# Recherche d'instances de motifs expressifs avec Logol. Application à la modélisation d'événements de frameshift -1

Catherine Belleannée, Olivier Sallou, Jacques Nicolas

# Expressive Pattern Matching with Logol. Application to the Modelling of -1 Ribosomal Frameshift events

Catherine BELLEANNÉE[1], Olivier SALLOU[1] and Jacques NICOLAS[1]

Irisa/Inria/Université de Rennes1, campus de Beaulieu, 35042 Rennes Cedex France
{Catherine.Belleannee,Jacques.Nicolas,Olivier.Sallou}@irisa.fr

**Abstract** *The current practice of pattern matching tools and the gap that may be observed with the actual modelling needs of people analysing genome structures clearly demonstrates the need for higher level languages to describe and search for these structures in genomic sequences. It appears necessary to offer new tools allowing to build more expressive models of families of biological sequences, on the basis of their content and structure. This article presents Logol, a new application designed to achieve pattern matching in possibly large sequences with realistic biological motifs. Logol consists in both a language for describing patterns, and the associated parser for effectively scanning sequences (RNA, DNA or protein) with such motifs. The language, based on an high level gramatical formalism, allows to express flexible patterns (with misparings and indels) composed of both sequential and structural elements (such as repeats or pseudoknots). A web page on the GenOuest BioInformatics Platform* http://www.genouest.org/ *gives access to the Logol application. It includes an interface for graphically drawing the motif model and an interface to display the resulting matches within the targetted pattern.*
*Logol is presented through an illustrative application using a quite intricate motif model, which is the detection of -1 ribosomal frameshifting events in messenger RNA sequences.*

**Keywords** Pattern Matching, Sequence Modelling, String Variable Grammar, Frameshift event, Expressive Motif, Pseudo-Knot, Complex Pattern

## Recherche d'instances de motifs expressifs avec Logol. Application à la modélisation d'événements de *frameshift -1*

**Résumé** *L'état de la pratique des outils de reconnaissance de motifs et l'écart qui peut être observé avec les besoins réels de modélisation des personnes en charge de l'analyse des structures génomiques montrent clairement le besoin de langages de plus haut niveau pour décrire et rechercher ces structures dans les séquences génomiques. Il apparaît ainsi nécessaire de proposer de nouveaux outils permettant de définir des modèles expressifs de familles de séquences biologiques, modèles basés à la fois sur le contenu et la structure des séquences. Cet article présente Logol, une application de reconnaissance de motifs conçue pour analyser des séquences potentiellement grandes avec des motifs biologiques réalistes. Logol est constitué d'un langage de description de motifs et de la suite logicielle associée, permettant de réaliser effectivement l'analyse de séquences (d'ADN, ARN ou protéines) avec ces motifs. Le langage, basé sur un formalisme grammatical de haut niveau, permet d'exprimer des motifs flexibles (autorisant substitutions et indels) composés à la fois d'éléments de séquences et de structures (tels que des répétitions ou des pseudo-noeuds). La suite logicielle est accessible sur le web, sur la plateforme bioinformatique GenOuest* http://www.genouest.org/*. Elle contient notamment deux interfaces, l'une pour dessiner graphiquement le modèle de motif et la seconde pour afficher les résultats comme des instances de ce modèle.*
*Logol est présenté au travers d'une application illustrant les concepts utiles via l'utilisation d'un modèle de motif assez riche. Il s'agit de la détection d'événements de décalage de phase en -1 dans les ARN messagers.*

**Mots-clés** Reconnaissance de motifs, Modélisation de séquences, Grammaire à variables de chaînes, événements de décalage de cadre en -1, motifs expressifs, pseudo-noeuds, pattern complexes

## 1 Introduction

La reconnaissance de motifs consiste, étant donné un motif et une séquence d'entrée, à trouver les occurrences du motif dans la séquence. En biologie, il existe un besoin croissant de recherche de motifs variés afin d'aider les biologistes à analyser leurs séquences. Il peut s'agir de rechercher toutes les instances exactes d'une chaîne dans une banque de protéines, de rechercher les instances approchées d'un transposon dans un génome entier, de localiser des pseudo-noeuds dans des séquences d'ARN etc. Ainsi, ces dernières années, de nombreux outils ont été conçus pour répondre à un certain nombre de ces objectifs, et beaucoup d'entre eux sont utiles et utilisés. Il apparaît cependant qu'un besoin subsiste d'un outil de modélisation de séquences générique, ouvert (i.e. évolutif et non spécialisé sur la reconnaissance d'un type de motif, ni sur l'analyse d'un type de séquence) et réellement opérationnel. C'est dans cette perspective que nous proposons Logol, un nouvel outil général de reconnaissance de motifs dont les objectifs sont à la fois d'être le plus expressif possible (dans le sens où il vise à traduire des modèles complexes, déjà imaginés ou émergeants) et raisonnablement efficace (dans le sens où il doit être utilisable en pratique sur de grandes séquences, de par ses performances et son ergonomie).
Après avoir présenté un panorama de travaux existants en reconnaissance de motifs, nous présentons les bases de Logol et ses principaux constituants, pour ce qui concerne à la fois le langage de motif et l'outil d'analyse de séquences. Une exploration plus détaillée de certains éléments de Logol est ensuite effectuée au travers l'observation d'un exemple d'application de Logol : l'aide à la détection d'événements de "frameshift-1" dans les ARN messagers. Le modèle concerné ici est particulièrement riche puisqu'il contient des informations de séquence (telles que la détection de codons start et stop, ou de "fenêtres glissantes") et de structure (recherche de pseudo-nœuds, calage de phases de lecture), et amène à investiguer la composition de segments (pourcentage de GC) ou le type d'appariement utilisé dans les tiges-boucles (Watson-Crick avec ou sans Wobble).

## 2 Reconnaissance de motifs

De nombreux outils existent pour effectuer de la recherche de motifs, répondant à un large spectre d'objectifs. Nous dressons ici un panorama -non exhaustif- de ces outils et de leurs objectifs pour situer Logol dans cette perspective.

### 2.1 Outils généraux

Certains des outils de reconnaissance de motifs sont conçus pour analyser les différents types de séquences (ADN, ARN ou protéines) et ont une expressivité "ouverte", dans le sens où ils ne sont pas dédiés à la recherche d'un type de motif particulier. Ce sont donc les outils les plus *généraux*. Parmi ceux-là, certains ont pour objectif principal d'être le plus *efficace* possible. C'est particulièrement le cas pour Vmatch[15] qui offre une suite logicielle pour résoudre efficacement une grande étendue de tâches de reconnaissance. L'efficacité est obtenue grace à un précalcul sur les séquences d'entrée qui génère une structure d'index. L'index, un "tableau de suffixes amélioré", référence toutes les sous-chaînes des séquences analysées. L'indexation peut réduire de façon considérable le temps nécessaire à la localisation des motifs, surtout si le motif contient des sous-chaînes discriminantes. De nombreux outils plus spécifiques basent leurs recherches sur la suite Vmatch (e.g. recherche de "tandem-repeats" ou de retrotransposons-LTR [15]). Un autre outil général, Biogrep[13], a été conçu avec l'objectif de *reconnaître rapidement de nombreux motifs simples* (plus de 100) contre les banques de séquences biologiques. Biogrep utilise POSIX, format standard d'expressions régulières étendues, et peut répartir la tâche de reconnaissance de motifs sur un certain nombre de processeurs.

Parmi les outils généraux, certains autres tendent à être le plus *expressif* possible. Une contribution essentielle à cet objectif est due à D. Searls, qui a posé les fondations d'une recherche dans le domaine. Il a été le premier à superviser des développements permettant aux utilisateurs de concevoir des grammaires puis analyser des séquences génomiques avec celles-ci [23,7,22]. Il a introduit un nouveau type d'objet dans les grammaires algébriques, la *variable de chaîne*, qui permet d'exprimer élégamment la notion de copie (directe ou inverse). Il a mis en œuvre le formalisme résultant, appelé SVG pour "String Variable Grammars", dans l'outil Genlang[7]. La copie directe (*e.g.* `X...X`) permet de rechercher deux copies d'une même chaîne inconnue, avec éventuellement une indication sur la taille de la chaîne. La copie inverse quant-à elle (*e.g.*

X... ∼X) permet de rechercher une chaîne et son complément inverse biologique, ce qui permet d'exprimer des palindromes biologiques tels que les tige-boucles (Stem, Loop, ∼Stem) ou les pseudo-nœuds (Stem1, Loop1,Stem2, Loop2, ∼Stem1, Loop3,∼Stem2). Genlang, Stan[18] (développé dans notre équipe), Patscan[8] et Patsearch[19] sont des outils de cette famille. Grâce aux variables de chaînes et à des composants additionnels, ces langages permettent de mêler facilement dans les modèles des informations de séquence et de structure.

### 2.2 Outils dédiés

Il est impossible ici de fournir une vue complète du foisonnement d'outils spécifiques qui ont été mis à disposition des bioanalystes. Certains sont spécifiques à une famille de séquences ou à un type de motif. L'outil le plus connu qui soit *dédié aux proteines* est ScanProsite[6], où les motifs sont basés sur les expressions régulières, recherchées soit dans une base de données précalculée soit par l'algorithme PS-SCAN[11].

Un grand nombre d'outils est *dédié aux séquences d'ARN*, du fait de la nécessité d'explorer les très complexes structures d'ARN, particulièrement chez les ARN non codants. Par exemple, RNAmotif[16], RNAbob[9], Hypasearch[12,24] et Palingol[5] permettent de décrire les motifs comme une succession de tiges et de boucles, avec généralement la possibilité de choisir le type d'appariement entre l'appariement standard Watson-Crick (A−U, G−C) ou l'appariement avec Wobble (A−U, G−C, G−U). Les motifs peuvent aussi contenir des informations de séquence qui doivent être présentes dans certaines parties des tiges ou des boucles. RNAmotif[16] est probablement le plus populaire de ces outils. Un nouvel outil de cette famille, Structator[17], améliore significativement le temps d'analyse grâce à une structure d'index adaptée à l'analyse de palindromes, les "tableaux affixes".

L'outil Locomotif[20] a presque la même expressivité (un peu moins) que les précédents mais il est conçu pour répondre à des objectifs supplémentaires intéressants. Un premier est de proposer une *conception graphique du motif*. Le motif est décrit graphiquement et l'analyseur correspondant est dérivé du graphique de façon complètement automatique. L'éditeur permet de dessiner des structures secondaires, par composition de tiges et de boucles, annotées avec des informations de séquence et de taille. De plus, un second objectif est de ne renvoyer qu'un seul résultat *le meilleur résultat d'après un modèle thermodynamique*.

## 3 Langage Logol : Quelle expressivité ?

### 3.1 A la base : Modèle grammatical

• **Grammaires à Variables de Chaînes** Ayant vocation à être un langage général et expressif, et permettant d'exprimer naturellement la notion de structure, le langage Logol a pris pour base les grammaires à variables de chaînes (grammaires SVG) définies par D.Searls [23,7,22] et introduites en section 2.1. Comme on l'a déjà évoqué, si l'expression de motifs ou de gaps est accessible dès le niveau des langages réguliers (*e.g.* PROSITE [6]), les grammaires SVG permettent de plus l'expression des palindromes (nécessaires pour traduire tiges-boucles et pseudo-nœuds) accessible à partir des langages algébriques, ainsi que la notion de répétition (duplication d'une sous-chaîne) qui nécessite une expressivité encore supérieure, et qui est captée par les variables de chaînes. Ainsi, les grammaires SVG et celles de Logol se situent au-delà des grammaires algébriques, dans la classe que A. Joshi appelle "faiblement contextuelle" (*midly context sensitive languages*) [14]. Si Logol a construit ses bases autour des grammaires SVG, le langage a été ensuite largement étendu avec l'objectif de le rendre le plus adapté possible à l'expression de motifs biologiques réalistes. Nous en décrivons ici les principaux constituants.

• **Premières grammaires Logol** La grammaire Logol suivante permet de rechercher toutes les instances du motif "aaa" dans une séquence :

```
mod1()==>"aaa"
mod1()==*> SEQ1
```

La dernière ligne est le point d'entrée de la grammaire, appelé "rule" dans le modèle gaphique. Elle indique quel est le modèle, ici mod1(), à rechercher dans la séquence. Les autres lignes donnent la définition du modèle (i.e. du motif). Le motif recherché par la deuxième grammaire est constitué de deux copies d'une même chaîne,

de taille comprise entre 5 et 8, les copies étant distantes l'une de l'autre de 1 à 10 caractères :
`mod2() ==> X1:#[5,8], .*:{#[1,10]}, X1.`
`mod2()==*> SEQ1`
Le modèle `mod2()` se lit de la manière suivante : X1 désigne une variable ; toute chaîne constituée de 5 à 8 caractères peut être instance de X1. '`.*`' désigne l'espaceur ('gap'). Il a pour contrainte une taille comprise entre 1 et 10 caractères. Après le gap, on attend une seconde occurrence de X1. Ainsi, **ACUGGC**CCGACUGGC**ACUGGC** est une instance de ce motif sur la séquence d'entrée UUCAG*ACUGGCCCGACUGGCACUGGC*CAC.
Voici une autre façon d'exprimer ce motif :
`mod2() ==> X1:{#[5,8],_IX1}, .*:{#[1,10]}, ?IX1`. Ici, l'instance de X1 est sauvegardée (par _) dans une variable (notée ici IX1). Après un gap de 1 à 10, on souhaite retrouver la même chaîne IX1 (que l'on rappelle par `?`). La deuxième version de `mod2()` est ici inutilement lourde. Elle permet cependant d'introduire une notion qui sera utilisée par la suite, la notion de mémorisation d'instance. En effet, les instances d'une variable ne sont pas forcément des copies exactes (cf la section suivante), et ce procédé de nommage explicite (ici `_IX1`) permet de distinguer les instances entre elles. Plus généralement, ce mécanisme permet de sauvegarder, pour s'y référer plus loin, les instances concrètes de n'importe quelle partie de modèle.

### 3.2 Principaux composants du langage

• **Copies non exactes** Les séquences génomiques évoluent à travers un processus de duplication entrainant des mutations ou des erreurs. Les variations élémentaires (d'un caractère) entre un modèle et son instance sont prises en compte par deux compteurs de coût : le compteur de *substitutions* et le compteur *indel* d'insertion/délétions. En pratique, les substitutions sont définies par une contrainte dite de contenu : `$[m,n]` avec `m` et `n` des entiers. Cette contrainte autorise de `m` à `n` substitutions. Une contrainte de substitution peut également s'exprimer sous la forme d'un pourcentage de substitutions autorisées : `p$[m,n]` dans ce cas, `n` représente le pourcentage maximum de substitutions autorisé (et `m` le minimum). En reprenant l'exemple précédent, on peut donc autoriser une substitution dans la seconde occurrence de X1 en le précisant dans le modèle de la manière suivante : `X1:{#[5,8],_IX1}, .*:{#[1,10]} , ?IX1:$[0,1]` Les indels sont définis de façon similaire, par `$$[m,n]` et `p$$[m,n]`. Ainsi, `"aaaa" ::$$[0,1]` accepte les chaînes `aaaa` (pas d'indel), `aaa` (une délétion) ou `aaaaa` (une insertion).

Voici un exemple pour compléter le propos sur le nommage des instances des variables de chaîne :
`X1:{#[5,8],_IX1},.*:{#[1,7]}, ?IX1: {_IX2}:{$[1,1]}, .*:{#[1,7]}, ?IX2:{$[1,1]}`
Ce modèle permet de chercher trois instances d'une même chaîne qui dériveraient successivement l'une de l'autre (e.g `IX1` = `aaaaa`, `IX2` = `aaaca` et `IX3` = `agaca`). L'individualisation des instances peut ainsi contribuer à traduire la notion d'évolution dans les séquences.

• **IUPAC** L'alphabet *IUPAC* définit des caractères ambigus (*e.g.* R désigne un A ou un G dans l'ADN). L'utilisation de cet alphabet est autorisée dans les modèles Logol, et contribue à la prise en compte de la variabilité élémentaire dans les séquences.

• **Morphismes** Un morphisme est une fonction applicable à une chaîne. Certains morphismes sont prédéfinis. Ainsi `"wc"` transforme une chaîne d'ARN en son complémentaire, en appliquant la complémentarité "Watson Crick" qui transforme A en U, G en C et inversement. Ainsi le motif `("wc" "ACUGGC")` représente la chaîne `"UGACCG"`. Un autre morphisme prédéfini est le morphisme inverse, noté −, qui renverse une chaîne. Ainsi `(-"UGACCG")` représente la chaîne `"GCCAGU"`. Ces deux morphismes combinés permettent de représenter le *complémentaire inverse* d'une chaîne, et donc de modéliser les palindromes biologiques que sont les tiges-boucles. Le motif suivant décrit par exemple une tige-boucle, dont la longueur de la tige varie entre 5 et 11, de la boucle entre 1 et 10, et dont l'appariement "Watson Crick" de la tige n'est pas forcément parfait, pouvant contenir jusqu'à deux substitutions et un indel (*i.e.* une insertion ou une délétion)
`STEM5:{#[5,11],_IS5}, .*:{#[1,9]}, -"wc" ?IS5 :{$[0,2],$$[0,1]}`
Ainsi, le contenu de "STEM5" (la tige aller, *i.e.* située du coté 5' de la séquence) est sauvegardé lors de l'analyse Logol. La partie "STEM3" (la tige retour) a pour contenu le complément inverse de STEM5 précédemment sauvegardé (donc `-"wc" ?IS5`), à deux substitutions et un indel près. L'utilisateur peut par ailleurs définir ses propres morphismes.

• **Compositions des segments** Pour exprimer la composition des séquences, telle que l'hydrophobicité d'une zone de protéines, ou la richesse en GC d'un fragment d'ARN, Logol propose l'expression de "contraintes d'alphabet" qui vérifient le taux de présence de certaines lettres dans la séquence. Ainsi, `X1:{#[2,43]}:{%gc :50}` désigne un segment de 2 à 43 caractères ayant au moins cinquante pour cent de GC.

• **Contraintes de contenu négatives** Les contraintes de contenu négatives permettent d'exclure certaines valeurs dans un motif. Elles s'expriment avec le symbole `!`. Ainsi, `("aaa"| "ttt"), !"ga":{#[2,2]}` désigne une chaîne constituée de cinq caractères, les trois premiers étant trois `a` ou trois `t`, et les deux suivants étant tout sauf le mot `ga`.

• **Répétitions** Les répétitions en tandem, qui sont les copies successives d'une même entité, constituent des structures fréquentes dans les séquences génomiques. Pour modéliser des structures de ce type, Logol propose un constructeur de répétitions, `repeat`, qui gère un compteur d'occurrences. Son format standard est `repeat(<entité>,<distance>)+<nombre d'occurrences>`.
Ainsi par exemple, `repeat("acgt",[0,3])+[7,38]` indique la répétition de l'entité `"acgt"` de 7 à 38 fois, avec un espacement autorisé d'au plus trois caractères entre deux répétitions.
Un autre format possible, `repeat(<entité>;<distance>)+<nombre d'occurrences>` (*i.e.* avec `;` à la place de `,`), indique que les occurrences successives peuvent être chevauchantes.

• **Analyses multiples** Parmi les caractéristiques importantes des séquences biologiques se situe la cohabitation de structures alternatives dans une même séquence. Les chevauchements de gènes, par exemple, sont des configurations fréquentes. Logol permet de modéliser de telles situations, en indiquant les modèles alternatifs au niveau de la règle principale de la grammaire (`==*> SEQ1`). Pour être acceptée par la grammaire, la séquence doit contenir une instance de chaque modèle alternatif.

```
mod1() ==> "YVCPFDGCNK"
mod2() ==> "NKLKSHIL"
mod1().mod2()==*> SEQ1
```

Ainsi, la grammaire ci-dessus accepte des séquences qui contiennent les deux chaînes `"YVCPFDGCNK"` et `"NKLKSHIL"`, indépendemment de leurs positions respectives, chevauchantes ou non. On peut passer des paramètres entre 2 motifs alternatifs, par exemple pour caler les positions respectives de certains éléments.

• **Structuration des modèles** Les modèles, dès lors qu'ils sont un peu complexes, sont amenés à être structurés. Ceci se fait naturellement dans la mesure où le formalisme grammatical est particulièrement adapté à l'écriture de modèles hiérarchiques.

```
mod2()==> repeat(("K"|"R"|"L"),[0,0])+[8,12]
mod1()==>"CVC", .*:{#[3,8]}, mod2()
```

• **Vues et portée des contraintes** Les contraintes (de contenu, de taille...) peuvent être posées sur différentes parties du modèle. On peut les appliquer à des entités élémentaires telles qu'une chaîne ou une variable, comme vu sur les exemples précédents, ou bien à un ensemble d'entités possèdant elles-mêmes leurs contraintes individuelles.
Si l'ensemble des éléments est contigu dans la séquence, on parle de *vue*. Syntaxiquement, la vue est définie par des parenthèses. Sur l'exemple suivant, les instances des trois variables X1,X2,X3 peuvent compter chacune jusqu'à dix caractères, mais la contrainte supplémentaire sur la vue (X1,X2,X3) impose que la totalité du segment soit inférieure à vingt caractères

```
(X1:{#[1,10]}, X2:{#[1,10]}, X3:{#[1,10]}) : {#[8,20]}
```

Il est également possible de poser des contraintes sur une collection d'éléments non contigus du modèle (par exemple sur les deux segments constituant la tige d'une tige-boucle). Ces contraintes sont alors placées dans un module global spécifique (le "panneau de contrôle").

## 4 Analyseur Logol : Quelles fonctionnalités ?

• **Construction du modèle** Le modèle Logol peut se construire de deux façons, grammaticalement via un éditeur de texte[2] ou graphiquement via une interface web[3]. L'interface graphique permet une abstraction de la syntaxe et une visualisation plus intuitive de ce que représente le modèle (les boucles par exemple) tout

en produisant un modèle grammatical. Des exemples des deux types de modèles sont présents dans cet article. L'outil accepte en entrée chacun de ces modèles indifféremment mais l'outil d'analyse offre des fonctionnalités supplémentaires avec le modèle graphique, comme la visualisation d'un match sur le modèle graphique originel.

• **Fonctionnement de l'analyse syntaxique** Le logiciel effectue son analyse en quatre étapes. La première étape convertit le modèle en un script Prolog. en affineant au mieux les recherches en fonction du modèle d'entrée (utilisation des propriétés de taille d'une variable connue *a posteriori* par exemple). La seconde étape découpe les séquences d'entrée, si possible, afin de paralléliser les traitements localement ou sur un cluster. La recherche peut également être exécutée en parallèle sur plusieurs CPU localement. La troisème est l'étape principale, celle d'analyse. Le programme va exécuter le programme Prolog généré pour rechercher le motif dans la séquence donnée en entrée. Cette recherche s'effectue en lecture gauche-droite . Basiquement, deux types de recherche vont être effectués, locale ou distante, pour trouver un élément du motif. La recherche distante est utilisée pour chercher un élément connu (avec ou sans erreurs) dont la position n'est pas connue. Pour cela plusieurs possibilitées sont offertes par le logiciel mais la solution la plus performante est l'utilisation de VMatch[15]. La recherche locale recherche un élement du motif quelconque (connu ou non) à la position courante, de manière récursive. Le programme va sauvegarder dans le contexte d'exécution les éléments du motif au fur et à mesure de ses correspondances avec la séquence jusqu'à ce qu'une équivalence complète soit trouvée. La correspondance est alors enregistrée comme un match. En cas d'échec, un retour arrière est effectué pour tester la condition suivante. La sauvegarde de chacun des éléments permet à la fois de tester des conditions sur des ensembles (condition de taille sur un ensemble d'éléments) et de fournir dans les résultats le détail de la correspondance entre l'élément trouvé et le motif. Une transformation est éventuellement appliquée à l'élément en cours avant de tester sa correspondance (complément inverse, matrice de transformation, ...). Enfin, la dernière étape regroupe les résultats, les reformate éventuellement et compresse les fichiers dans une archive zip.

• **Lancement de l'analyse** Le logiciel peut être exécuté en ligne de commande ou via une interface web. L'interface web permet d'exécuter le programme avec un nombre d'options réduit et propose une liste de banques de données disponibles (*i.e.* les banques accessibles sur la plateforme GenOuest). L'utilisateur peut ainsi lancer la reconnaissance de motif sur des séquences publiques, ou bien fournir sa propre liste de séquences. En ligne de commande, un fichier d'options permet de définir les préférences de l'utilisateur et d'entrer des valeurs par défaut pour ses recherches. De nombreuses options sont disponibles pour affiner la recherche ou limiter le nombre de résultats.

• **Format des résultats** Les résultats fournis par le programme sont par défaut sous la forme de fichiers XML archivés dans un fichier zip. Le programme permet également d'exporter les résultats au format FASTA et GFF. Il y a un fichier XML par séquence d'entrée. Chaque fichier contient la référence de la séquence d'entrée, le modèle utilisé (graphique ou non) et l'ensemble des résultats. Chaque résultat contient le détail de la correspondance avec le motif de départ, c.a.d conserve la même structure que le modèle (boucles, sous-modèles) avec les positions de la correspondance, les erreurs trouvées (nombre de substitutions et d'indels), le sous-motif d'origine... Ces détails permettent à l'utilisateur de réaffiner le modèle au besoin. Une interface web d'analyse permet de lister l'ensemble des résultats pour un fichier, d'afficher sous forme d'arbre un résultat, d'afficher sous forme statistique le nombre de résultats par branche du motif et d'afficher un résultat dans l'éditeur de modèle en colorant le chemin effectué.

## 5 Modélisation des décalages de phase en -1, ou 'frameshift -1'

Le recodage est un processus biologique qui peut intervenir lors de la traduction d'ARN messagers. Il génère une modification de la lecture traditionnelle du message génétique par le ribosome et permet ainsi la production de deux protéines distinctes à partir d'une même séquence d'ARN initiale. Parmi les événements de recodage se situe le "décalage de phase en -1" ("frameshift -1"). Le décalage de phase se fait par glissement du ribosome d'un nucléotide en amont au niveau de la "fenêtre glissante" `X XXY YYZ`. Le premier `X` est alors lu deux fois. Ainsi en phase 0, les codons `ABX XXY YYZ CDE ...` sont décodés par le ribosome alors qu'en phase -1 se sont les codons `ABX` **XXX YYY ZCD ...** qui sont décodés (le premier X est ainsi traduit 2 fois). La structure typique favorisant un événement de frameshift-1, qu'on appellera par la suite "motif F-1", est

représentée en Fig. 1. Elle est constituée successivement : d'un codon start, d'un certain nombre de codons, d'une fenêtre glissante (un heptamère) `XXXYYYZ` situé en phase -1, d'un "espaceur" de quelques nuléotides, et d'une structure secondaire. La structure secondaire est l'obstacle sur lequel vient buter le ribosome pendant qu'il traduit l'heptamère, et qui peut alors le faire reculer d'un nucléotide, provoquant le décalage de phase. La stucture secondaire typique est un "pseudo-nœud de type H", constitué de deux tiges-boucles imbriquées.

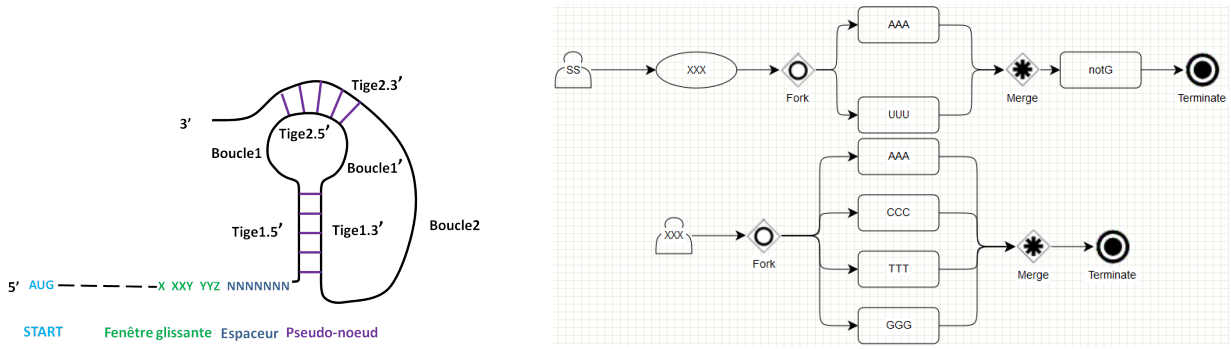


**Figure 1.** Fig1.A : Struture typique, appelée ici "motif F-1", favorisant un décalage de phase en -1. Fig1.B : modèle graphique Logol pour la fenêtre glissante du motif F-1

De nombreux outils existent pour tenter de détecter de potentiels sites de frameshift-1 [10], mais la détection reste un sujet de recherche actif, car le motif F-1 n'est pas universel (les caractéristiques de l'heptamère, du spacer, de la structure secondaire ne sont pas identiques d'un organisme à l'autre) et la détection de pseudo-nœuds est un problème difficile. Beaucoup de ces méthodes procèdent par filtres successifs, comme le fait par exemple KnotInFrame[25], un des outils le plus avancé sur le sujet. KnotInFrame détecte dans un premier temps tous les heptamères `XXXYYYZ`. Ensuite, il recherche de potentiels pseudo-nœuds en aval de ces motifs, au moyen d'une procédure de repliement d'ARN élaborée à cet usage, "pknotsRG-fs".

## 5.1 Modèle Logol

La complexité du motif F-1 en fait un bon candidat pour explorer l'expressivité de Logol. Sa modélisation nécessite d'utiliser un certain nombre de principes du langage tels que la multi-analyse, les contraintes de contenu négatif, les répétitions de motifs ou la recherche de palindromes biologiques. Nous en détaillons ici les éléments importants.

**5.1.1 Calage de phase et multi-analyse de deux ORF chevauchants** Parmi les caractéristiques structurelles indispensables à la survenue d'un événement de frameshift-1, certaines concernent le calage de phase. La traduction standard, "en phase 0", doit s'effectuer sur une séquence possédant une "cadre ouvert de lecture" : un codon start (`AUG`), suivi d'un certain nombre de codons (groupe de trois nucléotides qui va être traduit en un acide aminé), puis d'un codon stop (`UGA`,`UAG` ou `UAA`) qui arrête la traduction. L'ensemble des codons à traduire ne doit pas contenir le codon stop. La traduction avec frameshift-1 quant-à elle démarre sur le même start, puis à partir de la fenêtre glissante recule d'un nucléotide, ce qui conduit à continuer à avancer trois par trois mais "en phase-1", avant de s'arrêter sur un codon stop, situé de ce fait en phase -1. Pour qu'une séquence d'ARN ait le potentiel pour engendrer un événement de frameshift-1, il doit donc s'y trouver d'une part un cadre ouvert de lecture (un start suivi d'un nombre suffisant de codons non-stop puis un stop) et d'autre part un recalage de phase en -1 (le même start suivi à une distance suffisante d'un stop en phase -1).
Cette vérification correspond à faire en Logol une *analyse multiple* pour reconnaître les deux motifs alternatifs, "ORF" et "ORFminus", avec un *passage de paramètre* entre les deux modèles pour les caler sur le même start. Le modèle ORF contient ainsi un *repeat* qui accepte jusqu'à 300 codons non stop (les valeurs numériques sont inspirées de la littérature), soit : `repeat(notstop(),[0,0])+[0,300]` où le modèle `notstop` est un modèle, construit à partir d'une *vue*, qui accepte une chaîne de taille 3 qui ne soit pas un `stop`, pour cela une *contrainte de contenu négative* est posée sur la vue. Certains éléments du modèle graphique sont présentés en Fig. 2.

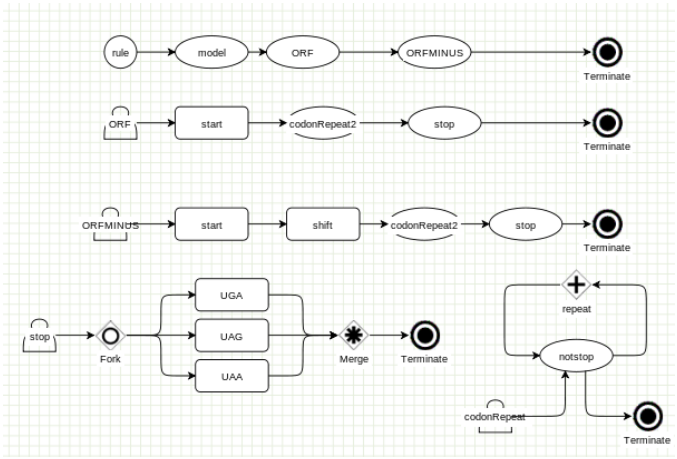


**Figure 2.** Modèle Logol graphique de calage de trames : 2 motifs alternatifs ORF et ORFminus ; existence de stops en phase 0 et en phase -1

**5.1.2 Fenêtre glissante et espaceur** Le motif F-1 contient trois compartiments principaux : la fenêtre glissante, l'espaceur et la structure secondaire. Le modèle Logol proposé pour la *fenêtre glissante* respecte le consensus établi : c'est un motif héptamérique de la forme XXXYYYZ, qui doit être positionné en phase-1, où X est un nucléotide quelconque répété 3 fois, Y est la base A ou U répétée trois fois, et Z est différent de G (cf Fig. 1.B). L'*espaceur* est réalisé par un simple gap (de taille ici inférieure à dix).

**5.1.3 Pseudo-nœuds** Comme évoqué précédemment la structure secondaire la plus efficace pour les frameshift-1 est le pseudo-nœud de type H (deux tiges-boucles imbriquées, cf Fig. 1.A) , même si elle n'est pas la seule possible (elle consiste parfois en une simple tige). C'est donc celle que nous avons modélisée ici.
Une vision graphique des différentes parties d'un pseudo-nœud est donnée en figure Fig. 3. `STEM15` désigne la tige 1 aller (côté 5'), `STEM13` désigne la tige 1 retour (côté 3'). `STEM25` et `STEM23` désignent les deux éléments de la deuxième tige. `L1A` ,`L1B` et `L2` sont les différents éléments des boucles.
Une première modélisation a été réalisée suivant cette structure de base, avec les valeurs numériques de la grammaire Logol suivante

```
STEM15:{#[4,16],_IS15},.*:{#[1,5]}, STEM25:{#[3,8],_IS25},.*:{#[0,4]},
      -"wc" ?IS15 :{$[0,4]},.*:{#[4,40]},-"wc" ?IS25 :{$[0,2]}
```

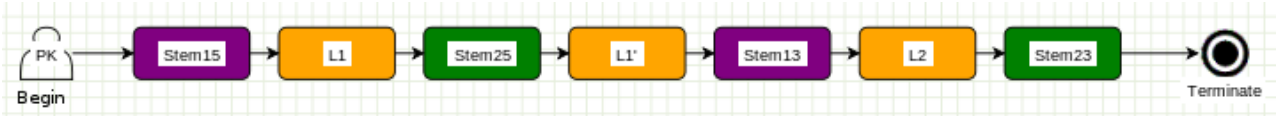


**Figure 3.** Premier modèle Logol pour représenter un pseudo-nœud

Notre démarche de validation de ce modèle sur des données réelles (cf section 5.2), nous a amené à le raffiner considérablement (cf Fig. 4). Le modèle final utilise une grande variété d'éléments de langage de Logol. Nous avons ainsi introduit la prise en compte du pourcentage de GC dans les tiges, la dissociation des deux nucléotides aux extrémités des tiges -pour interdire des mismatchs à cet endroit, ou encore l'acceptation de l'appariement non canonique "wobble" (G-U) dans certaines parties de tiges [21] (dans notre modèle, l'appariement "wcw" est utilisé aux extrémités de tige et "wc" en partie centrale).

## 5.2 Une première validation du modèle

Pour pouvoir tester la pertinence de notre modèle de frameshift-1, et l'affiner, nous avons élaboré un jeu de séquences de test[21]. Ce jeu est constitué en premier lieu de séquences connues pour produire des événements de frameshift-1. Nous avons choisi pour cela les trente séquences avérées ("validated -1 frameshift") de la

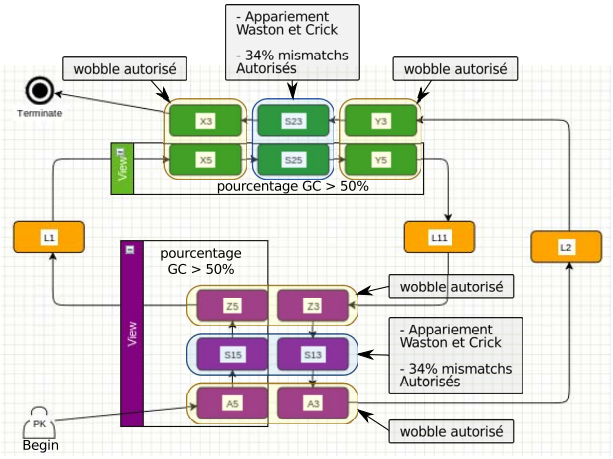


**Figure 4.** Modèle Logol final pour représenter le pseudo-nœud du Motif F-1

base de référence Recode2 *http ://recode.genetics.utah.edu*. Le jeu de données est complété par des séquences aléatoires obtenues ainsi : chacune des trente séquences du jeu de données de référence est aléatoirement mélangée cent fois par le logiciel Shuffleseq *http ://emboss.bioinformatics.nl/cgi-bin/emboss/shuffleseq*, ce qui a pour intérêt de conserver les longueurs de séquence et un pourcentage nucléotidique constant pour chaque set constitué. On dispose ainsi de trente séquences "positives" (dans lesquelles on tente de retrouver, au bon emplacement, un motif F-1) et trois cents séquences "négatives" (dans lesquelles on souhaite retrouver le moins possible d'instances du motif F-1). Ces travaux de validation [21] nous ont amené à effectuer des comparaisons entre les prédictions de pseudo-nœuds faites par Logol et celles réalisées par "DotKnot", un logiciel de repliement de structures nouées *http ://dotknot.csse.uwa.edu.au/*, et ils ont permis de faire évoluer de façon importante notre modélisation des pseudo-nœuds, en ajustant les paramètres : appariement wobble, pourcentage de gc et taux de mésappariement acceptés dans les tiges (cf section 5.1.3). Au final, sur les séquences de référence de Recode2, le modèle Logol trouve une centaine de matchs par séquence, parmi lesquels se situe généralement le match Recode2. En appliquant a posteriori un calcul de score (sur la qualité des tiges) pour trier ces matchs, on localise la zone de frameshift officiel dans 20 cas sur 30.

• **Temps de calcul** Pour donner un ordre d'idée des temps de calcul, l'analyse de la plus grosse séquence de référence (30K) avec le modèle Logol final dure 1mn30s (sur Intel X5550, 144Go RAM) - versus réponse immédiate sur le site KnotinFrame. L'analyse du génome complet de Bacillus subtilis (`str 168 NC_000964.3`, 4215606 bp) produit 7000 matchs en 2h. KnotinFrame ne peut pas analyser cette séquence.

## 6 Conclusion

Logol a été conçu pour initier la construction d'un outil de reconnaissance de motifs générique qui permette d'exprimer et de rechercher dans les séquences moléculaires les structures existantes typiques du mécanisme du vivant. C'est ainsi qu'après avoir développé un pemier outil, Stan[18], et d'en avoir perçu les limites, nous avons posé les bases d'un nouveau langage expressif et évolutif autour du principe des grammaires à variables de chaînes. Logol est actuellement opérationnel (il est par exemple utilisé pour rechercher les tiges-boucles des CRISPRS pour alimenter la base CRISPI[26]), et accessible sur la plateforme Genouest[1]. Le fait que le langage s'avère assez bien adapté pour traduire le modèle complexe des structures de frameshift-1, alors qu'il n'a pas été conçu à cette intention, nous semble un signe encourageant sur la généricité des éléments de langage de Logol. Ceci n'exclut pas que Logol reste un langage ouvert, amené à s'adapter aux nouveaux besoins d'expressivité, car sa finalité est de permettre d'exprimer des motifs complexes et réalistes, tels qu'ils sont découverts par les biologistes. Parmi les ajouts majeurs à envisager, afin de pouvoir prendre en compte l'extrême variété des interactions moléculaires possibles au sein de la cellule, se situe la prise en compte dans un même modèle de différents types de données (génomes, ARN et protéines) d'un même organisme.

Logol est accessible sur le site internet de la plateforme bioinformatique Genouest [1].

• **Remerciement** à la fondation Rennes1 qui a financé un "semestre pour l'innovation" à C.B. en 2011.

## Références


[1] URL - Genouest bioinformatics platform : http ://www.genouest.org/.

[2] URL - Logol Designer and language tutorial : http ://training.genouest.org/claroline/claroline/learnpath/navigation/viewer.php.

[3] URL - Logol Designer graphical interface : http ://webapps.genouest.org/logoldesigner/.

[4] C. BELLEANNÉE et J. NICOLAS : Logol : Modelling evolving sequence families through a dedicated constrained string language. Research Report 6350, INRIA, 11 2007.

[5] B. BILLOUD, M. KONTIC et A. VIARI : Palingol : a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res*, 24(8), avr. 15 1996.

[6] E. de CASTRO, C. J. A. SIGRIST, A. GATTIKER, V. BULLIARD, P. S. LANGENDIJK-GENEVAUX, E. GASTEIGER, A. BAIROCH et N. HULO : Scanprosite : detection of prosite signature matches and prorule-associated functional and structural residues in proteins. *Nucleic Acids Research*, 34(suppl 2):W362–W365, july 2006.

[7] S. DONG et D. B. SEARLS : Gene structure prediction by linguistic methods. *Genomics*, 23(3):540–551, 1994.

[8] M. DSOUZA, N. LARSEN et R. OVERBEEK : Searching for patterns in genomic data. *Trends in Genetics*, 13(12): 497–498, dec 1997.

[9] S. EDDY : Rnabob : a program to search for rna secondary structure motifs in sequence databases. 1996.

[10] A. E. FIRTH, M. BEKAERT et P. V. BARANOV : Computational resources for studying recoding. *In* J. F. ATKINS et R. F. GESTELAND, éds : *Recoding : Expansion of Decoding Rules Enriches Gene Expression*, vol. 24 de *Nucleic Acids and Molecular Biology*, p. 435–461. Springer New York, 2010.

[11] A. GATTIKER, E. GASTEIGER et A. BAIROCH : Scanprosite : a reference implementation of a prosite scanning tool. *Applied Bioinformatics*, 1(2):107–108, 2002.

[12] S. GRAF, D. STROTHMANN, S. KURTZ et G. STEGER : HyPaLib : a Database of RNAs and RNA Structural Elements defined by Hybrid Patterns. *Nucleic Acids Res.*, 29(1):196–198, 2001.

[13] K. JENSEN, G. STEPHANOPOULOS et I. RIGOUTSOS : Biogrep : A multi-threaded pattern matcher for large pattern sets. 2002.

[14] A. K. JOSHI, K. VIJAY-SHANKER et D. WEIR : The convergence of midly context-sensitive grammars. *In* S. M. SHIEBER et T. WASOW, éds : *The Processing of Natural Language Structure*, p. 31–81. MIT Press, Bosto, MA, 1991.

[15] S. KURTZ : The vmatch large scale sequence analysis software.

[16] T. J. MACKE, D. J. ECKER, R. R. GUTELL, D. GAUTHERET, D. A. CASE et R. SAMPATH : Rnamotif, an rna secondary structure definition and search algorithm. *Nucleic acids research*, 29(22):4724–4735, nov 2001.

[17] F. MEYER, S. KURTZ, R. BACKOFEN, S. WILL et M. BECKSTETTE : Structator : fast index-based search for rna sequence-structure patterns. *BMC Bioinformatics*, 12(1):214, 2011.

[18] J. NICOLAS, P. DURAND, G. RANCHY, S. TEMPEL et A.-S. VALIN : Suffix-tree analyser (stan) : looking for nucleotidic and peptidic patterns in chromosomes. *Bioinformatics*, 21(24):4408–4410, 2005.

[19] G. PESOLE, S. LIUNI et M. DSOUZA : Patsearch : a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics*, 16(5):439–450, 2000.

[20] J. REEDER, J. REEDER et R. GIEGERICH : Locomotif : from graphical motif description to rna motif search. *Bioinformatics*, 23(13):i392–i400, 2007.

[21] A. ROCHETEAU et C. BELLEANNÉE : Recherche d'éléments structurés dans les génomes par modèles logiques. Rapport de recherche PI-1994, Dyliss - Inria - Irisa, avr. 2012.

[22] D. B. SEARLS : String variable grammar : A logic grammar formalism for the biological language of DNA. *Journal of Logic Programming*, 24(1 & 2):73–102, 1995.

[23] D. B. SEARLS et S. DONG : A syntactic pattern recognition system for DNA sequences. *In* C. R. CANTOR, H. A. LIM, J. FICKETT et R. J. ROBBINS, éds : *Proceedings 2nd International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, p. 89–101. World Scientific, 1993.

[24] D. STROTHMANN, S. A. GRÄF, S. KURTZ et G. STEGER : The syntax and semantics of a language for describing complex patterns in biological sequences. Rap. tech., Universität Bielefeld, Technische Fakultät, Arbeitsgruppe Praktische Informatik, août 2000.

[25] C. THEIS, J. REEDER et R. GIEGERICH : Knotinframe : prediction of -1 ribosomal frameshift events. *Nucleic Acids Research*, 36(18):6013–6020, 2008.

[26] C. VROLAND, C. BELLEANNÉE et J. NICOLAS : Recherche et annotation des structures de CRISPR dans l'ensemble des génomes procaryotes. Rapport de recherche PI-1986, Symbiose - Inria - Irisa, nov. 2011.