programme le plus spécifique en terme de gène exact, et est donc particulièrement fiable,
- Exogean prédit mieux les transcrit Havana dont le CDS est complet. De plus une analyse plus fine des CDS complets d'Exogean qui chevauchent des CDS incomplets annotés manuellement, suggère qu'Exogean puisse être capable d'annoter complètement des transcrits que l'annotateur manuel n'a réussi à annoté que partiellement,
- Exogean a été amélioré et atteint aujourd'hui des performances en gène exact supérieures de 4% par rapport à celles obtenues au concours EGASP'05, et qui classaient déjà Exogean en tête des programmes d'annotation de gènes,
- Exogean est particulièrement robuste au bruit présent dans les séquences biologiques de grande taille.

Catégorie	Description	Programmes participants
1	utilise *tout type* d'information	– Augustus-any, – Pairagon-any, – Jigsaw [28], – fgenesh++ [123]
2	*ab initio* sur un seul génome	– Augustus-abinit [133], – GenemarkHMM [98], – Genezilla
3	utilise les *ADNc* (EST, ARNm) et les *protéines*	– Exonhunter [40], – Exogean [61], – Augustus-EST, – Aceview, – Pairagon/NSCAN [43], – Ensembl [54], – Aspic [109]
4	fondé sur la comparaison de *plusieurs génomes*	– Augustus-dual, – NSCAN [43], – Saga, – CSTminer [134], – Dogfish, – MARS
5	prédit des gènes *nouveaux*	– novel
6	prédit des gènes *non usuels* (pseudogènes, épissage non canonique, gènes petits, sans introns ..etc)	– uncover, – sbpseudo, – geneid+sgp [112]
7	ne prédit *que des exons*	– Spida, – Dogfish-exon, – Augustus-exon [133]

TAB. 5.1 – **Programmes d'annotation de gènes participant à EGASP'05.**

Cat	Programme	NSn (%)	NSp (%)	ESn (%)	ESp (%)	ExT	TSn (%)	TSp (%)	TrG	GSn (%)	GSp (%)
1	Augustus-any	94.42	82.43	74.67	76.76	6.74	22.65	35.59	1.00	47.97	35.59
1	Pairagon-any	87.77	92.78	76.85	88.91	7.33	39.29	60.34	1.28	69.59	61.32
1	Jigsaw	94.56	92.19	80.61	89.33	7.65	34.05	65.95	1.00	72.64	65.95
1	fgenesh++	91.09	76.89	75.18	69.31	6.42	36.21	41.61	1.17	69.93	42.09
2	Augustus-abinit	78.65	75.29	52.39	62.93	5.50	11.09	17.22	1.00	24.32	17.22
2	GenemarkHMM	78.43	37.97	50.58	29.01	3.47	6.93	3.24	1.00	15.20	3.24
2	Genezilla	87.56	50.93	62.08	50.25	5.21	9.09	8.84	1.00	19.59	8.84
3	Exonhunter	90.46	59.67	64.44	41.77	4.15	10.48	6.33	1.00	21.96	6.33
3	**Exogean**	**84.18**	**94.33**	**79.34**	**83.45**	**9.80**	**42.53**	**52.44**	**2.34**	**63.18**	**80.82**
3	Augustus-EST	92.62	83.45	74.10	77.40	6.94	22.50	37.01	1.00	47.64	37.01
3	Aceview	90.94	79.14	85.75	56.98	5.78	44.68	19.31	4.05	63.51	48.65
3	Pairagon/NSCAN	87.56	92.77	76.63	88.95	7.34	39.29	60.64	1.28	69.59	61.71
3	Ensembl	90.18	92.02	77.53	82.65	7.79	39.75	54.64	1.51	71.62	67.32
3	Aspic	NA	NA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	Augustus-dual	88.86	80.15	63.06	69.14	6.10	12.33	18.64	1.00	26.01	18.64
4	NSCAN	85.38	89.02	67.66	82.05	7.97	16.95	36.71	1.00	35.47	36.71
4	Saga	52.54	81.39	38.82	50.73	5.60	2.16	3.44	1.00	4.39	3.44
4	CSTminer	66.54	27.64	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
4	Dogfish	64.81	88.24	53.11	77.34	8.67	5.08	14.61	1.00	10.81	14.61
4	MARS	84.25	74.13	65.56	61.65	8.55	15.87	15.11	1.67	33.45	24.94
6	Acescan	NA	NA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	GeneID-U12	75.03	78.83	51.41	63.92	6.99	5.39	10.69	1.00	11.49	10.69
6	SGP-U12	82.84	66.80	59.73	49.20	4.72	9.71	8.44	1.00	20.27	8.44
7	Spida	35.99	94.25	29.81	35.09	1.00	0.00	0.00	1.00	0.00	0.00
7	Dogfish-exon	8.065	95.77	1.66	27.22	1.00	0.00	0.00	1.00	0.00	0.00
7	Augustus-exon	94.42	82.43	39.80	40.89	1.00	0.00	0.00	1.00	0.00	0.00

TAB. 5.2 – **Performances des programmes en prédictions exactes.**

Mesure	Sensibilité (%)	Spécificité (%)	Exogean correct	Exogean faux	Havana non vus	Havana total
			Chevauchant			
Nucléotides	84.18	94.33	371369	22301	69791	441160
Exons	90.12	94.94	2495	133	273	2764
Transcrits	89.37	97.08	498	15	151	649
Gènes	82.09	96.35	211	8	53	296
			Exact			
Nucléotides	84.18	94.33	371369	22301	69791	441160
Exons	79.34	83.45	2193	435	571	2764
Transcrits	42.53	52.44	267	237	373	649
Gènes	63.18	80.82	187	32	109	296

TAB. 5.3 – **Performances d'Exogean à EGASP'05.**

Classe de gène FP	Nombre de cas	Cause du problème
1	2	Elément rétrotransposable révélé par au moins un ADNc
2	1	ARN non codant
3	1	Transcrit putatif Havana révélé par un ADNc
4	4	Pseudogène

TAB. 5.4 – **Les gènes faux positifs d'Exogean.**

Classe de gène FN	Nombre de cas	Cause du problème
1	33	Protéine(s) qui s'aligne(nt) au niveau de plusieurs gènes voisins en cluster
2	8	CDS Havana de longueur inférieure à 300 nucléotides
3	3	CDS Havana interrompu par le début ou la fin d'une région ENCODE
4	2	Trop peu de ressources pour prédire le gène
5	3	Protéine éliminée par Exogean
6	1	ADNc éliminé par Exogean
7	2	Site donneur/accepteur d'épissage non toléré par Exogean
8	1	ADNc mal alignée

TAB. 5.5 – **Les gènes faux négatifs d'Exogean.**

Chapitre 6

Implantation

Ce chapitre décrit l'implantation d'Exogean de façon plus détaillée. Exogean est écrit en langage Objective CAML (OCAML) [19], version 3.8. OCAML est un langage fonctionnel qui intègre les principales caractéristiques des langages actuels, et est actuellement développé dans l'équipe Gallium de l'INRIA de Rocquencourt dirigée par Xavier Leroy [1]. OCAML s'appuie sur une longue expérience de conception de langages de la famille ML, et possède l'immense avantage (efficacité du programme et sûreté de l'exécution) d'être statistiquement typé.

6.1 Principales étapes d'Exogean

Exogean (EXpert on GEne ANnotation) est un environnement logiciel qui permet d'annoter les transcrits alternatifs des gènes d'une séquence d'ADN génomique eucaryote. A cette fin Exogean applique des règles heuristiques sur des alignements de séquences, de la même façon que la ferait un annotateur humain.

Ainsi Exogean prend en entrée la séquence à annoter ainsi que les résultats d'un alignement local (par exemple Blat [90]) entre d'une part la séquence à annoter et d'autre part un ensemble d'ADNc de la même espèce et/ou un ensemble de protéines d'une autre espèce.

Exogean se compose de quatre étapes principales (figure 6.1) :

1. le **pré-traitement** des HSPs d'ADNc et de protéines, qui élimine les HSPs artéfactuels et rend l'ensemble des HSPs propres à la constitution de structures de transcrits, ou transmods de niveau 1 (TM1)
2. l'**annotation experte par DACM**, qui produit les structures des transcrits alternatifs des gènes de la séquence à annoter à partir des HSPs pré-traités. Cette étape se compose de trois sous-étapes dont chacune se fonde sur la réduction d'un multi-graphe orienté acyclique coloré ou DACM :
   - la génération des modèles de transcrits mono-molécule ou transmods de niveau 2 (TM2), fondée sur la réduction du DACM1,

- la génération des modèles de transcrits multi-molécule mono-type, ou transmods de niveau 3 (TM3), fondée sur la réduction du DACM2,
- la génération des modèles de transcrits multi-molécule multi-type, ou transmods de niveau 4 (TM4), fondée sur la réduction du DACM3.

3. la **recherche du CDS** (Coding Sequence), qui détermine pour chaque chaque transcrit les bornes des exons qui codent les acides aminés de la protéine associée à ce transcrit,

4. le **filtre transcrit/pseudo-transcrit**, qui élimine les pseudogènes de ces structures.

6.2 Complexité en temps de l'annotation experte par DACM

L'annotation experte par DACMs (étape 2 d'Exogean, figure 6.1) se décompose en trois étapes dont chacune procède à une réduction de DACM. En réalité chaque réduction de DACM est précédée d'une étape d'ajout de couleurs d'arêtes sur ce DACM, ce qui donne un total de six étapes. Ces six étapes permettent de constituer des transmods (TM) de plus en plus complexes (niveaux 2, 3 et 4) à partir des HSPs pré-traités ou transmods de niveau 1 (TM1). Ces six étapes sont décrites en figure 6.2.

Les complexités en temps de chacune de ces six étapes sont données au tableau 6.1.

Etape	**Tâche accomplie**	**Complexité**
Ajout de couleurs d'arêtes dans le DACM1	Tri selon molécule et ordre génomique	$\begin{cases} \theta(N_1 \times log(N_1)) \\ N_1 = Nombre\ de\ TM1 \end{cases}$
Réduction du DACM1	Parcours de graphe filiforme	$\begin{cases} \theta(N_1) \\ N_1 = Nombre\ de\ TM1 \end{cases}$
Ajout de couleurs d'arêtes dans le DACM2	Tri selon ordre génomique	$\begin{cases} \theta(N_2 \times log(N_2)) \\ N_2 = Nombre\ de\ TM2 \end{cases}$
Réduction du DACM2	Parcours de graphe filiforme	$\begin{cases} \theta(N_2) \\ N_2 = Nombre\ de\ TM2 \end{cases}$
Ajout de couleurs d'arêtes dans le DACM3	Tri selon ordre génomique	$\begin{cases} \theta(N_3 \times log(N_3)) \\ N_3 = Nombre\ de\ TM3 \end{cases}$
Réduction du DACM3	Parcours de graphe filiforme	$\begin{cases} \theta(N_3) \\ N_3 = Nombre\ de\ TM3 \end{cases}$

TAB. 6.1 – **Complexités en temps de l'annotation par DACMs**.

6.3 Modules d'Exogean

Exogean utilise le modèle des modules paramétrés du langage OCAML [19]. Son code source est constitué d'environ 10000 lignes réparties en 25 fichiers, dont la moitié regroupe des fonctions d'entrée/sortie, et l'autre moitié les fonctions utiles à l'une des quatre étapes principales d'Exogean (figure 6.1). Le tableau 6.2 donne la répartition de ces fichiers selon les quatre étapes principales d'Exogean, ainsi qu'une brève description de la fonction de chacun et leur nombre de ligne.

Etape	**Fichier source**	**Description**	**Nb ligne fichier**	**Nb ligne étape**
Pré-traitement des HSPs	normal_mrna.ml pretraitement1.ml pretraitement2.ml	pré-traitement des HSPs ADNc rognage des HSPs protéiques fusion et élimination des HSPs protéiques	180 124 365	669
Annotation experte par DACM	graph.ml graph_aux.ml traiteCollection.ml	librairie de graphes construction de graphes réduction de graphe	767 197 1003	1967
Recherche du CDS	traiteSignal.ml sigsearch.ml matrice.ml orfsearch.ml	fonctions générales de recherche de signaux recherche de signaux recherche de signaux d'épissage par matrice de poids recherche de CDS	118 674 113 882	1787
Filtre transcrit/pseudo-transcrit	classifgene.ml	filtre transcrit/pseudo-transcrit	30	30
Total				4453

TAB. 6.2 – **Répartition des fichiers sources selon les étapes d'Exogean.**

6.4 Utilisation d'Exogean

Fichiers d'entrée d'Exogean. Exogean prend en entrée :

- le fichier de la séquence d'ADN à annoter,
- l'un et/ou l'autre des deux fichiers suivants :
  - le fichier du résultat d'un alignement local entre la séquence à annoter et des protéines d'une autre espèce (HSPs protéiques),

- le fichier du résultat d'un alignement local entre la séquence à annoter et des ADNc de la même espèce (HSPs d'ADNc).

De plus, si un fichier d'HSPs protéiques est spécifié, il est obligatoire de fournir le fichier des séquences de protéines qui ont servi à réaliser l'alignement correspondant. Notons que les fichiers d'HSPs mentionnés ci-dessus peuvent être dans l'un des trois formats suivants : psl, gff ou exf (spécifique à Exogean, voir annexe D).

Fichiers de sortie d'Exogean. Exogean produit en sortie six fichiers :

1. le fichier d'annotation des transcrits alternatifs des gènes de la séquence à annoter (ceux qui codent des protéines fonctionnelles) dans un format tabulaire (gtf ou bed),
2. le fichier des séquences de ces transcrits au format standard (fasta),
3. le fichier des séquences des traductions en acides aminés des CDS de ces transcrits (protéines fonctionnelles éventuellement partielles) au format standard (fasta),
4. le fichier de ces transcrits au format HTML afin de permettre une meilleure visualisation et navigation,
5. le fichier d'annotation des pseudogènes putatifs ou artéfacts de la séquence à annoter dans un format tabulaire (gtf ou bet),
6. le fichier des alignements d'ADNc après l'étape de pré-traitement (étape 1 d'Exogean, figure 6.1).

La figure 6.3 donne un schéma d'utilisation d'Exogean.

Autres paramètres d'Exogean. Exogean possède 37 paramètres qui concernent essentiellement les parties de filtrage des données, autrement dit l'étape 1 de pré-traitement des HSPs, et l'étape 4 de filtre transcrit/pseudo-transcrit (figure 6.1). A l'exception des fichiers de la séquence d'ADN, des HSPs d'ADNc et de protéines et des séquences protéiques, ces options possèdent des valeurs par défaut. Il est possible de les modifier et/ou de les spécifier de deux façons :

- via un fichier de configuration nommé `exogean.ini`, qui se situe dans le même repertoire que l'executable. Chaque ligne de ce fichier est de la forme :
  `nom_paramètre = valeur_paramètre`
- directement en ligne de commande, en faisant suivre l'executable d'une expression de la forme :
  `-nom_paramètre valeur_paramètre`
  Notons que le nom d'un paramètre n'est en général pas le même pour une utilisation par fichier de configuration et en ligne de commande (plus court dans ce dernier cas).

Pour une explication détaillée du sens et de la façon de spécifier les différents paramètres d'Exogean voir l'annexe D.

6.5 Le langage DACMLang

Le langage DACMLang permet d'automatiser la double action de mise en relation et de regroupement d'alignements de séquences effectuée par l'expert. Cette automatisation se fait par l'ajout de couleurs d'arêtes sur un DACM suivie de la réduction de ce dernier selon certaines couleurs d'arêtes. Ainsi une commande du langage DACMLang prend en entrée un DACM, le réduit selon certaines couleurs d'arêtes, et rend en sortie le DACM réduit sur lequel des couleurs d'arêtes ont été ajoutées. Un programme en DACMLang est une succession de telles commandes, et permet donc d'appliquer en cascade une succession de regroupements et de mise en relation de modèles de transcrits. En ce sens le développement de DACMLang peut être vu comme une manière de généraliser la partie centrale et essentielle d'Exogean : l'annotation experte par DACM.

Le langage DACMLang est le fruit d'une collaboration de trois ans entre un biologiste et deux informaticiens. Dans cette collaboration, le rôle du biologiste était d'énoncer la démarche et les règles utilisées par les annotateurs humains, celui des informaticiens de comprendre ces règles, de les formaliser et de les intégrer dans un environnement logiciel. Bien entendu de nombreux essais et erreurs ont été nécessaires pour éprouver le domaine complexe de l'annotation de gènes, et la synthèse de connaissances qui en est découlée est à l'origine du langage DACMLang.

Le statut actuel du langage DACMLang correspond à une implantation ad hoc de fonctions OCAML utiles à l'étape centrale d'annotation experte par DACM d'Exogean. Notre but est de réaliser un compilateur du langage DACMLang. Dans cette optique, l'analyse lexicale et syntaxique ont déjà été définies (fichiers ocamlLex et ocamlYacc), ainsi que la sémantique. Reste à réaliser la génération de code à partir des éléments de DACMLang. Une fois cette étape franchie, chaque annotateur en mesure de décrire son protocole d'annotation par une succession de mises en relation et de regroupements d'alignements de séquence, pourra écrire et tester son propre programme d'annotation. Cela nous permettra en retour d'améliorer le langage DACMLang.

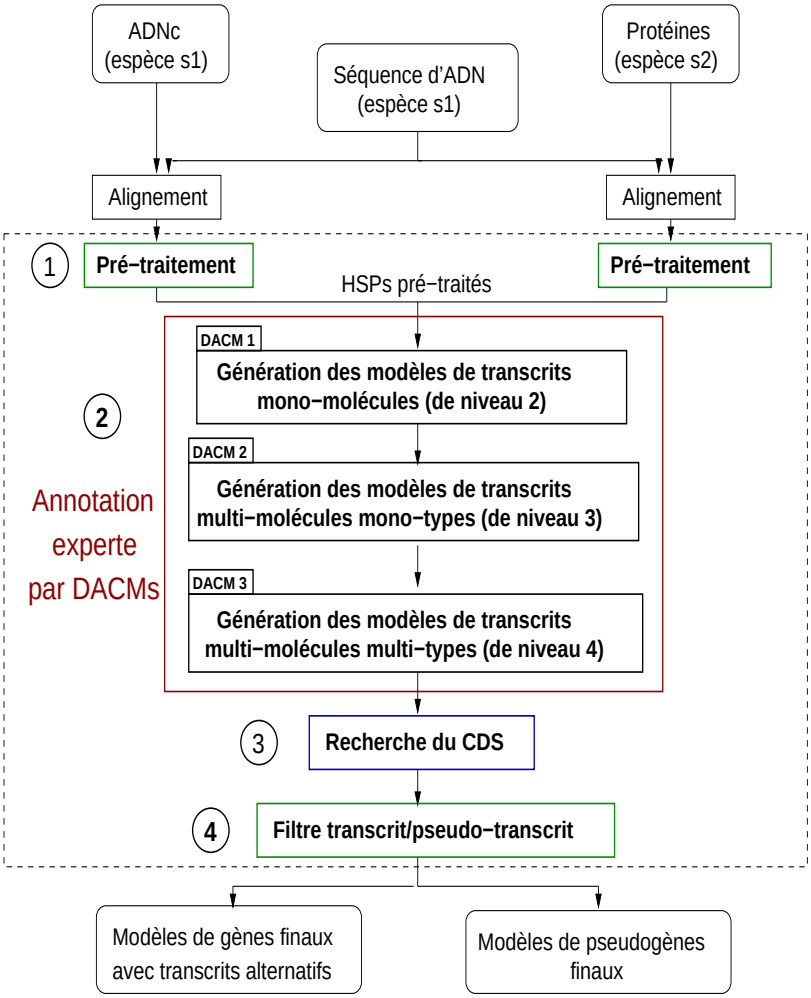


FIG. 6.1 – **Les différentes étapes du programme Exogean.** Exogean prend en entrée la séquence à annoter ainsi que le résultat (HSPs) de l'alignement entre cette séquence et d'une part des séquence d'ADNc de la même espèce, et d'autre part des séquences de protéines d'une autre espèce. Exogean se compose ensuite de quatre étapes : (1) le **pré-traitement** des HSPs d'ADNc et de protéines ; (2) l'**annotation experte par DACM** qui produit les structures des transcrits alternatifs des gènes de la séquence à annoter à partir des HSPs pré-traités. Cette étape se compose de trois sous-étapes dont chacune produit des modèles de transcrits de «plus grande complexité» par réduction d'un DACM ; (3) la **recherche du CDS** (CoDing Sequence) qui définit les bornes de la partie de chaque transcrit qui codera les acides aminés de la protéine correspondante ; (4) le **filtre transcrit/pseudo-transcrit** qui élimine les pseudogènes de ces structures.

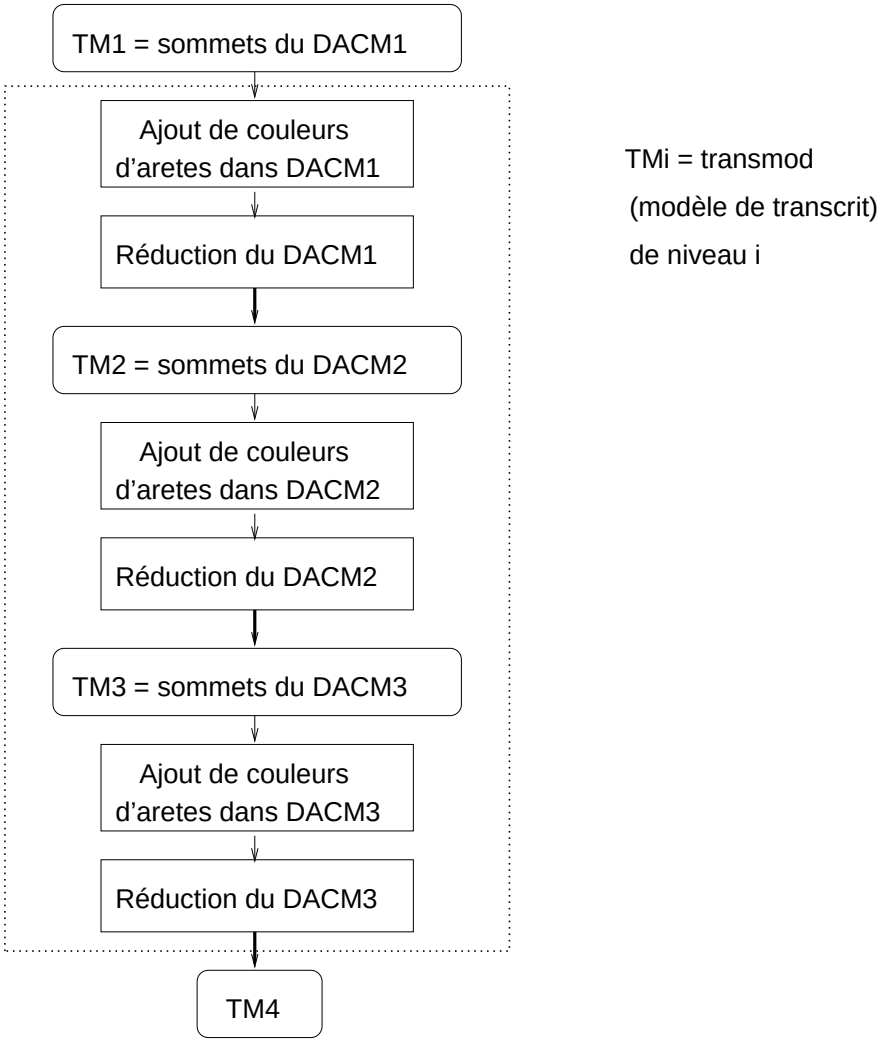


FIG. 6.2 – **Annotation experte par DACMs.**

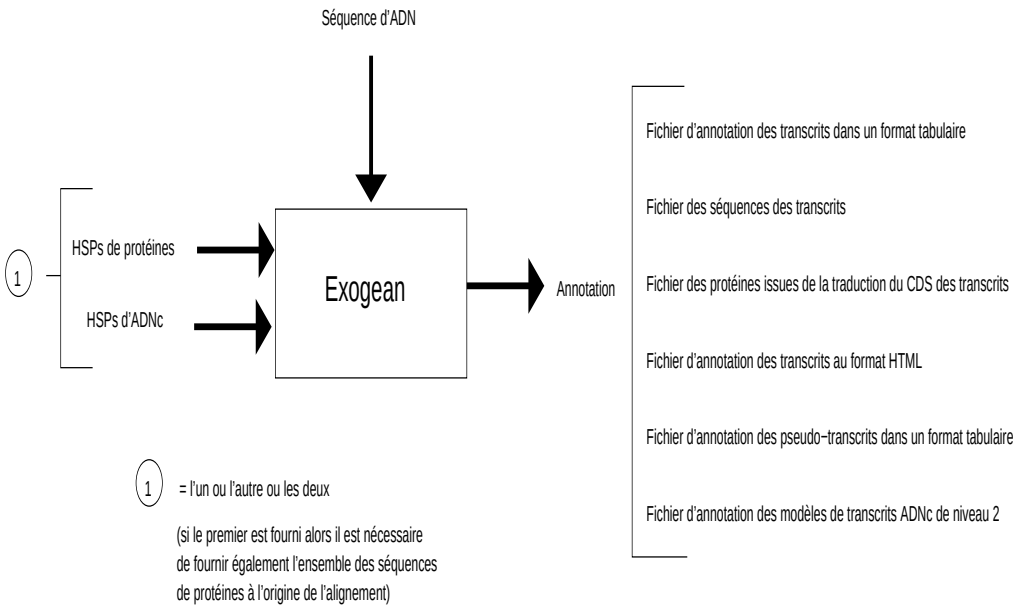


FIG. 6.3 – **Schéma d'utilisation d'Exogean.**

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Dans cette thèse, nous avons développé une méthode d'annotation de gènes eucaryotes qui modélise le protocole d'annotation de l'expert humain : le programme Exogean (EXpert On GEne ANnotation). Comme l'expert, Exogean applique des règles heuristiques sur des alignements de séquences relatifs à la séquence à annoter, et identifie ainsi les transcrits alternatifs codants des gènes qu'elle comporte. Exogean a été évalué lors du concours EGASP'05 et a été classé premier dans la tâche essentielle d'identifier au moins un CDS par gène de façon totalement exacte. De plus la spécificité en gènes d'Exogean (80%) a dépassé d'une large mesure celle des autres programmes (68% pour le deuxième). Nous pouvons donc affirmer que coder les règles heuristiques utilisées par l'expert humain est aujourd'hui la façon à la fois la plus efficace et la plus fiable d'annoter les gènes dans les génomes eucaryotes.

Etant donné que les règles heuristiques utilisées par l'expert changent au cours du temps, il est nécessaire de développer un cadre formel à la fois flexible, expressif et générique pour l'annotation de gènes. Pour répondre à ces besoins, nous avons conçu le langage DACMLang, dédié à l'annotation de gènes experte par mise en relation et regroupement d'alignements de séquences. La puissance de ce langage peut par exemple s'illustrer par le fait qu'il permet de coder la partie centrale et essentielle du programme Exogean par le programme suivant :

```
/* 1ere commande */
(même_molécule && taille_int_max && ordre_gx_molécule)+
:>
| 5'[Prot] >>- 3'[Prot] -> chevauchement
| (5'[Adnc] >>- 3'[Adnc]) &&
  (5'[Adnc] >< 3'[Adnc])  &&
  ((3'[Adnc]) >=* (5'[Adnc])) -> extension
| (5'[Adnc] >>- 3'[Adnc]) &&
  (5'[Adnc] >< 3'[Adnc])  &&
  ((3'[Adnc]) <* (5'[Adnc])) -> inclusion ;

/* 2eme commande */
[Prot]chevauchement+ || [Adnc]extension+
:>
| 5'[Adnc] >>- 3'[Prot] -> chevauchement
| 3'[Adnc] >>- 5'[Prot] -> chevauchement
| (5'[Prot]) >>= (3'[Adnc]) -> epissage_compatible
| (3'[Prot]) >>= (5'[Adnc]) -> epissage_compatible ;

/* 3eme commande */
(chevauchement && epissage_compatible) || [Adnc]
:>
;
```

D'autre part, le concours EGASP'05 a montré que les stratégies heuristiques (Exogean) donnaient de meilleurs résultats que les modèles purement numériques (modèles de Markov de Jigsaw et de Pairagon). Nous pouvons donc nous poser la question suivante : les modèles purement numériques ont-ils atteint leurs limites ? En fait, les modèles numériques actuels sont le plus souvent des combinateurs de ressources (alignements de séquences, résultats de programmes *ab initio*, ...etc), qui elles-mêmes augmentent en quantité et en qualité avec le temps. Ainsi, les modèles numériques ont sans doute encore un potentiel : celui des ressources qu'ils combinent. Mais qu'en est-il du potentiel de la modélisation du gène sous-jacente ? Comment peut-on être sûr de modéliser correctement ce dont on sait si peu de choses ? Quoiqu'il en soit, les modèles numériques possèdent un défaut conceptuel majeur : leur hermétisme. En effet ces modèles sont fondés sur un apprentissage des données, et empêchent toute communication avec le biologiste : celui-ci ne peut ni leur incorporer de connaissances, ni interpréter les résultats qu'ils produisent (valeur de confiance sous forme d'un chiffre, comment les résultats ont-ils été obtenus ?).

Plus généralement, étant donné qu'aucun programme d'annotation de gène actuel ne parvient à faire aussi bien que l'expert humain, nous pouvons nous poser la question suivante :

est-il possible qu'un jour, un programme d'annotation annote les gènes aussi bien, voire mieux, que l'expert humain ? Peut-être pas sans une étroite collaboration avec les experts, en leur donnant les moyens d'exprimer leur expertise aisément, et surtout pas sans une réflexion poussée pour mettre en place un cadre formel le plus ouvert possible sur l'expertise à venir. Si cette stratégie permet un jour de faire aussi bien que l'humain, elle fera en réalité mieux que lui, car ce sera sans erreur et sans variabilité.

7.2 Perspectives

7.2.1 Intégrer d'autres ressources

Les ESTs de la même ou d'une autre espèce. Pour annoter les gènes d'une séquence d'ADN, les annotateurs utilisent des alignements d'ADNc de type EST d'une espèce proche de celle de la séquence à annoter, le plus souvent de la même espèce. Ces alignements représentent des annotations plus partielles que les alignements d'ADNc pleine longueur, mais peuvent parfois être complémentaires de ces dernières. En effet les ESTs sont beaucoup plus facile à obtenir que les ADNc pleine longueur, et représentent donc un type de données particulièrement abondant et redondant. Même si les séquences d'ESTs sont des données de moins bonne qualité que les séquences d'ADNc pleine longueur (contaminations génomiques, mauvais étiquetage, ..etc), ils permettent parfois d'identifier des ARNm alternatifs non représentés dans les données d'ADNc pleine longueur.

L'ADN génomique d'une autre espèce. Si l'on annote la séquence ADN génomique d'une espèce sur laquelle peu de protéines d'autres espèces d'alignent, il peut être utile d'utiliser des alignements avec des séquences d'ADN génomique d'une autre espèce. En effet l'ADN génomique d'une espèce est en général plus complet que l'ensemble des protéines annotées de cette espèce. Il faudra cependant prendre en compte le fait que les alignements ainsi obtenus peuvent se situer dans des introns ou des régions intergéniques.

Les ADNc d'une autre espèce. Les protéines contenues dans les bases de données publiques sont en général obtenues par traduction du CDS d'une séquence d'ADNc, si celui-ci est annoté. Si une séquence d'ADNc est soumise sans aucun CDS annoté, il est probable que la protéine correspondant à cet ADNc ne soit pas disponible. Il peut donc être utile d'utiliser les ADNc d'une espèce donnée en complément des protéines de celle-ci. Cependant, il faut noter que contrairement aux ADNc de la même espèce que la séquence à annoter, les alignements d'ADNc d'une autre espèce ne donnent pas nécessairement les bornes d'épissage exactes.

7.2.2 Intégrer d'autres règles heuristiques

L'analyse du protocole d'annotation de l'expert nous a permi d'identifier deux types de règles :

- les règles biologiquement fondées, qui se réfèrent à une connaissance précise en biologie, très souvent une publication ayant identifié un mécanisme biologique,
- les règles purement empiriques, dont les experts savent qu'elles fonctionnent bien en pratique mais dont on ne connaît pas la raison précise.

En respectant cette classification, nous donnons ici des exemples de règles heuristiques qui n'ont pas encore pu être intégrées dans Exogean ou DACMLang.

Règles biologiquement fondées. Une cause importante de prédiction de CDS erronée est le choix de la méthionine initiale. En effet il arrive fréquemment que pour un transcrit donné, Exogean prédise un CDS dans la même phase que celui annoté manuellement, et de même fin et bornes d'épissage, mais avec une méthionine initiale différente. De plus la méthionine prédite par Exogean est toujours à une faible distance de celle annotée manuellement, et cette distance est toujours multiple de 3. Une manière de résoudre ce problème est peut-être d'utiliser une matrice de poids pour pondérer les méthionines initiales potentielles d'un CDS. En effet **Kozak** a montré qu'il existe un signal consensus autour de la méthionine initiale des CDS eucaryotes. Une séquence consensus englobant l'ATG initial serait GCCGCC[AouT]CC**ATG**G [94], et pourrait être généralisé par une matrice de poids obtenue sur des CDS eucaryotes connus.

Le **NMD** (Nonsense-mediated mRNA decay) est un mécanisme eucaryote de surveillance des ARNm. Ce mécanisme consiste en la détection et la dégradation des ARNm contenant des codons Stop prématures, ce qui a pour conséquence qu'aucune protéine ne résultera de cet ARNm. Chez les mammifères, un codon Stop est considéré comme prématuré s'il se situe à plus de 50 nucléotides en amont de la position de début du dernier intron de l'ARNm. Chez l'homme il a été montré que plus d'un tiers des ARNm alternatifs subiraient ce mécanisme [81]. Les ADNc séquencés peuvent contenir de telles formes, toutefois il n'est pas souhaitable d'annoter les transcrits correspondants comme codants puisqu'ils ne donnent pas lieu à des protéines. Même si ce mécanisme a été découvert depuis peu, les annotateurs humains le prennent aujourd'hui en compte, et il serait souhaitable qu'Exogean fasse de même. Cela aurait lieu après la recherche du CDS, dans l'étape de filtrage des pseudo-transcrits.

Règles purement empiriques Chez les eucaryotes, un gène peut donner lieu à plusieurs transcrits alternatifs, dont la longueur (somme des longueurs des exons codants et non codants), et le nombre d'exons varient. Annoter un gène signifie donc localiser les différents transcrits alternatifs de ce gène. Pour ce faire l'expert humain regroupe les ressources relatives à chacun d'entre eux puis annotent leurs CDS. Or il semblerait que cette dernière tâche soit d'abord exécutée pour le transcrit le plus long, puis pour le deuxième transcrit le plus long,...etc. De plus les experts humains semblent utiliser la règle suivante : «si deux transcrits codants t_1 et t_2 d'un même gène sont tels qu'il existe un exon e_{1i} de t_1 qui chevauche un exon e_{2j} de t_2, alors e_{1i} et e_{2j} sont dans la même phase». Ainsi, l'annotation du CDS d'un transcrit dépend de son ordre d'inspection. Autrement dit, plus un transcrit est annoté de façon tardive au sein d'un gène, plus il a de chance d'être considéré comme non codant car

plus il a de chance de ne pas suivre cette règle de phase (les CDS des transcrits déjà annotés ne sont en effet pas remis en question). Nous souhaiterions tester cette règle dans Exogean.

Les résultats d'Exogean au concours EGASP'05 montrent qu'Exogean prédit mieux de façon exacte les exons internes que les exons initiaux et terminaux des CDS des transcrits (voir chapitre 5). Nous avons donc analysé plus précisément les CDS d'Exogean dont les bornes internes coïncident avec les bornes internes d'un CDS annoté manuellement mais dont les bornes externes diffèrent. Cela nous a permi de constater que les bornes externes des CDS annotés manuellement sont parfois *arbitraires*. En effet les annotateurs humains ne peuvent annoter une base d'un transcrit que si une ressource expérimentale est présente à cet endroit. De plus, il arrive fréquemment qu'un CDS d'Exogean étende un CDS annoté manuellement en 5' et/ou en 3', tout en étant dans la même phase. Tout cela montre que la recherche du CDS est problématique. Actuellement dans Exogean la sélection du CDS d'un transcrit parmi plusieurs se fait uniquement sur la base de la longueur du CDS : parmi plusieurs CDS potentiels d'un même transcrit, Exogean élit le plus long, c'est-à-dire celui dont la somme des longueurs des exons est maximale. Nous pourrions tester *d'autres critères* de sélection du CDS, comme par exemple les suivants, ou encore toute combinaison de ceux-ci :

- le score de la méthionine initiale obtenue par matrice de poids du signal kozak (voir plus haut),
- le nombre d'exons du CDS,
- le rapport entre la somme des longueurs des exons du CDS et la somme des longueurs des exons du transcrit,
- le rapport entre le nombre d'exons du CDS et le nombre d'exons du transcrit,
- le type du CDS (complet, incomplet en 5', incomplet en 3', incomplet en 5' et en 3').

Notons que les fonctions de recherche de CDS actuellement implantées permettent de tester l'ensemble de ces critères tout à fait aisément.

7.2.3 Vers une expertise de l'expertise ?

DACMLang est un langage dédié au domaine de l'annotation experte des gènes eucaryotes. Pour identifier les structures de transcrits d'une séquence, l'expert utilise un ensemble de règles heuristiques de deux types (voir plus haut) :

- des règles biologiquement fondées,
- des règles purement empiriques.

Ces dernières ne sont pas fondées sur une connaissance précise, et n'ont donc *pas d'existence réelle*. Puisque DACMLang permet à l'expert d'exprimer les règles qu'il utilise de manière formelle et non ambigüe, nous pensons qu'il est un moyen de transfert des règles purement empiriques vers le domaine des règles biologiquement fondées. Etant donné que le domaine de l'annotation des gènes reste un domaine de la «tradition orale», nous pensons qu'un tel transfert représenterait une avancée considérable.

Annexe A

Les ADN complémentaires (ADNc)

Dans les génomes d'eucaryotes supérieurs, seule une faible fraction de l'ADN génomique code pour des protéines. Pour connaître l'ensemble des gènes d'un organisme sans pour autant avoir à séquencer tout son génome, un moyen consiste à obtenir les séquences des ARNm exprimés dans ses cellules. Mais les ARNm sont des molécules fragiles, que l'on ne peut séquencer directement. Pour cette raison on obtient leur séquence par l'intermédiaire d'un ADN dit complémentaire (ADNc), qui est la copie d'un ARNm mature (donc dépourvu d'introns).

Pour obtenir des séquences d'ADNc à partir d'ARNm matures, deux étapes sont nécessaires :

1. l'obtention des molécules d'ADNc, qui nécessite les trois sous-étapes suivantes :
   - l'extraction des ARNm matures des cellules,
   - la synthèse d'ADNc à partir des ARNm matures,
   - le clonage des ADNc dans des plasmides puis l'insertion des plasmides dans *Escherichia coli* pour obtenir une grande quantité d'ADNc.
2. le séquençage des molécules d'ADNc.

A.1 Obtention des molécules d'ADNc

Extraction des ARNm matures des cellules. A partir d'un tissu ou de cellules en culture, les ARNm matures sont isolées et purifiées. Les ARNm portent en effet une queue poly-A en 3', et peuvent donc être purifiés par chromatographie sur des résines où l'on a greffé chimiquement du poly-T [22].

Synthèse d'ADNc à partir d'ARNm matures. Les ARNm matures purifiés sont incubés avec une transcriptase inverse, qui comme son nom l'indique permet la transcription inverse des ARNm en ADN (les ARNm sont alors dégradés par l'ajout d'une enzyme).

L'ADN polymérase est ajouté et permet de synthétiser le deuxième brin d'ADN à partir du premier. Un ADN double brin *complémentaire* de l'ARNm mature initial est ainsi obtenu, d'où le terme d'ADN complémentaire ou ADNc (voir figure A.1).

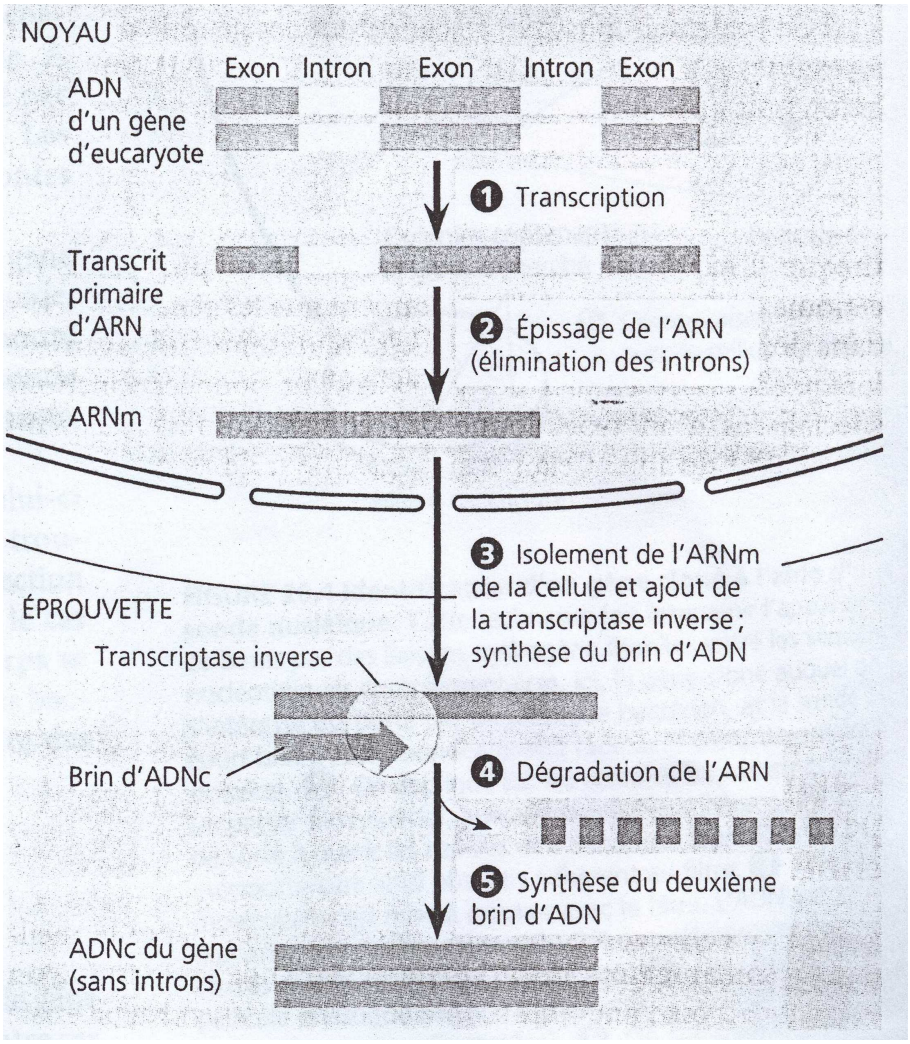


FIG. A.1 – **Production d'un ADN complémentaire (ADNc) d'un gène eucaryote.** On fabrique l'ADN complémentaire *in vitro* à partir d'une matrice d'ARNm et à l'aide de la transcriptase inverse. Les étapes 1 et 2 montrent la synthèse et l'épissage d'une molécule d'ARNm dans le noyau d'une cellule. A l'étape 3 on isole les molécules d'ARNm de la cellule et on leur ajoute la transcriptase inverse ; celle-ci fabrique un brin d'ADN sur la matrice formée par l'ARN. A l'étape 4, on ajoute une autre enzyme qui dégrade l'ARN puis, à l'étape 5, on fournit de l'ADN polymérase, qui synthétise un deuxième brin d'ADN. On obtient ainsi de l'ADNc portant sur la séquence codante complète du gène, mais aucun intron. Cette figure est extraite de [23].

Clonage des ADNc puis insertion dans *E. coli*. Les ADNc synthétisés sont insérés dans des vecteurs bactériens (plasmides), qui sont eux-mêmes introduits dans des bactéries *Escherichia coli*. En effet on souhaite effectuer un séquençage systématique et intensif (même si partiel) des ADNc, et il faut donc d'abord obtenir une grande quantité de ces derniers.

E. coli possède la capacité de se multiplier très facilement, par conséquent si un plasmide contenant un ADNc est introduit, elle donnera lieu par multiplication à une grande quantité de ce plasmide, et donc de l'ADNc.

Les plasmides dans lesquels un ADNc est inséré sont dits *recombinés*, et la position du plasmide à laquelle l'ADNc est inséré est appelée *site de clonage*.

A.2 Séquençage des molécules d'ADNc

Le séquençage des ADNc se fait à partir des plasmides recombinés et multipliés. Il s'effectue à partir d'une amorce correspondant à une séquence du plasmide située au niveau du site de clonage. Si le séquençage de l'ADNc n'est que partiel, dans la direction 5' ou 3' à partir du site de clonage, la séquence obtenue est appelée *EST pour Expressed Sequence Tag*, 5' ou 3'. Dans les cas où l'on sait de façon *certaine* que la séquence obtenue correspond à l'ARNm mature initial complet, la séquence obtenue est dite *ADNc pleine longueur*. Cependant cette terminologie est parfois aussi utilisée pour décrire les cas où la totalité de la séquence d'ADNc est insérée dans le plasmide, même si celle-ci ne correspond pas à un ARNm complet.

Annexe B

Phylogénie et relations d'homologie

La théorie de l'évolution selon Darwin suggère que toutes les espèces possèdent un *ancêtre commun* [14]. Cette relation, appelée *phylogénie*, est traditionnellement représentée par un arbre, appelé *arbre phylogénétique*, où les noeuds sont des taxons (unité formelle représentée par un groupe d'organismes, à chaque niveau de la classification) et où les branches définissent les relations entre les taxons en terme de descendants et d'ancêtres. La longueur d'une branche de l'arbre représente généralement le nombre de changements qui ont eu lieu dans cette branche (divergence). Les taxons représentés par les noeuds de l'arbre phylogénétique peuvent aussi bien être des espèces, des populations, des individus ou des gènes [17]. Un exemple d'arbre phylogénétique est donné en figure B.1.

Pour construire des arbres phylogénétiques, les phylogénéticiens ont d'abord comparé les organismes du point de vue morphologique et physiologique, et ont ainsi pu déterminer les aspects majeurs de l'histoire évolutive des organismes. Cependant les changements évolutifs des caractères morphologiques et physiologiques sont extrêmement complexes. De ce fait cette approche est incapable de donner une image précise de l'histoire évolutive des organismes, et les détails des arbres phylogénétiques ainsi recontruits ont très souvent donné lieu à controverse. Les progrès en biologie moléculaire ont apporté une solution à ce problème. En effet puisque l'ADN contient l'information sur les caractères héréditaires d'un être vivant, les relations d'évolution entre les organismes peuvent être étudiées par comparaison de leur ADN [20].

L'évolution est la combinaison de deux processus : d'une part les mutations, et d'autre part la dérive (évolution neutre) et/ou la sélection naturelle. Un gène ou une séquence d'ADN ayant subi une mutation (substitution de nucléotide, insertion, délétion, duplication, transposition, ...etc) peut se transmettre à toute une population par dérive et/ou sélection naturelle, et ainsi se fixer au sein d'une espèce [20]. Les espèces divergent et une espèce donnée peut donner lieu à une ou plusieurs espèces différentes par un processus dit de *spéciation*. L'ensemble des gènes de deux espèces partagent ainsi des relations d'homologie, dont les plus importantes sont les relations *d'orthologie* et *de paralogie*, précisées par les définitions suivantes :

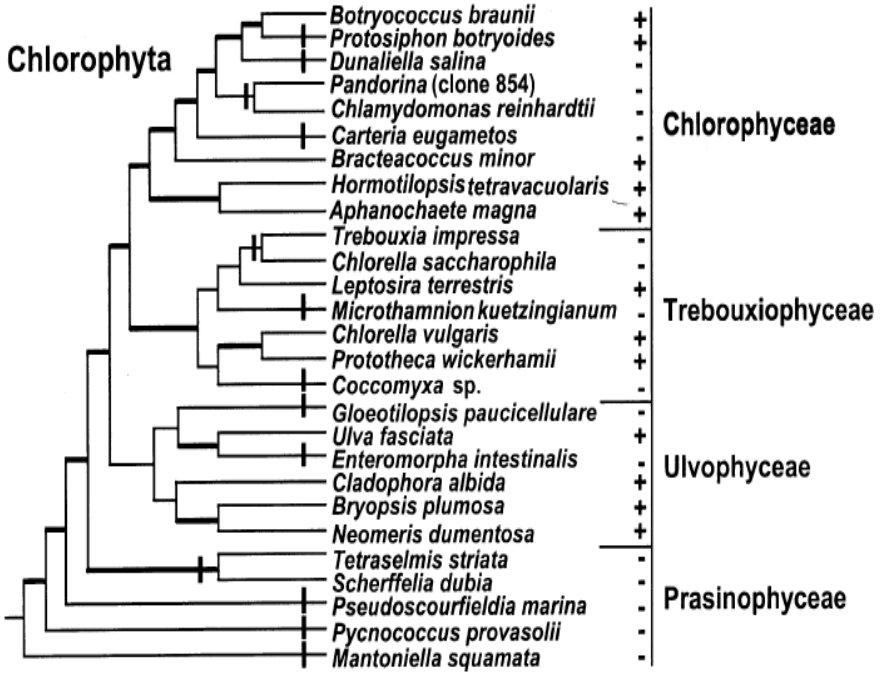


FIG. B.1 – **Arbre phylogénétique.** Ce schéma décrit l'arbre phylogénétique des algues chlorophytes, extrait de [55].

- gènes orthologues : se dit de deux gènes appartenant à deux espèces différentes et issus directement du gène appartenant au plus proche ancêtre commun (figure B.2),
- gènes paralogues : se dit de deux gènes appartenant à la même espèce et issus d'un même gène ancestral par duplication (figure B.2).

Par extension, des protéines issues de gènes orthologues sont dites orthologues, et des protéines issues de gènes paralogues sont dites paralogues.

Comparer une séquence d'ADN génomique contenant un gène donnant lieu à une protéine p, avec l'ensemble des protéines d'une autre espèce contenant une protéine orthologue à p, permettra donc de localiser les exons de la protéine p sur l'ADN génomique. Il est également possible de comparer directement l'ADN génomique de deux espèces différentes, directement en nucléotide ou après avoir traduit l'ADN dans les 6 phases. Ces deux stratégies sont les fondements de la *génomique comparative*. Notons que les séquences de protéines contenues dans les bases de données publiques, sont le plus souvent obtenues par traduction du CDS d'un ADNc, lorsque celui-ci est annoté.

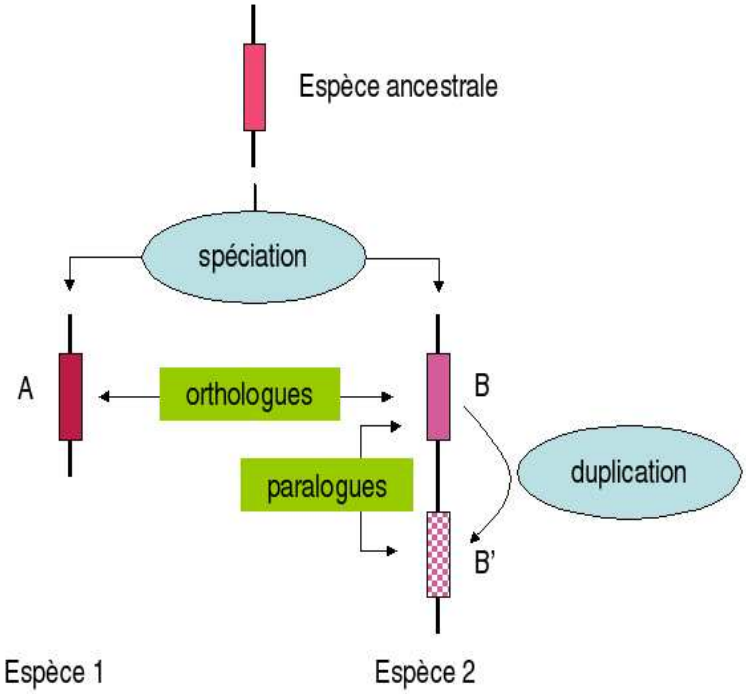


FIG. B.2 – **Gènes orthologues et gènes paralogues.**

Annexe C

Fonctionnement du programme Blast

Pour identifier les gènes d'une séquence d'ADN, les méthodes dites par similarité de séquence recherchent des homologies entre cette séquence et d'autres séquences biologiques telle que :

1. des protéines de la même espèce,
2. des protéines d'une autre espèce,
3. des ADNc de la même espèce,
4. des ADNc d'une autre espèce,
5. de l'ADN génomique d'une autre espèce,
6. des répétitions/transposons.

Pour trouver des homologies entre deux séquences il suffit de réaliser un *alignement* de ces deux séquences, autrement dit de trouver une suite de transformations élémentaires permettant de passer de l'une à l'autre. Les opérations élémentaires dont il est question peuvent être des trois types suivants :

- la substitution d'un résidu par un autre,
- l'insertion d'un résidu,
- la délétion d'un résidu.

Notons que ces deux dernières opérations sont regroupées sous le terme *indel*. Certains algorithmes d'alignements tolèrent les indels, d'autres non.

D'autre part un alignement entre deux séquences peut être global c'est-à-dire porter sur la totalité des deux séquences à aligner, ou encore local c'est-à-dire porter sur des sous-séquences des deux séquences à aligner. Etant donnés une matrice de score de substitution de résidus, et un modèle de pénalité de gap, le problème de l'alignement avec gap entre deux séquences est résolu de manière exacte par les algorithmes de Needleman-Wunch [107] et de Smith-Waterman [128] respectivement. Pour aligner des séquences de tailles n et m respectivement, ces deux algorithmes utilisent une matrice de taille $(n+1)\times(m+1)$ à laquelle ils appliquent une méthode de programmation dynamique. Ainsi, la complexité en temps de ces algorithmes est en $\theta(n\times m)$ et ne convient pas pour les comparaisons intensives de séquences

de très grande taille.

Des algorithmes heuristiques d'alignement ont été proposés qui n'appliquent la programmation dynamique qu'à une fraction de la matrice précédemment décrite. Ainsi ces algorithmes ont une sensibilité un peu moins élevée que les algorithmes exacts d'alignement, mais ils permettent un gain de temps drastique et sont de ce fait les plus utilisés aujourd'hui. Ces heuristiques produisent des alignements locaux de deux séquences qui représentent des approximations des alignements locaux réels de ces deux séquences. Le principe général des heuristiques d'alignement est que des alignements biologiquement valides entre deux séquences, doivent très certainement comporter des sous-séquences identiques ou très similaires entre les deux séquences. Il suffit donc de déterminer ces sous-séquences très similaires dans les séquences à aligner, puis d'essayer *d'étendre* ces sous-séquences de part et d'autre des séquences à aligner.

Aujourd'hui, de nombreuses heuristiques d'alignement existent, cependant la plus utilisée demeure Blast [29]. Blast permet d'aligner une séquence requête avec plusieurs séquences d'un base de données. Selon le type des séquences requête et de la base, un type de Blast différent est utilisé (cf tableau C.1).

Type de Blast	Type de la séquence requête	Type des séquences de la base se données	Mode
BlastN	Nucléique	Nucléique	Nucléique
BlastP	Protéique	Protéique	Protéique
BlastX	Nucléique traduit[1]	Protéique	Protéique
TBlastN	Protéique	Nucléique traduit	Protéique
TBlastX	Nucléique traduit	Nucléique traduit	Protéique

TAB. C.1 – **Différents types de Blast.**

Pour attribuer un score aux alignements qu'il produit, Blast utilise une matrice de score de substitution de résidu. Cette matrice est nucléique de taille 4×4 pour une utilisation de Blast en mode nucléique, et protéique de taille 20×20 pour une utilisation de Blast en mode protéique. Un alignement de deux séquences peut en effet être vu comme une suite de paires de résidus alignés. Une paire de résidus alignés identiques est appelée *match*, et une paire de résidus alignés différents est appelée *mismatch*. Le score d'une paire de résidus alignés est donné par la matrice de score de substitution de résidu, et est bien entendu plus élevé pour un match que pour un mismatch. Le score d'un alignement s'obtient ainsi en faisant la somme des scores de chaque paire de résidu. Blast utilise aussi sept paramètres numériques, décrits dans le tableau C.2.

Parmamètre	Description	Valeur par défaut en mode protéique	Valeur par défaut en mode nucléique
W	longueur des sous-séquences d'ancrage de l'alignement	3	11
T	score de similarité minimum pour la génération de sous-séquences d'ancrage	évaluation dynamique pour chaque mot	non applicable
X	valeur maximale de la valeur absolue du cumul des scores de plusieurs mismatches consécutifs lors de l'extension des sous-séquences d'ancrage	20	20
S	score minimal primaire d'un alignement	1	20
S_2	score minimal secondaire d'un alignement	0	20
E	valeur attendue primaire maximale	10	10
E_2	valeur attendue secondaire maximale	0.5	0.05

TAB. C.2 – **Paramètres de Blast.**

Comme les autres heuristiques d'alignement local, Blast procède en deux étapes pour aligner la séquence requête R avec une séquence B de la base de données :

1. identification dans R et B des paires de sous-séquences d'ancrage d'alignement (matrice de score de substitution et paramètres W et T), en deux étapes :
   (a) génération de sous-séquences d'ancrage potentielles de R de taille W,
   (b) identification des matches exacts entre les sous-séquences d'ancrage potentielles de R de taille W et les sous-séquences de B de taille W,

   De plus la génération de sous-séquences potentielles de R de taille W se fait différemment selon que le mode de Blast utilisé :
   – en mode nucléique, seuls les mots de R de taille W sont générés,
   – en mode protéique, tous les mots de taille W dont le score d'alignement avec un mot de R de taille W est supérieur à T sont générés.

2. extension des paires de sous-séquences d'ancrage d'alignement sur R et B et dans les deux directions (matrice de score de substitution et paramètre X). L'extension d'un

alignement dans une direction donnée est stoppée quand la valeur absolue du cumul des scores de plusieurs mismatches consécutifs devient supérieure à X.

Les alignements de R et B ainsi obtenus sont ensuite filtrés pour obtenir les meilleurs alignements ou HSP (High-scoring Segment Pair) (paramètres S, S_2, E, E_2).

Une version de Blast qui tolère les indels existe : Blast2 [30]. Les exécutables des programmes Blast et Blast2 peuvent être téléchargés [9], et ainsi utilisés en ligne de commande. Blast et Blast2 peuvent également être utilisés en ligne, par exemple sur le site du NCBI [8].

Annexe D

Utilisation d'Exogean

D.1 Le format exf

Le format exf est un format de fichier d'HSPs spécifique à Exogean dans lequel chaque ligne correspond à un HSP et est constitué de 8 champs (séparés par des espaces) qui sont des attributs de l'HSP :

1. Nom de la séquence d'ADN cible,
2. Nom de la séquence source (ADNc ou protéine) de l'HSP,
3. Score de l'HSP,
4. Début de l'HSP sur la séquence d'ADN,
5. Fin de l'HSP sur la séquence d'ADN,
6. Début de l'HSP sur la séquence source,
7. Fin de l'HSP sur la séquence source,
8. Brin de la séquence d'ADN sur lequel l'HSP se situe.

D.2 Les paramètres d'Exogean

D.2.1 Via le fichier de configuration `exogean.ini`

```
# ##############################################################################
#                                                                              #
# exogean.ini : A file to specify option values for the exogean program        #
#                                                                              #
# This file should be placed in the directory where the exogean program is     #
# executed. Then simply type "exogean" on the command line                     #
#                                                                              #
# All additional arguments specified on the command line will overide those    #
# specified in this file.                                                      #
```

```
# Type exogean -h for a more detailed description of the options.            #
# Comments can be indicated with a "#" at the beginning of the line          #
# The syntax is: [option name][some spaces]=[some spaces][option value]      #
# for boolean values, use 'true' or 'false'                                  #
#                                                                            #
# Each comment line below shortly describes an option, and ends with the     #
# name of the corresponding option name that would overide it on the command #
# line version.                                                              #
#                                                                            #
# NOTE: this version of the exogean.ini file has been pre-filled with        #
# default values.                                                            #
#                                                                            #
##############################################################################


######################
# Options for files #
######################

# The fasta file with the genomic DNA to annotate (-gseq)
# (mandatory)
gseqfile = mygenomicDNA.fa

# The format:filename with protein-DNA alignments (-hspp)
# (optional)
hsppfile = psl:my_prot_alignments.psl

# The fasta file with protein sequences used for the alignment (-bqp)
# (mandatory if hsppfile option used, optional otherwise)
bqpfile = myproteins.fa

# The format:file containing mRNA alignments (-hspa)
# (optional)
hspafile = psl:my_rna_alignments.psl

# The format:basename for the output files (-o)
outfile = gtf:my_out_files

# optional redirection. Allowed values are prot or cdna or gene
# or pseudo (-pipe)
pipe = prot
```

```
# offset to generate predictions on a different genome coordinate
# system (-offset)
offset = 0

# verbose mode (-v)
verbose = false



########################################
# Options for pre-processing alignments #
########################################

# use proteins that form a single HSP on the genomic DNA.
# default: not used (-pmono)
pmono = false

# tolerance in bp for fusing adjacent HSPs when using the protein as guide (-f)
fusionthresh = 30

# Maximal size of introns in bp for transcripts predicted by exogean (-I)
tmaxintron = 75000

# search depth (in codons) for splice sites flanking protein HSPs (-naa)
nbaamanq = 10

# search depth (in codons) for splice sites inside protein HSPs (-ni)
nninhsp = 8

# tolerance (in codons) for extending splice search depth(-ns)
nnsuppl = 15

# Minimal intron size (bp) for predicted transcripts based on protein
# alignments (-i)
tminintron = 60

# The fasta files containing 10 bp donor and acceptor sites (-mag ; -mgt)
magfile = sig_AG.fa
mgtfile = sig_GT.fa


# minimal percentage of mono or di-HSP alignment size versus total
```

```
# protein size (-pm)
propmol = 33




################################################
# Options for post-processing of predicted CDSs #
################################################

# minimal size of predicted CDS with no starting methionine (-scnmint)
szcdsnomet = 300

# minimal size of predicted CDS with starting methionine (-scmint)
szcdswithmet = 210

# minimal % of monoexonic CDS size versus total transcript size (-pcdsint)
propcdsmonoex  = 20

# Minimal size of predicted monoexonic CDS (-scdsmint)
sizecdsmonoex = 420

# Minimal size of predicted bi-exonic CDS (-scdsdint)
sizecdsdiex = 210
```

D.2.2 En ligne de commande

```
exogean -gseq DNA_file.fa  [options]
```

As an alternative to options being specified on the command line, all the arguments given to the program can be recapitulated in a file called exogean.ini that must be in the directory where exogean is executed. In this case, simply execute exogean with no arguments. If arguments are nevertheless provided on the command line, they override the corresponding ones in the exogean.ini file.

Exogean annotates protein-coding genes in a genomic DNA sequence based on information provided by alignments of protein and/or mRNA sequences to this DNA. The only mandatory argument is the DNA sequence. Obviously if nothing else is provided, Exogean will not annotate any genes and will not produce any files. Alignments are provided to exogean in separate files for mRNAs (-hspa) and proteins (-hspp). If a protein alignment file has been provided, Exogean also requires the file of protein sequences that were used to compute the alignments, with the option -bqp.

Exogean uses rules extracted from human expertise to process the alignments and combine this information to annotate protein coding genes in DNA. Exogean is able to identify

several transcripts per genes if mRNA alignments are provided, but not if only protein alignments were given. Depending on the evidence given, Exogean may predict no genes, genes and/or pseudogenes, genes with overlapping boundaries, genes nested in introns of other genes on the same strand, etc.

Briefly, Exogean will process the information in 4 stages : pre-processing, DACM algorithm, CDS identification and post-processing. Options below allow the user to modulate the pre- and post-processing stages, while rules from the DACM algorithm and for the CDS identification are currently hard coded. A new language is being developed to let the user modify all the rules and create new ones.

The output of Exogean consists in six files :

1. The gene annotation file, in either gtf or bed format. The extension is *.gene.gtf or *.gene.bed.
   For a description of these formats please see http ://genes.cs.wustl.edu/GTF2.html or http ://genome.ucsc.edu/goldenPath/help/customTrack.html#BED
2. The cDNA sequence of each annotated transcript, in fasta format. The extension is *.cdna.fa ;
3. The translated CDS sequences of each annotated transcript, in fasta format. The extension is *.cds.fa ;
4. The gene annotations in html format. The extension is *.gene.html. This file can be uploaded in a browser to allow for an easier navigation in the gene annotations and their respective transcripts ;
5. Putative pseudogenes and/or artifacts in the same format as the gene annotation file. The extension is either *.pseudo.bed or *.pseudo.gtf ;
6. A list of mRNA alignments that passed the initial filters, before entering the DACM algorithm. The file is empty if no mRNA alignments were provided. This file can generally be ignored, but can be useful to understand why some genes where poorly annotated even if mRNA sequences for these genes were provided.

```
List of file arguments:
-----------------------

-gseq string          string=DNA sequence fasta file (mandatory).
                      Currently a single DNA sequence per file
                      is supported.

-hspp format:file     The file containing the protein alignments
                      (optional). The alignments can be provided
                      in one of three formats that can be specified
                      before the semicolons: psl, gff or exf. If
                      only the file name is indicated, the format
                      is assumed to be PSL. The exf format is a
```

```
                    simple format designed for Exogean and consists
                    in 8 fields separated by spaces, tabs or a
                    combination of spaces and tabs:
                    (1) DNA sequence name,
                    (2) protein/mRNA name,
                    (3) a score for the alignment,
                    (4) start in genomic DNA,
                    (5) end in genomic DNA,
                    (6) start in molecule,
                    (7) end in molecule,
                    (8) a plus or minus sign for the strand.
                    When the strand is negative, then the start
                    and end of the alignment are given relative
                    to the plus strand. The PSL format is typically
                    produced by BLAT alignments and is described here:
                    http://genome.ucsc.edu/goldenPath/help/customTrack.html#PSL.
                    A description of the Generic Feature Format (GFF)
                    can be found here:
                    http://www.sanger.ac.uk/Software/formats/GFF/.
                    If a protein alignment file is provided, then a
                    protein fasta file must also be provided with the
                    -bqp argument (see below).

-bqp string       string = protein sequence fasta file (mandatory
                    if the -hspp option is used). Typically this is
                    the same file that was used to compute the
                    alignments with the genomic DNA, although here
                    the protein sequences are preferably unmasked.
                    The protein names in the alignment file and in
                    the fasta file must be identical. The fasta file
                    must contain all the proteins listed in the alignment
                    file, although additional proteins that did not align
                    may also be included and will be ignored. Protein
                    names *must not* contain pipe signs like this
                     -> | <- .

-hspa format:string   The file containing mRNA alignments (optional).
                    The filename can be preceded by a semicolon and the
                    format: psl, gff or exf. If only the file name is
                    indicated, the format is assumed to be PSL.
                    See the -hspp option for information on file formats.
```

```
-o format:string      The base for naming output files (optional).
                      The name can be preceded by semicolon and the
                      desired format: gtf or bed.
                      See the -hspp option for information on file formats.
                      If this option is not used, output files will be named
                      after the name of the DNA sequence and the date and
                      time of day, (sequencename_month_day_hour_minute) and
                      the default format will be GTF.
                      See http://genome.ucsc.edu/goldenPath/help/customTrack.html#GTF
                      for a description of the GTF format.

-offset integer       nucleotide offset to add to the genomic coordinates
                      of Exogean annotations (optional; default = 0).
                      If annotations are computed on a subsequence of a
                      chromosome, this option can be used to replace the
                      positions on chromosome coordinate.

-pipe string          string = prot or cdna or gene or pseudo (optional;
                      default= no redirection). Enables one of the result
                      files to be redirected to standard output.
                      The other five files will still be written.

-v                   verbose mode (optional)


List of arguments to customize the behaviour of Exogean for pre-processing
-------------------------------------------------------------------------
alignments.
-----------
Most are needed when building transcripts from structures based on protein
alignments. In the following, the term HSP (inherited from BLAST) is used
to describe a local alignment of a protein sequence on a genomic sequence.
A single protein sequence may produce several HSPs, (e.g. corresponding to
different exons). Options are listed below in the order in which they are
used by exogean.

-sh integer           integer = threshold in nucleotide (default=25).
                      Proteic HSPs less than this threshold are totally
                      eliminated from Exogean processing.

-pmono                If used, allows Exogean to use proteins that produce
```

```
                    a single HSP on the genomic DNA sequence. Otherwise,
                    Exogean only uses proteins that produce more than one
                    HSP on the genomic sequence (optional).

-f integer          integer = threshold in nucleotides (default=30).
                    If two neighbouring HSPs are separated by a distance
                    Dg in the genomic DNA and a distance Dp in the protein
                    sequence, the two alignments are fused if Dg=(Dp*3) +/-
                    threshold.

-I integer         integer = threshold in nucleotides (default=75000).
                    If two protein alignments are contiguous in both the
                    genomic DNA and the aligned molecule but separated by
                    more than the threshold on the genomic DNA, they will
                    be considered as being part of different transcripts.

-I2 integer         integer = threshold in nucleotide (default=10000).
-sih integer        integer = threshold in nucleotide (default=45).
                    If two protein alignments are contiguous in both the
                    genomic DNA and the aligned molecule, and separated
                    by a distance less than the threshold given by the I
                    parameter but more than the threshold given by the I2
                    paramater on the genomic DNA, and one of them is less
                    than the threshold given by the sih paramter then they
                    will be considered as being part of different transcripts.

-pm integer         integer = percentage between 1 and 100 (default=33).
-pm2 integer        integer = percentage between 1 and 100 (default=20).
-pnamq integer      integer = percentage between 1 and 100 (default=5).
-pnfus integer      integer = percentage between 1 and 100 (default=10).
                    Once the protein alignments have been processed and
                    just before entering the DACM algorithm, a last
                    verification is made on the validity of the HSPs.
                    If a preliminary gene structure satisfies either one
                    of the two following conditions, it will be considered
                    as an artifact and eliminated :
                    - it is based on one or two HSPs and the ratio between
                      the sum of their size and the size of the original
                      protein is less than the threshold given by the pm option;
                    - it is based on more than three HSPs and either one
                      of the three following conditions is satisfied :
                         * the ratio between the number of missing amino acids
```

```
                        and the number of matching amino acids is more than
                        the threshold given by the pnamq option,
                      * the ratio between the sum of the HSP sizes and the
                        size of the original protein is less than the threshold
                        given by the pm2 option,
                      * the ratio between the number of nucleotides merged
                        between all the HSPs and the total number of nucleotides
                        formed by the HSPs is more than the threshold given by
                        the pnfus option.


-naa integer       integer = distance in codons (default = 10).
                      This option is used only for exons that have protein
                      alignment support, but no mRNA alignments.
                      HSPs are often shorter than the real exons and in such
                      cases exogean must look outwards from the HSP for
                      suitable splice donor and acceptor sites.
                      Exogean will use the distance given to limit the region
                      where putative splice sites will be searched for, unless
                      the aligned protein has poorly conserved residues that
                      did not align between the two HSPs, in which case the
                      number of such amino-acids is used as the distance.

-ni integer       integer = distance in nucleotides (default=8).
                      This option is used only for exons that have protein
                      alignment support, but no mRNA alignments.
                      Sometimes protein alignments extend into introns.
                      To allow for such cases, Exogean will also search
                      putative splice sites within this distance inside of HSPs.

-ns integer           integer = distance in nucleotides (default=15; must be
                      a multiple of 3). This option is used only for exons
                      that have protein alignment support, but no mRNA alignments.
                      When looking for splice signals outside or inside of HSPs,
                      exogean will either be guided by the aligned protein if
                      adjacent HSPs are from the same protein, or use the values
                      given by the -naa option to limit the search.
                      In either cases, the value given with the -ns option is added
                      as a tolerance threshold to extend further outside of HSPs,
                      or to extend the limits provided by the aligned protein.

-i integer            integer = threshold in nucleotides (default=60).
```

```
                    This option is used only for exons that have protein
                    alignment support, but no mRNA alignments.
                    When searching for putative splice sites between two adjacent
                    HSPs, exogean will eliminate all combinations of donor and
                    acceptor sites that will create an intron of size less than
                    this threshold. An important consequence is that when only
                    protein alignments are provided, final genes structures will
                    always have introns larger than this threshold

-mgt string         string = name of fasta file (optional).
                    The multi fasta file must contain a list of 10 bp sequences
                    that are used to compute a position weight matrix for splice
                    donor sites. Position 1-4 are in the exon and 5-10 are in the
                    intron. Exogean comes with a sig_GT.fa file that can be used
                    here for human genes (and also works well for other mammalian
                    species) that was extracted from the HS3D database
                    (http://www.sci.unisannio.it/docenti/rampone/).

-mag string       string = name of fasta file (optional).
                    The multi fasta file must contain a list of 10 bp sequences
                    that are used to compute a position weight matrix for splice
                    acceptor sites. Position 1-6 are in the intron and 7-10 are
                    in the exon. Exogean comes with a sig_AG.fa file that can be
                    used here for human genes (and also works well for other
                    mammalian species) that was extracted from the HS3D
                    database (http://www.sci.unisannio.it/docenti/rampone/).

-wmat | -waa        makes a choice between two alternative methods to select
                    the best splice donor and acceptor sites when information
                    flanking a putative intron is only based on proteins
                    (default: wmat). The -wmat option uses a position weight
                    matrix score (built from the files given with the -mgt
                    and -mag arguments) to select the most favourable intron
                    between two HSPs. The -waa option instead uses the protein
                    alignment as guide (if available), and will fill in a number
                    of codons at the ends of putative exons, to approach the
                    number of amino acids that are in the protein at this position
                    but that could not be aligned.
```

```
List of arguments that control the last validation of Exogean predictions
-------------------------------------------------------------------------
(post-processing)
-----------------
These filters are applied on Exogean predictions to classify them as
\"correct\" or artifacts/pseudogenes. Ideally, Exogean predicts genes
with a CDS starting with a methionine and ending at a stop codon.
When these signals cannot be found or the CDS is small, Exogean may
filter out some predictions based on the arguments given below.

-scnmint              integer = size in nucleotides. Must be multiple of 3
                      (Default=300). If a CDS is found but without a starting
                      methionine, it must be at least this size.

-scmint               integer = size in nucleotides. Must be multiple of 3
                      (Default=210). If a CDS is found with a starting
                      methionine, it must nevertheless be larger than this
                      size to be considered valid.

-pcdsint              integer = percentage between 1 and 100 (Default=20).
                      The size of the CDS of a mono-exonic transcript must
                      represent at least this fraction of the total size
                      of the transcript (UTRs included) to be considered
                      valid.

-scdsmint             integer = size in nucleotides (Default=420).
                      If a CDS spans a unique exon, it must be at least
                      this size to be considered valid, regardless of the
                      presence of a starting methionine.

-scdsdint             integer = size in nucleotides (Default=210).
                      If a CDS spans only two exons, it must be at least
                      this size to be considered valid, regardless of a
                      starting methionine.
```

Bibliographie

[1] http ://gallium.inria.fr/.

[2] http ://genome.gov/10005107.

[3] http ://genome.imim.es/gencode/.

[4] http ://image.llnl.gov/image/html/igoals.shtml.

[5] http ://www-igm.univ-mlv.fr/ dr/xpose2002/neurones/index.php ?rubrique=accueil.

[6] http ://www.acedb.org/.

[7] http ://www.fruitfly.org/annot/apollo/userguide.html.

[8] http ://www.ncbi.nlm.nih.gov.csulib.ctstateu.edu/blast/.

[9] http ://www.ncbi.nlm.nih.gov/ftp/.

[10] http ://www.sanger.ac.uk/hgp/havana/.

[11] www-prima.inrialpes.fr/prima/homepages/jlc/courses/1999/ensi3.se/ensi3.se.s7.pdf.

[12] www.dmi.usherb.ca/ batanado/work/presentation/arbres_decision_panawe.ppt.

[13] www.regionetdeveloppement.u-3mrs.fr/pdf/r5_thuillier.pdf.

[14] *The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life.* 1859.

[15] *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[16] *Théorie des langages et automates.* Masson, 1994.

[17] *Molecular Evolution.* Sinauer, 1997.

[18] *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[19] *Développement d'applications avec Objective Caml.* O'Reilly France, 2000.

[20] *Molecular Evolution and Phylogenetics.* Oxford university press, 2000.

[21] *Algorithmique du texte.* Vuibert, 2001.

[22] *Bioinformatique, genomique et post-génomique.* Ellipses, 2002.

[23] *Biologie.* de Boeck, 2004.

[24] Goffeau A., Barrell B.G., Bussey H., Davis R.W., Dujon B., and Feldmann H. et al. Life with 6000 genes. *Science*, 1996.

[25] Lomsadze A., Ter-Hovhannisyan V., Chernoff Y. O., and Borodovsky M. Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, 2005.

[26] Marina Alexandersson, Simon Cawley, and Lior Pachter. SLAM : cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Res*, 13(3) :496–502, Mar 2003.

[27] Jonathan E Allen, Mihaela Pertea, and Steven L Salzberg. Computational gene prediction using multiple sources of evidence. *Genome Res*, 14(1) :142–148, Jan 2004.

[28] Jonathan E Allen and Steven L Salzberg. JIGSAW : integration of multiple sources of evidence for gene prediction. *Bioinformatics*, 21(18) :3596–3603, Sep 2005.

[29] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3) :403–410, Oct 1990.

[30] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST : a new generation of protein database search programs. *Nucleic Acids Res*, 25(17) :3389–3402, Sep 1997.

[31] J.L. Ashurst, C.-K. Chen, J.G.R. Gilbert, K. Jekosh, S. Keenan, P. Meidl, S.M. Searle, J. Stalker, R. Storey, and S. Trevanion. The vertebrate genome annotation (vega) database. *Nucl. Acids Res.*, 2005.

[32] V. Bafna and D. H. Huson. The conserved exon method for gene finding. *Proc Int Conf Intell Syst Mol Biol*, 8 :3–12, 2000.

[33] V.B. Bajic. Comparing the success of different prediction software in sequence analysis : a review. *Brief Bioinform.*, 2000.

[34] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure : comparative analysis and application to exon prediction. *Genome Res*, 10(7) :950–958, Jul 2000.

[35] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank. *Nucleic Acids Res*, 31, 2003.

[36] E. Birney, J. D. Thompson, and T. J. Gibson. PairWise and SearchWise : finding the optimal alignment in a simultaneous comparison of a protein profile against all DNA translation frames. *Nucleic Acids Res*, 24(14) :2730–2739, Jul 1996.

[37] Ewan Birney, Michele Clamp, and Richard Durbin. GeneWise and Genomewise. *Genome Res*, 14(5) :988–995, May 2004.

[38] P. Blayo, P. Rouzé, and M.-F. Sagot. Orphan gene finding - an exon assembly approach. *Theoretical Computer Science*, 290 :1407–1431, 2003.

[39] M. Borodovsky and J. McIninch. Recognition of genes in DNA sequence with ambiguities. *Biosystems*, 30(1-3) :161–171, 1993.

[40] Brona Brejová, Daniel G Brown, Ming Li, and Tomás Vinar. ExonHunter : a comprehensive approach to gene finding. *Bioinformatics*, 21 Suppl 1 :i57–i65, Jun 2005.

[41] V. Brendel, L. Xing, and W. Zhu. Gene structure prediction from consensus spliced aligment of multiple ests matching the same genomic locus. *Bioinformatics*, 20 :1157–1169, 2004.

[42] M.R. Brent. Genome annotation past, present, and future : How to define an orf at each locus. *Genome Res.*, 2005.

[43] Randall H Brown, Samuel S Gross, and Michael R Brent. Begin at the beginning : predicting genes with 5' UTRs. *Genome Res*, 15(5) :742–747, May 2005.

[44] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *J Mol Biol*, 268(1) :78–94, Apr 1997.

[45] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34(3) :353–367, Jun 1996.

[46] D. Cai, A. Delcher, B. Kao, and S. Kasif. Modeling splice sites with bayes networks. *Bioinformatics*, 16(2) :152–158, Feb 2000.

[47] R. Castelo and R. Guigo. Splice site identification by idlbns. *Bioinformatics*, 20 Suppl 1 :I69–I76, Aug 2004.

[48] Te-Ming Chen, Chung-Chin Lu, and Wen-Hsiung Li. Prediction of splice sites with dependency graphs and their expanded bayesian networks. *Bioinformatics*, 21(4) :471–482, Feb 2005.

[49] Trees-Juen Chuang, Feng-Chi Chen, and Meng-Yuan Chou. A comparative method for identification of gene structures and alternatively spliced variants. *Bioinformatics*, 20(17) :3064–3079, Nov 2004.

[50] Alexander Churbanov, Mark Pauley, Daniel Quest, and Hesham Ali. A method of precise mRNA/DNA homology-based gene structure prediction. *BMC Bioinformatics*, 6 :261, 2005.

[51] J. M. Claverie. From bioinformatics to computational biology. *Genome Res*, 10(9) :1277–1279, Sep 2000.

[52] The ENCODE Project Consortium. The encode (encyclopedia of dna elements) project. *Science*, 2004.

[53] H. Roest Crollius, O. Jaillon, A. Bernot, C. Dasilva, L. Bouneau, C. Fischer, C. Fizames, P. Wincker, P. Brottier, F. Quétier, W. Saurin, and J. Weissenbach. Estimate of human gene number provided by genome-wide analysis using Tetraodon nigroviridis DNA sequence. *Nat Genet*, 25(2) :235–238, Jun 2000.

[54] Val Curwen, Eduardo Eyras, T. Daniel Andrews, Laura Clarke, Emmanuel Mongin, Steven M J Searle, and Michele Clamp. The Ensembl automatic gene annotation system. *Genome Res*, 14(5) :942–950, May 2004.

[55] Simon D., Fewer D., Friedl T., and Bhattacharya D. Phylogeny and self-splicing ability of the plastid trna-leu group i intron. *J Mol Evol.*, 2003.

[56] R. V. Davuluri, I. Grosse, and M. Q. Zhang. Computational identification of promoters and first exons in the human genome. *Nat Genet*, 29(4) :412–417, Dec 2001.

[57] Sven Degroeve, Yvan Saeys, Bernard De Baets, Pierre Rouzé, and Yves Van de Peer. SpliceMachine : predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8) :1332–1338, Apr 2005.

[58] A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg. Improved microbial gene identification with GLIMMER. *Nucleic Acids Res*, 27(23) :4636–4641, Dec 1999.

[59] Li Ding, Aniko Sabo, Nicolas Berkowicz, Rekha R Meyer, Yoram Shotland, Mark R Johnson, Kymberlie H Pepin, Richard K Wilson, and John Spieth. EAnnot : a genome annotation tool using experimental evidence. *Genome Res*, 14(12) :2503–2509, Dec 2004.

[60] S. Djebali, F. Delaplace, and H. Roest Crollius. Exogean : an expert on eukaryotic gene annotation. Technical report, LaMI laboratory, Evry University, 2004.

[61] S. Djebali, F. Delaplace, and H. Roest Crollius. Exogean : a framework for annotating protein-coding genes in eukaryotic genomic dna. *Genome Biology*, 2006.

[62] I. Dunham, N.Shimizu, B.A. Roe, S. Chissoe, A.R. Hunt, J.E. Collins, R. Bruskiewich, D.M. Beare, M. Clamp, and L.J. Smink et al. The dna sequence of human chromosome 22. *Nature*, 1999.

[63] E. Eden and S. Brunak. Analysis and recognition of 5' UTR intron splice sites in human pre-mRNA. *Nucleic Acids Res*, 32(3) :1131–1142, 2004.

[64] Lander E.S., Linton L.M., Birren B., Nusbaum C., Zody M.C., and Baldwin J. et al. Initial sequencing and analysis of the human genome. *Nature*, 2001.

[65] R. Guigo et al. Egasp : The encode genome annotation assessment project. *Genome Biology*, 2006.

[66] Eduardo Eyras, Mario Caccamo, Val Curwen, and Michele Clamp. ESTGenes : alternative splicing from ESTs in Ensembl. *Genome Res*, 14(5) :976–987, May 2004.

[67] J. W. Fickett and C. S. Tung. Assessment of protein coding measures. *Nucleic Acids Res*, 20(24) :6441–6450, Dec 1992.

[68] Paul Flicek, Evan Keibler, Ping Hu, Ian Korf, and Michael R Brent. Leveraging the mouse genome for gene prediction in human : from whole-genome shotgun reads to a global synteny map. *Genome Res*, 13(1) :46–54, Jan 2003.

[69] L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Res*, 8(9) :967–974, Sep 1998.

[70] Liliana Florea, Valentina Di Francesco, Jason Miller, Russell Turner, Alison Yao, Michael Harris, Brian Walenz, Clark Mobarry, Gennady V Merkulov, Rosane Charlab, Ian Dew, Zuoming Deng, Sorin Istrail, Peter Li, and Granger Sutton. Gene and alternative splicing annotation with AIR. *Genome Res*, 15(1) :54–66, Jan 2005.

[71] Sylvain Foissac, Philippe Bardou, Annick Moisan, Marie-Josée Cros, and Thomas Schiex. EUGENE'HOM : A generic similarity-based gene finder using multiple homologous sequences. *Nucleic Acids Res*, 31(13) :3742–3745, Jul 2003.

[72] Sylvain Foissac and Thomas Schiex. Integrating alternative splicing detection into gene prediction. *BMC Bioinformatics*, 6(1) :25, 2005.

[73] Schuler G.D. Pieces of the puzzle : expressed sequence tags and the catalog of human genes. *J Mol Med*, 1997.

[74] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proc Natl Acad Sci U S A*, 93(17) :9061–9066, Aug 1996.

[75] R. Guigo and M. G. Reese. Egasp : collaboration through competition to find human genes. *Nature Methods*, 2005.

[76] R. Guigó. Assembling genes from predicted exons in linear time with dynamic programming. *J Comput Biol*, 5(4) :681–702, 1998.

[77] R. Guigó, P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Res*, 10(10) :1631–1642, Oct 2000.

[78] C. A. Gunter and D. S. Scott. *Handbook of Theoretical Computer Science*, volume 2, chapter Semantic Domains, pages 633–674. Elsevier Science, 1990.

[79] S. M. Hebsgaard, P. G. Korning, N. Tolstrup, J. Engelbrecht, P. Rouzé, and S. Brunak. Splice site prediction in Arabidopsis thaliana pre-mRNA by combining local and global sequence information. *Nucleic Acids Res*, 24(17) :3439–3452, Sep 1996.

[80] J. Henderson, S. Salzberg, and K. H. Fasman. Finding genes in DNA with a Hidden Markov Model. *J Comput Biol*, 4(2) :127–141, 1997.

[81] R. T. Hillman, R. E. Green, and S. E. Brenner. An unappreciated role for rna surveillance. *Genome Biology*, 2004.

[82] Loi Sy Ho and Jagath C Rajapakse. Splice site detection with a higher-order markov model implemented on a neural network. *Genome Inform Ser Workshop Genome Inform*, 14 :64–72, 2003.

[83] P. M. Hooper, H. Zhang, and D. S. Wishart. Prediction of genetic structure in eukaryotic DNA using reference point logistic regression and sequence alignment. *Bioinformatics*, 16(5) :425–438, May 2000.

[84] K.L. Howe, T. Chothia, and R. Durbin. Gaze : a generic framework for the integration of gene-prediction data by dynamic programming. *Genome Res.*, 2002.

[85] D. Hyatt, J. Snoddy, D. Schmoyer, G. Chen, K. Fischer, M. Parang, I. Vokler, S. Petrov, P. Locascio, V. Olman, M. Land, M. Shah, and E. Uberbacher. Improved analysis and annotation tools for whole-genome computational annotation and analysis : Grail-exp genome analysis toolkit and related analysis tools.

[86] Biju Issac and Gajendra Pal Singh Raghava. EGPred : prediction of eukaryotic genes using ab initio methods after combining with sequence similarity approaches. *Genome Res*, 14(9) :1756–1766, Sep 2004.

[87] Munch K. and Krogh A. Automatic generation of gene finders for eukaryotic species. *BMC Bioinformatics*, 2006.

[88] Z. Kan, E. C. Rouchka, W. R. Gish, and D. J. States. Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. *Genome Res*, 11(5) :889–900, May 2001.

[89] E. Keibler and M.R. Brent. Eval : A software package for analysis of genome annotations. *BMC Bioinformatics*, 2003.

[90] W. James Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4) :656–664, Apr 2002.

[91] P. J. Kersey, J. Duarte, A. Williams, Y. Karavidopoulou, E. Birney, and R. Apweiler. The international protein index : An integrated database for proteomics experiments. *Proteomics*, 4(7), 2004.

[92] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17 Suppl 1 :S140–S148, 2001.

[93] Ian Korf. Gene finding in novel genomes. *BMC Bioinformatics*, 5 :59, May 2004.

[94] M. Kozak. The scanning model for translation :an update. *Journal of Cell Biology*, 1989.

[95] A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. *Proc Int Conf Intell Syst Mol Biol*, 5 :179–186, 1997.

[96] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. *Proc Int Conf Intell Syst Mol Biol*, 4 :134–142, 1996.

[97] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. Integrating database homology in a probabilistic gene structure model. *Pac Symp Biocomput*, pages 232–244, 1997.

[98] A. V. Lukashin and M. Borodovsky. GeneMark.hmm : new solutions for gene finding. *Nucleic Acids Res*, 26(4) :1107–1115, Feb 1998.

[99] William H Majoros, Mihaela Pertea, Corina Antonescu, and Steven L Salzberg. GlimmerM, Exonomy and Unveil : three ab initio eukaryotic genefinders. *Nucleic Acids Res*, 31(13) :3601–3604, Jul 2003.

[100] William H Majoros, Mihaela Pertea, Arthur L Delcher, and Steven L Salzberg. Efficient decoding algorithms for generalized hidden Markov model gene finders. *BMC Bioinformatics*, 6(1) :16, 2005.

[101] Salzberg SL. Majoros WH, Pertea M. Tigrscan and glimmerhmm : two open source ab initio eukaryotic gene-finders. *Bioinformatics*, 2004.

[102] Catherine Mathé, Marie-France Sagot, Thomas Schiex, and Pierre Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res*, 30(19) :4103–4117, Oct 2002.

[103] I. M. Meyer and R. Durbin. Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Res*, 32(2) :776–783, 2004.

[104] Irmtraud M Meyer and Richard Durbin. Comparative ab initio prediction of gene structures using pair HMMs. *Bioinformatics*, 18(10) :1309–1318, Oct 2002.

[105] P. D. Mosses. *Handbook of Theoretical Computer Science*, volume 2, chapter Denotational Semantics, pages 575–631. Elsevier Science, 1990.

[106] R. Mott. EST_GENOME : a program to align spliced DNA sequences to unspliced genomic DNA. *Comput Appl Biosci*, 13(4) :477–478, Aug 1997.

[107] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3) :443–453, Mar 1970.

[108] P. S. Novichkov, M. S. Gelfand, and A. A. Mironov. Gene recognition in eukaryotic DNA by comparison of genomic sequences. *Bioinformatics*, 17(11) :1011–1018, Nov 2001.

[109] Bonizzoni P., Rizzi R., and Pesole G. Aspic : a novel method to predict the exon-intron structure of a gene that is optimally compatible to a set of transcript sequences. *BMC Bioinformatics.*, 2005.

[110] Y. Pan and C. W. Sensen. Modular neural networks and their application in exon prediction.

[111] G. Parra, E. Blanco, and R. Guigó. GeneID in Drosophila. *Genome Res*, 10(4) :511–515, Apr 2000.

[112] Genís Parra, Pankaj Agarwal, Josep F Abril, Thomas Wiehe, James W Fickett, and Roderic Guigó. Comparative gene prediction in human and mouse. *Genome Res*, 13(1) :108–117, Jan 2003.

[113] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85(8) :2444–2448, Apr 1988.

[114] A. G. Pedersen and H. Nielsen. Neural network prediction of translation initiation sites in eukaryotes : Perspectives for est and genome analysis. *ISMB*, 5 :226–233, 1997.

[115] C. N. S. Pedersen and T. Sharling. Comparative methods for gene structure prediction in homologous sequences.

[116] M. Pertea, X. Lin, and S. L. Salzberg. GeneSplicer : a new computational method for splice site prediction. *Nucleic Acids Res*, 29(5) :1185–1190, Mar 2001.

[117] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition.

[118] Fleischmann R.D., Adams M.D., White O., Clayton R.A., Kirkness E.F., Kerlavage A.R., Bult C.J., Tomb J.F., Dougherty B.A., and Merrick J.M. et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 1995.

[119] M. G. Reese, F. H. Eeckman, D. Kulp, and D. Haussler. Improved splice site detection in Genie. *J Comput Biol*, 4(3) :311–323, 1997.

[120] M. G. Reese, D. Kulp, H. Tammana, and D. Haussler. Genie–gene finding in Drosophila melanogaster. *Genome Res*, 10(4) :529–538, Apr 2000.

[121] S. Rogic, A. K. Mackworth, and F. B. Ouellette. Evaluation of gene-finding programs on mammalian sequences. *Genome Res*, 11(5) :817–832, May 2001.

[122] A. A. Salamov, T. Nishikawa, and M. B. Swindells. Assessing protein coding region integrity in cdna sequencing projects. *Bioinformatics*, 14(5) :384–390, Jun 1998.

[123] A. A. Salamov and V. V. Solovyev. Ab initio gene finding in Drosophila genomic DNA. *Genome Res*, 10(4) :516–522, Apr 2000.

[124] S. Salzberg, A. L. Delcher, K. H. Fasman, and J. Henderson. A decision tree system for finding genes in DNA. *J Comput Biol*, 5(4) :667–680, 1998.

[125] S. L. Salzberg, M. Pertea, A. L. Delcher, M. J. Gardner, and H. Tettelin. Interpolated Markov models for eukaryotic gene finding. *Genomics*, 59(1) :24–31, Jul 1999.

[126] T. Schiex, L. Duret, and P. Rouzé. A simple yet effective gene finder for eukaryotic organisms (arabidopsis thaliana).

[127] Sohrab P Shah, Graham P McVicker, Alan K Mackworth, Sanja Rogic, and B. F Francis Ouellette. GeneComber : combining outputs of gene prediction programs for improved results. *Bioinformatics*, 19(10) :1296–1297, Jul 2003.

[128] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1) :195–197, Mar 1981.

[129] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA sequences : an application of dynamic programming and neural networks. *Nucleic Acids Res*, 21(3) :607–613, Feb 1993.

[130] E. E. Snyder and G. D. Stormo. Identification of protein coding regions in genomic DNA. *J Mol Biol*, 248(1) :1–18, Apr 1995.

[131] V. Solovyev and A. Salamov. The Gene-Finder computer tools for analysis of human and model organisms genome sequences. *Proc Int Conf Intell Syst Mol Biol*, 5 :294–302, 1997.

[132] V. V. Solovyev, A. A. Salamov, and C. B. Lawrence. Identification of human gene structure using linear discriminant functions and dynamic programming. *Proc Int Conf Intell Syst Mol Biol*, 3 :367–375, 1995.

[133] Mario Stanke and Stephan Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19 Suppl 2 :II215–II225, Oct 2003.

[134] Castrignano T., Canali A., Grillo G., Liuni S., Mignone F., and Pesole G. Cstminer : a web tool for the identification of coding and noncoding conserved sequence tags through cross-species genome comparison. *Nucleic Acids Res.*, 2004.

[135] J. E. Tabaska, R. V. Davuluri, and M. Q. Zhang. Identifying the 3'-terminal exon in human DNA. *Bioinformatics*, 17(7) :602–607, Jul 2001.

[136] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy. Prediction of probable genes by Fourier analysis of genomic sequences. *Comput Appl Biosci*, 13(3) :263–270, Jun 1997.

[137] E.C. Uberbacher and R.J. Mural. Locating protein coding regions in human dna sequences using a multiple sensor-neural network approach. *Proceedings of the National Academy of Sciences of the USA*, 1991.

[138] J. Usuka and V. Brendel. Gene structure prediction by spliced alignment of genomic DNA with protein sequences : increased accuracy by differential splice site scoring. *J Mol Biol*, 297(5) :1075–1085, Apr 2000.

[139] A. J. Viterbi. Error bounds for convolutional codes and an asymtotically opti mum decoding algorithm.

[140] S. J. Wheelan, D. M. Church, and J. M. Ostell. Spidey : a tool for mRNA-to-genomic alignments. *Genome Res*, 11(11) :1952–1957, Nov 2001.

[141] R. Wheeler. A method of consolidating and combining est and mrna alignments to a genome to enumerate supported splice variants.

[142] T. Wiehe, S. Gebauer-Jung, T. Mitchell-Olds, and R. Guigó. SGP-1 : prediction and validation of homologous genes based on sequence alignments. *Genome Res*, 11(9) :1574–1583, Sep 2001.

[143] Yi Xing, Alissa Resch, and Christopher Lee. The multiassembly problem : reconstructing multiple transcript isoforms from EST fragment mixtures. *Genome Res*, 14(3) :426–441, Mar 2004.

[144] T. Yada, T. Takagi, Y. Totoki, Y. Sakaki, and Y. Takaeda. DIGIT : a novel gene finding program by combining gene-finders. *Pac Symp Biocomput*, pages 375–387, 2003.

[145] R. F. Yeh, L. P. Lim, and C. B. Burge. Computational inference of homologous gene structures in the human genome. *Genome Res*, 11(5) :803–816, May 2001.

[146] M. Q. Zhang. Identification of protein coding regions in the human genome by quadratic discriminant analysis. *Proc Natl Acad Sci U S A*, 94(2) :565–568, Jan 1997.

[147] C. L. Zheng, V. R. de Sa, M. Gribskov, and T. M. Nair. On selecting features from splice junctions : An analysis using information theoretic and machine learning approaches. *Genome Informatics*, 14 :73–83, 2003.

[148] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K. R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9) :799–807, Sep 2000.