# Recherche d'éléments structurés dans les génomes par modèles logiques

Antoine Rocheteau, Catherine Belleannée

**Publications Internes de l'IRISA**
**ISSN : 2102-6327**
**PI 1994 – mars 2012**

# Looking for structured elements in the genomes with logical models

Antoine Rocheteau, Catherine Belleannée *
*Catherine.Belleannee@irisa.fr*

**Abstract:** Recoding is a biological process that allows an alternative reading of the genetic information. It results in producing two distinct proteins in different proportions from a same messenger RNA sequence. This study focuses on "-1 ribosomal frameshift", which is a sub-class of recoding. The shift takes place within a specific part of the RNA sequence called "slippery side" and is initiated by a stable RNA secondary structure like a pseudo-knot. This structure induces a pause of the ribosome during the translation and thus cause the shift. We developed a method for detecting -1 frameshift events based on the use of a pattern matching tool, Logol. Logol has been conceived by our team, Dyliss, in collaboration with the GenOuest bioinfomatics platform. Some successive Logol models have been designed in order to properly figure the frameshift signals. The resulting model can detect with a good accuracy the -1 ribosomal frameshift signals of a reference set. The approach is original and is distinguished by the fact that there is no real use of a minimum free energy calcul for ranking the secondary structures (that is pseudo-knots) according to their stability. Using in depth the Logol tool led to propose new features to improve the modeling language. This study also validates the use of Logol as a mean to help finding complex biological patterns.

**Key-words:** Recoding, Programmed -1 Ribosomal Frameshift Signal, Logic Model, RNA Secondary Structure, Free Energy, Pattern Matching, Biological Sequence Modelling, Pseudo-Knot, Complex Pattern

---

## *Recherche d'éléments structurés dans les génomes par modèles logiques*

**Résumé :** *Le recodage est un processus biologique qui permet une lecture alternative du message génétique. Ainsi, il permet la production de deux protéines distinctes en proportions différentes à partir d'une unique séquence d'ARN messager. Les travaux décrits ici concernent le "décalage de cadre de lecture en -1", dit "frameshift -1", qui est une des situations de recodage. Dans ce type d'événements, le décalage s'opère au sein d'une zone particulière de la séquence d'ARN en train d'être traduite, qu'on appelle "fenêtre glissante", et il est provoqué par une structure secondaire stable de l'ARN, de type pseudo-noeud, présente en aval de la fenêtre glissante. Le ribosome, en train d'effectuer la traduction, vient buter sur la structure, et recule alors d'un nucléotide. Nous avons élaboré une méthode de détection des événements de frameshift-1 basée sur l'utilisation de Logol, un outil de reconnaissance de motifs (pattern matching) développé par notre équipe Dyliss en collaboration avec la plateforme bioinformatique GenOuest. Des modèles Logol ont été élaborés et affinés afin de fournir une modélisation la plus fidèle possible des différents compartiments qui composent le "motif de frameshift-1", ce qui a permis au final de détecter avec une bonne précision les événements de décalage de phase en -1 présents dans un jeu de référence. L'approche est originale et se démarque par le fait qu'il n'y ait pas de véritable calcul de stabilité énergétique des structures secondaires recherchées (les pseudo-noeuds). Ces travaux ont amené à utiliser Logol de façon approfondie. Cela a contribué à le faire évoluer, par l'ajout de nouveaux éléments de langage utiles à la modélisation de séquences biologiques. Cela a également contribué à valider l'utilisation de cet outil générique au service de la recherche de motifs biologiques complexes.*

**Mots clés :** *Recodage, Décalage de cadre de lecture en -1, Modèle logique, Structure secondaire d'ARN, Energie libre, Reconnaissance de motifs, Modèlisation de séquences biologiques, Pseudo-noeuds, Motif complexe*

---

Travaux réalisés avec le soutien de la fondation Rennes1, par le financement d'un "semestre pour l'innovation"à C.B.
* université de Rennes 1, projet Dyliss Irisa/Inria

# I. Introduction

Le recodage est un processus biologique qui intervient lors de la traduction de l'ARN messager. Il implique une modification de la lecture traditionnelle du message génétique par le ribosome[1] et permet ainsi la production de deux protéines distinctes à partir d'une même séquence d'ARN initiale. Parmi les événements de recodage, le décalage de phase ou « *frameshift* » agit en provoquant un décalage du cadre de lecture. Ce processus est largement répandu puisqu'on le retrouve par exemple à la jonction entre les gènes gag et pol présent chez le virus du sida, ou leurs équivalents, chez la plupart des rétrovirus, dans des virus de plantes, de levure, des éléments transposables bactériens ou eucaryotes[2], ce qui représente un intérêt pour la compréhension des mécanismes de transcription et peut se révéler intéressant dans l'optique de recherche de nouvelles cibles thérapeutiques. Les travaux présentés ici[1] concernent la détection des *frameshifts -1,* qui provoquent un décalage du cadre de lecture d'un nucléotide en amont. Le mécanisme du décalage de phase a été étudié mais il reste difficile à identifier dans les séquences d'ARN du fait de la présence de structures secondaires nouées telles que les pseudo-nœuds. D'un point de vue technique, nous avons utilisé Logol[3], qui est un outil développé dans l'équipe Dyliss (INRIA – IRISA), dans laquelle est se sont effectuées ces recherches. Logol est un outil dit de « *pattern matching* » qui permet de décrire des modèles expressifs adaptés pour la recherche de motifs complexes. L'objectif de cette étude est double, il s'agit d'une part de mettre au point une méthode de détection des événements de *frameshift -1* dans les séquences d'ARN, et d'autre part de consolider Logol. En effet, le fait d'expérimenter Logol sur une application pointue a pour visées de contribuer à son évolution, en proposant de nouvelles fonctionnalités et éléments de langages, et de valider sa pertinence dans l'aide à la détection de motifs biologiques complexes.

# II. Matériels & Méthodes

Pour commencer, nous décrivons le matériel de base de cette étude. Nous détaillons tout d'abord les processus biologiques de recodage et plus précisément le *frameshift* -1. Dans un second temps, nous présentons l'outil Logol que nous avons utilisé pour mettre en œuvre la détection des événements de *frameshift* -1 ainsi que l'environnement technologique dans lequel s'effectue les travaux. Nous continuons par l'analyse de l'existant sur les *frameshift* -1 qui concerne à la fois les données de référence, les différents éléments et informations exploitables des modèles établis ainsi que les méthodes de détection de *frameshift* -1 précédemment développées. Nous détaillons ensuite la méthodologie que nous avons élaborée avec la constitution de notre jeu de données de référence, la réalisation d'un modèle de base ainsi que la méthode de validation des résultats.

## II.1. Recodage et *frameshift* -1

Les événements de recodage interviennent lors de la traduction de l'ARN messager en protéine au niveau du ribosome. Ces événements lorsqu'ils se déclenchent vont venir modifier la traduction et la nature des protéines produites. Ils peuvent être divisés en trois catégories :

- La translecture (*readthrough*) permet de continuer la synthèse protéique alors qu'un codon stop (UAG) est identifié. La protéine produite sera plus longue que prévue puisque le codon stop ne sera pas traduit en tant que tel.
- Le saut de ribosome (*ribosom hopping*): il s'opère lorsque le ribosome ignore une partie de l'information traductible en se déplaçant le long de l'ARNm sans décoder la séquence correspondante. Deux sites sont nécessaires: un site de « décollage » et un site « d'atterrissage » qui correspondent à deux codons identiques séparés par une portion de séquence de type boucle[1].

1 Une grande part des travaux a été réalisée lors d'un stage de M1 Modélisation de Systèmes Biologiques en 2011

- Le décalage de cadre de lecture ou *frameshift*, permet également la production de deux protéines à partir d'une unique séquence initiale. Le cadre de lecture peut être décalé en amont ou en aval selon le type de *frameshift* en question. Actuellement quatre catégories de *frameshift* ont été observées : les *frameshift* -1, -2, +1 et +2. Le plus documenté d'entre eux est le *frameshift -1* (figure 1 d'après Giedroc et Cornish 2009 [8]) qui provoque un décalage du cadre de lecture d'un nucléotide en amont. Leur identification est un sujet de recherche actif pour la compréhension des mécanismes sous-jacents. Ce sont ces *frameshift -1* qui font l'objet de notre étude. Leurs sites de décalage de phase sont le plus souvent composés d'une séquence glissante de sept nucléotides (appelée également heptamère, fenêtre glissante, *slippery side*) et d'un élément stimulateur positionné en aval[4] généralement de type pseudo-nœud. Ces deux éléments sont séparés par une courte portion d'ARN: l'espaceur (*linker, spacer*). L'élément stimulateur agit comme un blocage mécanique temporaire au niveau du ribosome qui effectuera une pause ce qui va permettre le décalage lors de la traduction[5].

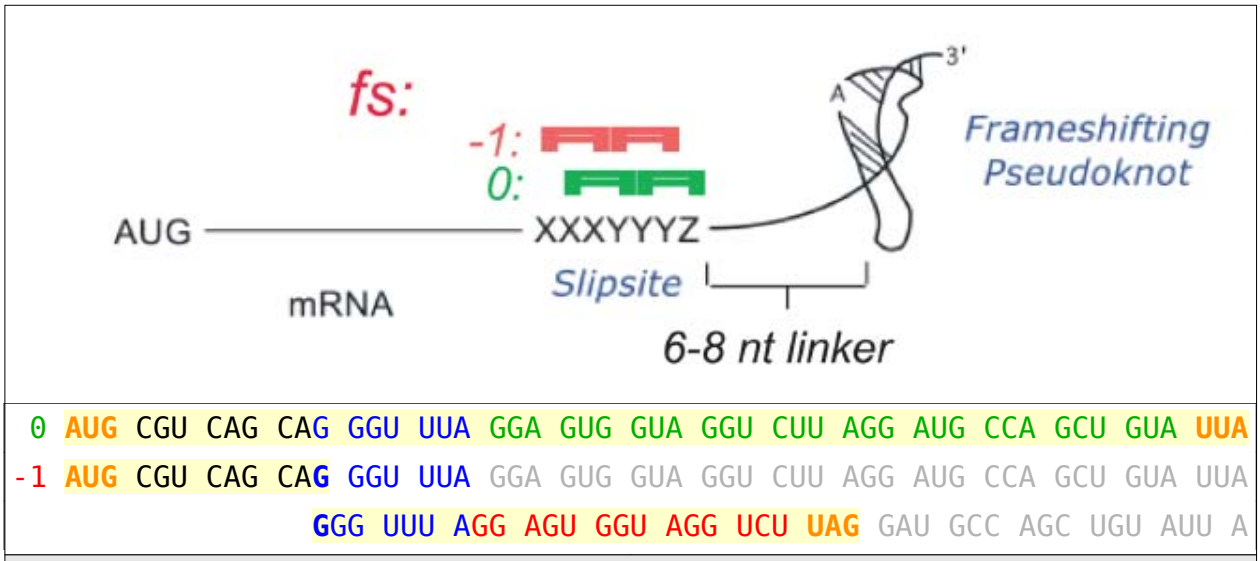



**Figure 1:** *Représentation du fonctionnement d'un événement de frameshift -1*

Deux protéines différentes sont ainsi productibles par ce mécanisme. Elles auront une première partie commune jusqu'au site de décalage mais ensuite, en phase -1, lorsque le décalage s'opère, ce qui n'est pas systématique, un nucléotide sera traduit deux fois[6] (**G** dans l'exemple). Une première fois comme en phase 0 en tant que troisième nucléotide d'un codon (CA**G** dans l'exemple) et une seconde fois en tant que premier nucléotide du codon suivant (**G**GG). Lorsque le *frameshift* intervient, la séquence traduite ne sera dès lors plus la même puisque le cadre de lecture a « glissé » d'un cran en amont et le codon stop en phase 0 ne sera plus décodé à cette position, c'est un codon stop calé en phase -1 qui marquera la fin de la traduction (qui intervient avant dans cet exemple). Les deux protéines productibles par ce mécanisme auront une taille et une composition en acides aminés différentes à partir du site de décalage. Le ratio des deux protéines produites dépend de la fréquence à laquelle le décalage intervient.
La structure initiatrice peut être une simple tige-boucle ou une structure nouée comme le sont les pseudo-noeuds[7]. Plusieurs conformations de pseudo-nœuds sont répertoriées[8], ce qui représente une variété et un polymorphisme important pour lequel il est actuellement délicat de concevoir un modèle consensus pour détecter tous les sites de *frameshift -1*.
Nous nous sommes concentrés sur les *frameshift -1* initiés par la catégorie la plus typique de structure stimulatrice: les pseudo-nœuds de type H[9,10] (ABAB). Un pseudo-nœud H-type se

compose, conformément à la figure 2 (d'après Yann Ponty 2010), de deux tiges (A et B) entrelacées par deux boucles[11] L1 et L2, la troisième, L1', étant facultative.

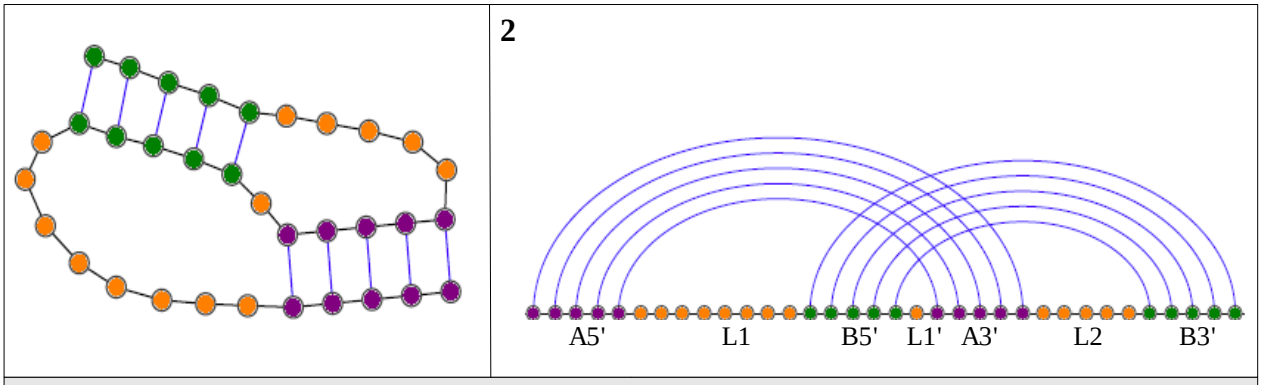



**Figure 2:** *Pseudo-noeud de type H* (**1**) *et sa représentation linéaire* (**2**)

## II.2. Logol

Logol est un logiciel développé dans l'équipe de recherche en bioinformatique Symbiose(INRIA – IRISA), devenue Dyliss, en collaboration avec la plateforme bioinformatique GenOuest. C'est un outil de *pattern matching* au même titre que RNAmotif[12], PROSITE[13] ou STAN[14] (qui est le prédécesseur de Logol également développé par l'équipe Symbiose - ex Dyliss- en 2005). Logol désigne à la fois un langage de modélisation et un analyseur syntaxique mettant en œuvre la recherche d'instance du modèle Logol. Ainsi, Logol permet de décrire des modèles composés de motifs complexes et de rechercher les instances de ces modèles dans des séquences biologiques (ADN, ARN, ou protéines). Logol, disponible sur la plateforme GenOuest http://www.genouest.org/, continue à évoluer, notamment lorsque des travaux de recherche font émerger de nouveaux besoins en terme d'expressivité du langage ou de fonctionnalité de l'outil.

a) Langage Logol

Le langage Logol est basé sur les grammaires à variables de chaînes (grammaires SVG) définies par D.Searls[15]. Dans le cadre de la théorie des langages, les grammaires SVG sont situées entre les grammaires algébriques et contextuelles. Si l'expression de motif ou de gap est accessible dès le niveau des langages réguliers (utilisés par exemple par PROSITE[13]), les grammaires SVG permettent de plus l'expression des palindromes accessibles à partir des langages algébriques, ainsi que la notion de répétition qui nécessite quant-à elle une expressivité supérieure aux langages algébrique. Les grammaires SVG se situent ainsi dans la classe des grammaires dites « faiblement contextuelles » (*mildly context-sensitive grammars*).

Logol a permis d'ajouter de nouveaux opérateurs aux éléments de base des grammaires SVG, pour rendre le langage le plus adapté possible à l'expression de motifs biologiques réalistes et donc complexes. On peut ainsi exprimer en Logol la notion de répétition d'un même motif, de morphisme (tel que la complémentarité inverse) ou de composition d'une chaîne (exemple: pourcentage en GC contenu dans une portion de séquence). Logol autorise aussi les motifs approchés avec la possibilité d'autoriser des substitutions et insertions/délétions.

Voici un exemple de modèle Logol destiné à la recherche de deux copies d'une même portion de séquence séparées par une autre portion de séquence (figure 3a).

```
X:{#[5,8],_valX},GAP:{#[1,10]},X2:{?valX}
```

**Figure 3(a):** *Modèle Logol - Copie et gap*

Cela se traduit en Logol de la manière suivante: la première variable définie est nommée X. Elle se compose d'au moins 5 nucléotides et n'excède pas 8 nucléotides. Elle est sauvegardée dans une nouvelle variable puisque l'on souhaite la réutiliser plus tard (notée ici valX). Une autre variable (GAP) est définie pour l'espaceur. Il a pour contrainte une taille comprise entre 1 et 10 nucléotides (on ne le réutilise pas par la suite donc il est inutile de le sauvegarder). Enfin, on souhaite une seconde occurrence de X que l'on obtient en appliquant sur une nouvelle variable (X2) une contrainte, non plus de taille cette mais de contenu, composée de la portion de séquence X, précédemment sauvegardée dans la variable valX.
Exemple d'instance vérifiant ce modèle : ACUGGC CCGACUGGC ACUGGC

*Voici un inventaire non exhaustif des contraintes exprimables en Logol (figure 3(b à h)).*

- Substitutions:

Elles sont définies par une contrainte dite de contenu : **$[*m*, *n*]** avec ***m*** et ***n*** des entiers. La contrainte autorise dans le cas présent de ***m*** à ***n*** substitutions. Cette contrainte est également exprimable sous la forme d'un pourcentage de substitutions autorisées pour la variable : **p$[*m*, *n*]**, où ***m*** représente le pourcentage minimum de substitutions autorisées et ***n*** le maximum.
En reprenant l'exemple précédent, on peut autoriser une substitution dans la seconde occurrence de X en le précisant dans le modèle de la manière suivante :

```
X:{#[5,8],_valX},GAP:{#[1,10]},X2:{?valX}:{$[0,1]}
```

**Figure 3(b):** *Modèle Logol - Contrainte substitution*

Instance du modèle : ACUGGC CCGACUGGC AUUGGC

- Indels:

De la même manière, les insertions/délétions sont définies par une autre contrainte de contenu dite de « distance » : **$$[*m*, *n*]** pour la notation avec des entiers, **p$$[*m*, *n*]** pour la notation en pourcentage.

```
X:{#[5,8],_X'},GAP:{#[1,10]},X2:{?X'}:{$$[0,1]}
```

**Figure 3(c):** *Modèle Logol - Contrainte insertion/délétion*

Instance du modèle : ACUGGC CCGACUGGC ACUGC

Les contraintes peuvent naturellement être combinées :

```
X:{#[5,8],_X'},GAP:{#[1,10]},X2:{?X'}:{$[0,1],$$[0,1]}
```

**Figure 3(d):**
*Modèle Logol - Contraintes combinées substitution et insertion/délétion*

Ce qui permet dans ce cas de retrouver : ACUGGC CCGACUGGC AUUGC

- Composition en GC

L'expression de contenu nucléotidique dans une portion de séquence se définit en Logol par l'intermédiaire d'une contrainte structurelle dite d'alphabet. Pour exprimer un pourcentage d'un ou plusieurs caractères, il faut suivre la syntaxe suivante: **%''*string*'': *n***, où ***n*** est un entier représentant le pourcentage minimum autorisé pour la variable à laquelle cette contrainte est appliquée.

```
X:{#[5,8],_X'},GAP:{#[1,10]},X2:{?X'}:{%'gc':50}
```

**Figure 3(e):**
*Modèle Logol - Contraintes combinées (substitution / insertion - délétion)*

1. ACUGGC CCGACUGGC ACUGGC
2. ACUAAC CCGACUGGC ACUAAC

La chaîne 1 respecte les contraintes de tailles des différentes variables et le pourcentage en GC présent dans X est supérieur à 50%, ce qui fait de cette chaîne une instance du modèle. En revanche ce n'est pas le cas de la chaîne 2 pour laquelle la portion relative à X possède un pourcentage en GC inférieur à 50%

- Gap:

Un gap est exprimé ici par l'intermédiaire d'une variable qui a pour seule contrainte une limite de taille **#[i, j]**. Cette variable a au minimum **i** nucléotides et au maximum **j** nucléotides.

- Morphisme:

Il s'agit d'une fonction applicable à une chaîne. Un exemple typique d'utilisation de morphisme est la définition et l'expression de la complémentarité inverse. Il est défini en deux temps. Tout d'abord la complémentarité qui relie un A avec un U, un G avec un C et inversement pour ce qui est des liaisons standards (« Watson Crick ») présentes dans l'ARN. Ce morphisme est prédéfini dans Logol et directement utilisable par l'utilisateur. Il se note « **wc** ».
Une variante intéressante de ce morphisme existe. Elle s'opère en combinant un second morphisme; le morphisme inverse qui se note« - », placé devant « **wc** ». On obtient ainsi la complémentarité inverse, ce qui peut se révéler intéressant pour la recherche de tiges-boucles par exemple. L'utilisateur peut par ailleurs définir ses propres morphismes.

Morphisme combiné: ACUGGC $\xrightarrow{wc}$ UGACCG $\xrightarrow{-}$ GCCAGU

```
X:{#[5,8],_valX},GAP:{#[1,10]},"wc" X2:{?valX}
```

**Figure 3(f):** *Modèle Logol - Morphisme complément*

Instance du modèle: ACUGGC CCGACUGGC GCCAGU

```
X:{#[5,8],_valX},GAP:{#[1,10]},-"wc" X2:{?valX}:{$[0,1]}
```

**Figure 3(g):** *Modèle Logol - Morphisme complémentaire inverse*

Instances du modèle :
ACUGGC CCGACUGGC GCCAGU
ACUGGC CCGACUGGC GCUAGU

- Vues *(View)*

Les Vues permettent de grouper des variables consécutives afin d'y appliquer des contraintes. Il est ainsi possible de définir une contrainte de taille ou de pourcentage en GC pour un ensemble de variables alors qu'elles-mêmes possèdent leurs propres contraintes individuelles. Le modèle suivant se compose de trois variables X,Y et Z qui ont une contrainte de taille comprise entre 1 et 10 nucléotides. Ces trois variables sont rassemblées dans une vue à laquelle on applique une contrainte de taille qui concerne l'ensemble des trois variables (dans cet exemple entre 8 et 20 nucléotides).

```
(X:{#[1,10]},Y:{#[1,10]},Z:{#[1,10]}) : {#[8,20]}
```

**Figure 3(h):** *Modèle Logol - Utilisation des vues*

La vue est définie par des parenthèses. L'ensemble du segment à l'intérieur de la parenthèse est concerné par les contraintes appliquées à la vue.
Pour cet exemple, l'utilisation de la vue permet de limiter le nombre d'instances du modèle, en éliminant des combinaisons de X,Y et Z qui seraient trop courtes ou trop longues.

1. ACGU UG GUUA
2. A CG UGGU
3. ACGU UGGUUA GCCAAAUGUCU

Les trois portions de séquences ci-dessus sont conformes aux contraintes de taille de chacune des variables X, Y et Z puisqu'elles sont toutes comprises entre 1 et 10 nucléotides. Par contre le fait de les avoir assemblées dans une vue à laquelle on applique une contrainte de taille comprise entre 8 et 20 nucléotides fait que les chaînes 2. et 3. ne sont pas des instances du modèle Logol, 2. étant trop courte et 3. trop longue.

b) Outil Logol

Logol est disponible sur la plateforme bioinformatique de genouest (http://www.genouest.org/spip.php?article757). Il est muni d'une interface graphique, LogolDesigner, qui permet de dessiner les modèles plutôt que d'écrire les règles de grammaire (figure 5 à 7). Une fois le modèle défini, il est soumis au module LogolAnalyser (http://webapps.genouest.org/org.irisa.genouest.logol.LogolAnalyser/) qui va chercher les instances du modèle dans la ou les séquences qui lui sont fournies en entrée. Ces séquences peuvent être des séquences personnelles ou des séquences de banques publiques, accessibles via l'interface. En résultat, l'analyse renvoie un fichier XML contenant les instances du modèle qui ont été identifiées (nous appellerons par la suite ces instances: *hit* ou *match*). Ce fichier XML est interprétable par l'application web pour la visualisation des résultats détaillés.

c) Environnement d'exécution Genouest

Il est également possible d'utiliser Logol en ligne de commande par l'intermédiaire du genocluster (genocluster2.irisa.fr), à condition d'avoir un compte sur celui-ci. C'est d'ailleurs la méthode qui a été utilisée le plus souvent lors de cette étude. Voici quelques informations techniques sur cet environnement.
La plate forme bioinformatique de genouest (http://www.genouest/org/) est une des plates-formes technologiques de Biogenouest. Elle est portée par l'INRIA et l'IRISA et est située sur le campus universitaire de Beaulieu à Rennes.
Elle met à disposition des utilisateurs disposant d'un compte un accès à de nombreuses ressources: puissance de calcul, espaces de stockage, banques de données et logiciels au service de la recherche en biologie. En mars 2012, l'architecture de la plate-forme est la suivante:

- 16 nœuds AMD Opteron bi-processor (Sun V20Z) 4Go de mémoire.
- 14 nœuds bi-processor quad-core (SGI), 64 Go RAM
- 7 nœuds bi-processor quad-core (SGI), 144 Go RAM
- Stockage des données avec un rack Panasas d'une capacité de 50 To.

La plate-forme fonctionne sous Linux. L'accès aux machines s'effectue en mode terminal (SSH) avec une connexion sur un frontal (genocluster2.irisa.fr) à partir duquel les exécutions de Logol (dans notre cas) sont lancées via la commande « qsub » qui se charge d'envoyer les travaux sur différents nœuds du cluster selon leur disponibilité. La gestion des tâches est effectuée grâce à l'ordonnanceur SGE (*Sun Grid Engine*).

## II.3. Analyse de l'existant sur les frameshift -1

De nombreux travaux ont été publiés sur les *frameshift* -1 avec pour objectif de les identifier[7,16,17,18,19,20]. La détection des évènements de *frameshift* -1, ainsi que la compréhension des mécanismes mis en jeu, reste cependant un sujet de recherche actif.

Afin de mettre en place notre propre méthodologie de détection des événements de *frameshift*s -1 avec l'élaboration d'un modèle Logol, nous avons exploré la littérature suivant trois volets : les méthodes ou approches d'identification de sites de *frameshift* -1 employées, les modèles établis ainsi que les données utilisées.

a) Modèles et compartiments qui composent le *frameshift* -1

Les approches employées par les différents auteurs pour la détection de *frameshift* -1 ont pour point commun d'utiliser un modèle de base, une sorte de patron, qui va décrire les différents compartiments composant le *frameshift* -1 et en borner les tailles (figure 4). Les auteurs s'accordent pour un certain consensus autour des compartiments de la *fenêtre glissante* et de l'*espaceur*. La fenêtre glissante est héptamérique, de la forme X XXY YYZ avec X représentant n'importe quelle base répétée trois fois, Y est une base faible, soit une adénine (A) soit un uracile (U) répétée trois fois également et enfin Z est une base différente de la guanine (G)[21,22,23,24] à l'exception de deux études qui tolèrent un G à cette position[16,19]. L'espaceur est, dans chacun des modèles, considéré comme un intervalle entre la fenêtre glissante et la structure initiatrice. Si quelques informations sur le contenu nucléotidique de l'espaceur sont mentionnées pour agir comme un modulateur d'efficacité du décalage de phase[25], les modèles rencontrés prennent uniquement en compte des bornes de taille sur ce compartiment (il est inférieur à 12 nucléotides dans tous les cas). Là encore des travaux montrent expérimentalement que la taille de l'espaceur module l'efficacité du décalage de phase avec une efficacité maximum pour un espaceur de taille 7-8 nucléotides[25].

En ce qui concerne la *structure initiatrice,* nous avons vu qu'elle prenait généralement la forme d'un pseudo-nœud ou d'une tige-boucle et plusieurs informations intéressantes ont pu être extraites à leur sujet. Tout d'abord la présence d'un pseudo-nœud est plus efficace pour le décalage de phase qu'une tige-boucle[4]. Ceci s'explique par le fait que les pseudo-nœuds forment une structure plus robuste (plus stable) qui est plus apte à jouer le rôle de blocage mécanique du ribosome, dans la mesure où elle oppose plus de résistance à l'avancée du ribosome lors de la traduction. En conséquence un *frameshift* -1 initié par une structure de type tige-boucle est plus rare et nécessite, par exemple, une fenêtre « particulièrement » glissante comme l'est par exemple l'heptamère « U UUU UUA »[23]. Pour le cas plus spécifique des pseudo-nœuds, initiateurs du décalage de phase en -1, on distingue les pseudo-nœuds de type H (composés de deux tiges) des autres pseudo-nœuds (avec 3 tiges par exemple). Au sein même des pseudo-nœuds de type H trois sous catégories peuvent-êtres définies[11] qui diffèrent principalement au niveau des tailles des deux tiges et des boucles.

La taille des tiges est un élément déterminant pour la fréquence à laquelle s'opère le décalage, au même titre que la composition des tiges[26]. Ces deux éléments sont en effet en relation directe avec la stabilité de la structure formée. Au sujet des boucles, L1' est parfois facultative tandis qu'une taille minimum est requise pour L1 et L2. Ceci explique les différentes nominations rencontrées selon les études pour les boucles, la première : L1, L1' (facultative) et L2 ou la seconde qui est celle présente sur la figure 4: L1, L2 et L3 (facultative). La dernière boucle est généralement la plus longue. Nous avons choisi de conserver la première nomination qui conserve la linéarité du pseudo-nœud (L1, L1',L2). Des études laissent penser que des interactions entre la troisième boucle (L2) et la première tige interviennent dans certains cas, ce qui tend à stabiliser un peu plus la structure formée[27].

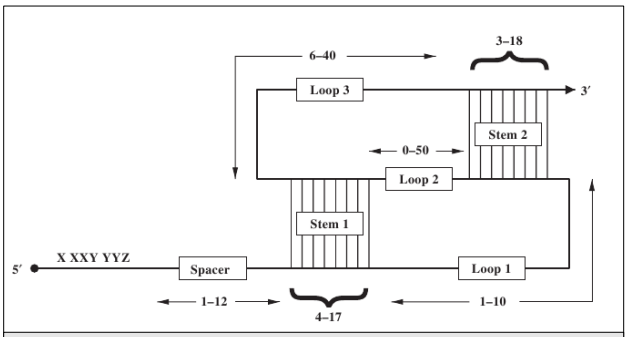



**Figure 4:** *Modèle de base - KnotInFrame Theis 2008*[16]

Tous ces éléments ne sont pas systématiquement pris en compte dans les modèles établis. De plus, aucun consensus précis n'est établi sur les paramètres qui définissent le pseudo-nœud (par exemple les tailles des tiges ou des boucles) et pour cause ces paramètres vont dépendre de l'approche employée, c'est-à-dire de la manière dont les auteurs tentent d'identifier les structures initiatrices des décalages de phase en -1. Les auteurs ciblent-ils un type précis de structure initiatrice? Avec quel degré de précision souhaitent-ils obtenir le détail des structures initiatrices de leurs candidats? Ces questions nous amènent à présenter les méthodes d'identification des sites de décalage de phase en -1.

b)Méthodes d'identification existantes

Les méthodes existantes sont généralement organisées en deux phases. Une première phase recherche des sites potentiels qui soient compatibles avec la première partie du modèle de base, i.e. correspondant à la fenêtre glissante. Une seconde phase recherche sur ces segments, en amont de la fenêtre glissante, des structures stimulatrices du décalage de cadre de lecture (i.e. généralement des pseudo-noeuds) compatibles avec le modèle défini au préalable. Cette phase fait appel à des calculs de stabilité se basant sur l'énergie libre de la conformation de la structure obtenue (de type MFE, *minimum free energy*). Des programmes spécialisés dans la détection des structures nouées -telles que les pseudo-nœuds - sont ainsi développés et utilisés lors de ces études (pknotRG-*fs*[16], FSFinder[19], pknot[28], pknotRG[29]).

Les sites de décalage de phase et le nombre de candidats étant pour chacun d'entre eux généralement très nombreux, une importante étape de filtrage est opérée se basant sur les calculs de stabilité effectués accompagnés parfois de calculs de vraisemblance statistique et de l'attribution de score pour les candidats. Ces scores permettent d'établir un classement des hits obtenus avec dans les rangs supérieurs les candidats les plus probables.

Sans détailler l'ensemble des études effectuées sur le sujet, nous nous attardons ici sur les travaux aboutis les plus récents, ceux de Theis *et al*[16], afin de pouvoir situer nos travaux dans la perspective des travaux existants. Theis *et al* ont développé en 2008 un outil complet destiné à la recherche de *frameshift* -1, appelé KnotInFrame. Pour constituer leur jeu de données de référence, ils ont extrait de la base Recode1 (base présentée dans la partie suivante) les données qui sont à la fois annotées pour un *frameshift* -1 et associées à un pseudo-nœud. Cela représente 28 identifiants. Leur modèle de base a été calibré sur ce jeu de référence de manière à ce que tous les identifiants soient conformes au modèle (figure 4). Le déroulement de l'analyse de nouvelles séquences pour y chercher de potentiels *frameshift* -1 se fait alors en quatre étapes :

1- une phase de recherche des fenêtres glissantes (avec élimination des hits ayant un stop en amont de la fenêtre glissante, ou étant trop près de la fin de la séquence - et qui ne laisserait pas de place pour un pseudo-noeud). Pour information, une séquence aléatoire contient environ neuf fenêtres glissantes par millier de nucléotides. L'étape 1 peut donc générer de nombreux résultats;

2- pour chaque fenêtre *f* trouvée, recherche de pseudo-nœuds putatifs en aval de *f*. La recherche est effectuée dans six séquences de plus en plus longues: *f*+26nt, *f*+46nt,...*f*+126nt, à la fois par un programme spécialisé pour les pseudo-nœuds spécialement développé pour cette l'étude (pknotRG-fs) et par un programme général de repliement (RNAfold[30]). PknotRG-fs cherche les meilleurs pseudo-nœuds (de moindre MFE) conformes au modèle de référence (celui de la figure 4). RNAfold quant-à lui recherche le meilleur MFE associé à cette séquence, avec des repliements qui ne sont pas forcément des pseudo-nœuds;

3- filtrage des nombreux repliements obtenus de façon à ne garder qu'un pseudo-nœud putatif par fenêtre glissante (suppression des pseudo-nœuds de trop forte énergie, puis ceux ayant une énergie trop éloignée du MFE optimal proposé par RNAfold, puis tri);

4- un calcul de score, dit de « dominance normalisée », est ensuite effectué pour juger l'ensemble des hits restant pour une même séquence. Au final, au maximum dix instances sont retenues par séquence d'entrée.

c) Bases de données sur les *frameshift* -1

Il s'agit ici de recenser les données utilisées par les différentes études, celles qui ont servi de référence pour la calibration des modèles ou pour les valider. Certaines bases de données spécialisées ont d'ailleurs été créées à l'issue de certaines études.

- La base de données Recode2[31] ou sa version antérieure Recode1 est largement utilisée. Cette base de données, comme son nom l'indique, recense les événements de recodage. Une partie des données de Recode2 concerne donc les événements de *frameshift* -1. Les identifiants de Recode2 rassemblent des informations très précises sur les séquences qui renferment un événement de recodage: le site où l'évènement se produit ainsi que le détail des structures initiatrices.

- La base de données FSDB[32] a été mise en place à l'issue des travaux de S.Moon[19]. et recense les événements de *frameshift* -1 et +1. Elle se veut complémentaire des bases existantes avec une interface complète en liaison avec leur outil de détection, FSFinder.

- La PseudoBase[33] est une base de données dédiée aux pseudo-nœuds. Certains identifiants sont associés à des événements de *frameshift* -1 mais ils représentent un nombre très limité. Certains identifiants sont intégrés dans les jeux de référence de certaines méthodes de recherche de *frameshift*-1 ou servent de données complémentaires aux données de Recode2 pour tester la validité des méthodes.

- PRFdb est une importante base de données qui recense des sites de *frameshift* -1 putatifs dans le génome de la levure (S. Cerevisieae). Elle fait suite aux travaux de J.L Jacobs[20] dédiés à cet organisme. Cette base rassemble de loin la plus grande quantité d'informations sur les *frameshift* -1 mais les résultats qui ont été insérés dans PRFdb se révèlent peu souvent adaptés à notre problématique puisque les structures initiatrices des décalages de cadre de lecture recensées proviennent d'un calcul d'énergie non spécialisé pour les structures nouées effectué avec RNAfold. Les événements et structures initiatrices recensées sont donc prédites et sont souvent de type tige-boucle. De plus la méthodologie adoptée pour constituer PRFdb est mise en question par une partie de la communauté[16].

## II.4. Élaboration des données de référence, modèles et méthodes de validation

a) Données

Nous avons choisi d'utiliser Recode2 pour établir un jeu de données de référence et ce pour plusieurs raisons. Les données y sont d'une part spécifiques aux événements de recodage et sont d'autre part bien détaillées. Chaque identifiant est annoté suivant l'origine du recensement de l'événement de recodage présent. C'est-à-dire soit l'événement est vérifié expérimentalement (identifiants « validated ») soit il est présent grâce à une méthode d'identification *in silico* sans

que l'événement ait jusqu'alors pu être vérifié (identifiants « reviewed »). L'événement est détaillé et dans le cas des *frameshift* -1 qui nous intéressent, on trouve par exemple la composition exacte (éventuellement putative) de la fenêtre glissante, de l'espaceur ainsi que le détail de la structure initiatrice. PRFdb rassemble quant-à-elle uniquement des événements putatifs. Leur validité n'est donc pas établie avec certitude, ce qui est un point négatif dans le cadre de l'élaboration d'un jeu de référence. La PseudoBase ne présente pour sa part pas assez de données pour pouvoir constituer un jeu de référence et FSDB n'est plus mise à jour depuis 2007.
On trouve dans la base Recode2 soixante-huit identifiants associés à un *frameshift* -1 et définis comme « validated » (pour lesquels il est donc avéré qu'un décalage de cadre de lecture en -1 intervient). Comme il a été mentionné précédemment, tous les *frameshift* -1 ne sont pas initiés par une structure de type pseudo-nœud. Dans le cas présent, 30 identifiants sur 68 sont associés à un pseudo-nœud. Nous avons collecté chacun des 30 identifiants sous la forme d'un fichier XML qui rassemble l'ensemble du détail de cet identifiant et nous avons pu en extraire des informations utiles à la calibration de notre modèle en complément de celles issues de la littérature. C'est ce jeu de données qui nous sert de référence.
Nous avons également collecté les identifiants associés à un *frameshift* -1 et à une structure initiatrice de type pseudo-nœud mais cette fois non validés, ce sont les identifiants putatifs (*reviewed)* qui représentent 26 identifiants supplémentaires. Ils ne sont pas directement intégrés au jeu de données de référence mais ils serviront pour tester les modèles Logol sur un jeu de données élargi.
**Notes**:
- 16 identifiants de notre jeu de données de référence sont partagés avec le jeu de données de référence établis par Theis *et al*[16] lors de l'élaboration de KnotInFrame. Ce nombre monte à 28 si l'on prend en compte les identifiants putatifs que nous avons extraits .
- On verra par la suite que six des trente identifiants du jeu de référence ne correspondent pas au modèle de base. Leurs anomalies (e.g. problème de format de la fenêtre glissante ou du pseudo-nœud) sont mises en lumière dans un deuxième temps, au moment de l'analyse (cf section III.3.a). Le « **noyau dur** » des séquences de référence est donc constitué de **24** séquences.

b) Élaboration d'un premier modèle
La trame du modèle est connue. Elle se compose de trois compartiments principaux: la fenêtre glissante, l'espaceur et la structure initiatrice. Notre modèle a été calibré à partir des informations de la littérature et de l'observation des séquences de référence les plus caractéristiques (les 24).

*1) Fenêtre glissante*

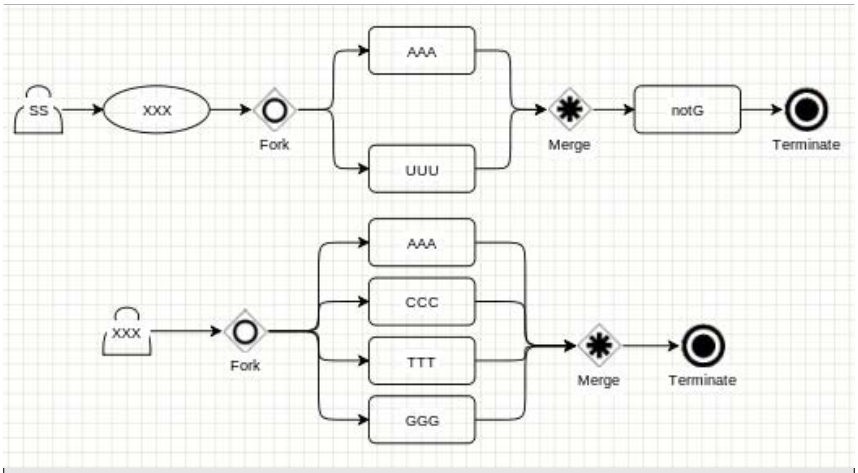



**Figure 5:** *Extrait du modèle frameshift Logol de base: fenêtre glissante (Effectué sous LogolDesigner)*

Conformément à la littérature, nous respectons le consensus établi en ce qui concerne la fenêtre glissante héptamérique de la forme X XXY YYZ. Avec X une répétition d'un nucléotide quelconque trois fois, Y une répétition de la base A ou U, et Z différent de G (cf figure 5).

*2) Espaceur*

C'est la zone intercalée entre la fenêtre glissante et la structure initiatrice du décalage du cadre de lecture. Sa limite de taille supérieure se situe, selon les modèles définis lors des différentes études, entre 8 et 12 nucléotides. Comme aucun identifiant du jeu des 24 données de référence n'a d'espaceur supérieur à 10 nucléotides, nous avons limité la taille de l'espaceur en Logol à 10 nucléotides. Aucune information n'est apportée sur le contenu de l'espaceur. Il se limite donc à un *gap* dans notre modèle; c'est à dire une variable dont la taille est inférieure ou égale à 10 nucléotides mais non nulle.

*3) Structure initiatrice*

Nous avons choisi de modéliser la structure initiatrice la plus typique, en l'occurrence le pseudo-nœud H-type.

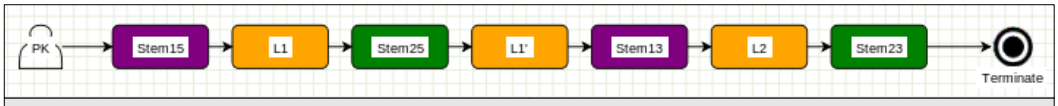



**Figure 6:** *Extrait du modèle frameshift Logol de base: pseudo-nœud H-Type (Effectué sous LogolDesigner)*

Voici comment a été modélisé le pseudo-nœud H-Type sous LogolDesigner (figure 6): la première tige est désignée par deux éléments Stem15 (la tige 1 aller) et Stem13 (la tige 1 retour), la deuxième tige par deux éléments Stem25 et Stem23, la première boucle est constituée de la succession des éléments L1, Stem25 et L1', enfin la deuxième boucle est désignée par L2.
Nous décrivons ici les détails de ce modèle. Le contenu de Stem15 est sauvegardé dans une variable. Pour Stem13, on impose comme contrainte de contenu la séquence de Stem15 précédemment sauvegardée à laquelle on applique le morphisme de complémentarité inverse (-wc), ainsi les éléments Stem15 et Stem13 constituent un palindrome biologique, et donc une tige. Il en va de même pour Stem25 et Stem23. La taille de la tige 1 est fixée entre 4 et 16 nucléotides et la taille de la tige 2 entre 3 et 8 nucléotides. On autorise initialement quatre erreurs (*mismatch*) pour la tige 1 et deux erreurs pour la tige 2. Les boucles L1, L1' et L2 mesure respectivement [1:5], [0:4] et [4:40].

*4) Contraintes de trame*

D'un point de vue biologique, il est nécessaire que les deux protéines productibles via le décalage de phase soient valides, notamment qu'elles commencent par un codon start et se terminent chacune par un codon stop.
Plus précisément les contraintes de trame sont les suivantes. La séquence analysée doit débuter par un codon start 'AUG'. Ensuite, il ne doit pas y avoir de codon stop intercalé en phase 0 entre le start et la fenêtre glissante, car ce ne serait dès lors plus un cadre ouvert de lecture (« Open Reading Frame »). De plus, la fenêtre glissante doit être calée en phase -1. Enfin, les deux phases de la séquence doivent chacune s'achever par un codon stop 'UAA,'UAG' ou 'UGA'. Il doit donc y avoir un stop en aval de la fenêtre glissante, à la fois en phase 0 et en phase -1.
Le modèle Logol correspondant à ces contraintes est présenté en figure 7. Le modèle model, dont le détail n'est pas présenté sur la figure 7, recherche un start suivi d'une fenêtre glissante en phase -1 puis d'un pseudo-nœud. Le modèle ORF recherche un start, suivi d'un certain nombre de

codons non-stop (recherche réalisée par codonRepeat) puis d'un stop, ainsi situé par construction en phase 0. Le modèle ORFMINUS quant à lui recherche un start, suivi de deux nucléotides (représentés par shift), d'un certain nombre de codons non-stop, puis d'un stop, ainsi situé, du fait du shift, en phase -1.
Les éléments model, ORF et ORFMINUS sont mis en séquence dans la règle principale rule. En Logol, cela signifie qu'ils expriment des **modèles alternatifs**, c'est-à-dire qu'ils amènent à faire trois analyses complémentaires de la séquence. Ainsi la séquence devra correspondre à la fois au prototype décrit par model (i.e. contenir un start puis les compartiments d'un *frameshift*-1), à celui décrit par ORF (contenir un start et un stop en phase 0) et à celui décrit par ORFMINUS (contenir un start et un stop en phase -1). Pour que ces trois vérifications aient un sens, elles doivent être calées sur le même codon start. Ce calage est réalisé par un passage de paramètre (la position du start) entre les trois modèles alternatifs.
Techniquement, le modèle codonRepeat utilise des éléments assez élaborés du langage Logol. Il est réalisé au moyen de l'opérateur de répétition « repeat » de Logol, à qui il est demandé ici de rechercher un certain nombre d'instances contiguës du modèle notstop, où notstop (cf figure 7.b) est défini en Logol par une « vue » décrivant un segment de trois caractères (par une contrainte de taille) qui ne soit pas un stop (par une contrainte de « contenu négatif »).

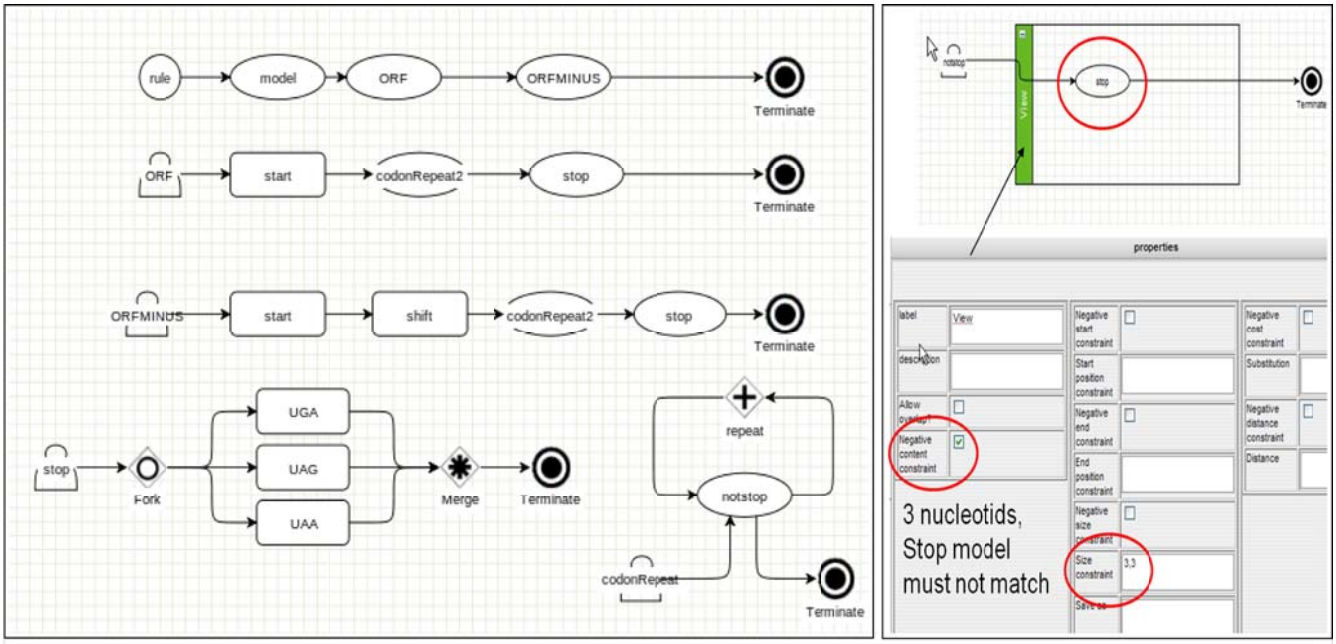



**Figure 7 :** *Extrait du premier modèle Logol de frameshift -1 : vérification de trames (Effectué sous LogolDesigner)*

**7. b :** *zoom sur le modèle* notstop

Ceci représente les grandes lignes du premier modèle de *frameshift* -1 réalisé en Logol. En pratique, ce modèle s'est révélé inefficace, et des raffinements successifs ont été élaborés au cours de cette étude afin de le rendre notamment moins consommateur en mémoire et plus spécifique.
Un des soucis concernant la consommation mémoire provient de l'utilisation du codonRepeat. En effet, avec le modèle de la figure 7, le fichier de résultats mémorise tous les codons rencontrés dans le codonRepeat et il en résulte des fichiers beaucoup trop volumineux. En l'état actuel de Logol, cette modélisation fine n'est donc utilisable que sur de petites séquences, et pour limiter la taille des fichiers résultats dans les traitements grande échelle, la vérification de calage de phase a ici finalement été réalisée par post-traitement.

Avant de décrire les autres étapes d'évolution du modèle, nous présentons notre méthode de validation.

## II.5. Méthode de validation des modèles

Au cours de cette étude, le modèle de base a été successivement modifié dans l'optique de remplir au mieux l'objectif suivant : détecter un maximum d'événements de *frameshift* -1 présents dans nos séquences de référence, tout en générant le moins possible de résultats parasites (i.e. meilleur ratio sensibilité/sélectivité).

a) Phase d'analyse

L'analyse, avec le modèle Logol à tester, est lancée en premier lieu sur les séquences du jeu de données de référence, puis elle est lancée dans un second temps sur un jeu de séquences aléatoires. Le jeu de données aléatoire est constitué de la façon suivante : chaque identifiant du jeu de donnée de référence est aléatoirement mélangé 100 fois par le logiciel Shuffleseq (http://emboss.bioinformatics.nl/cgi-bin/emboss/shuffleseq), ce qui a pour intérêt de conserver la longueur des séquences et le pourcentage nucléotidique pour chaque set constitué. Ces deux paramètres ont en effet un impact sur la nature et le nombre de hits obtenus. On ne peut donc se contenter de séquences avec une longueur et un pourcentage en GC constant. Nous disposons de 30 identifiants de référence, ce qui donne 30 sets de 100 séquences aléatoires.

Pour évaluer la qualité des résultats obtenus, plusieurs éléments sont pris en compte : le nombre de sites de *frameshift* -1 identifiés par le modèle, c'est-à-dire le nombre de fenêtres glissantes distinctes (en terme de position dans la séquence) présentes parmi les matchs; le nombre de matchs est également une source d'information car il peut y avoir plusieurs hits détectés pour un même site (plusieurs pseudo-nœuds); la pertinence des matchs est évaluée pour sa part suivant un calcul de score qui prend en compte la composition et la longueur des tiges. Ces calculs et informations diverses sont obtenus dans une phase de post-traitement.

b)Post-traitement

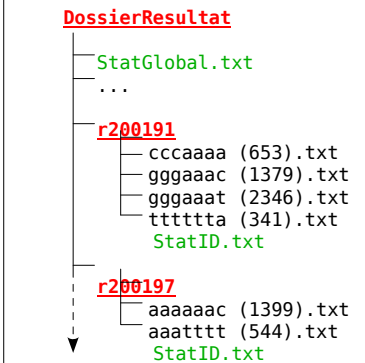
Il se compose de deux étapes, une étape d'exploration des fichiers résultats au format XML issus de l'analyse Logol et une étape de calcul statistique (quantitative) et de classement des hits obtenus. C'est lors de ce post-traitement que les hits sont répartis dans des groupes selon leur position dans la séquence. Ces groupes sont appelés « clusters ». Un cluster rassemble donc tous les hits dont la fenêtre glissante se situe à la même position dans la séquence d'entrée. Deux informations sont également ajoutées pour chaque hit, la notation du pseudo-nœud au format *Dot-bracket* (figure 8.2) ainsi qu'un score permettant d'établir un classement entre les hits. Le score est calculé à partir d'une matrice de calcul de score issue des travaux de Forest[34]; le score se base sur le nombre et la nature des liaisons présentes dans les deux tiges formant le pseudo-nœud. La formule de calcul est la suivante : {GC = +3, AU= +2, GU= +1, *mismatch* = -2}. On notera que l'appariement GU (*wobble*) possède un score (+1) beaucoup moins pénalisant qu'un *mismatch* (-2).

Trois scripts ont été développés pour le post-traitement. Ces scripts sont écrits dans le langage Python. Le premier est un script de *parsing* des fichiers XML résultats de l'analyse Logol. Il utilise la librairie libxml2 qui tient compte de la syntaxe XPath ce qui facilite l'extraction des éléments souhaités. Une première information est ajoutée pour chaque hit: il s'agit de la visualisation du pseudo-nœud au format dot-bracket (figure 8).

Le second script organise lors de son exécution les hits dans une arborescence de dossiers et fichiers selon la séquence analysée et les clusters identifiés (figure 8). Il attribue les scores aux différents hits et renvoie les premiers fichiers synthétiques de résultats statistiques. A l'origine, ce script avait également pour objectif, avec l'utilisation des premiers modèles Logol, d'agir en tant que filtre éliminant *a posteriori* les matchs dont les pseudo-nœuds n'étaient pas réalistes (du fait

principalement d'une trop faible proportion d'appariement GC dans les tiges). Ce type de postfiltre a été supprimé par la suite car ces contraintes de composition des pseudo-nœuds ont finalement pu être directement implantées dans le modèle Logol (cf section III.1 et figure 10).

**1**




**2**

```
match65
ID: r200191
Slippery Side (position fenêtre glissante): GGGAAAC (1379)
Spacer: GGAGUG
Stem15: CGCGG
Loop 1: CA
Stem25: CCGU
Loop 1': NULL
Stem13: CCGCG (0)
Loop 2: GAACAA
Stem23: ACGG (0)

Pseudonoeud: CGCGGCACCGUCCGCGGAACAAACGG
             ((((((..[[[[))))))......]]]]

Score: 34
```

**Figure 8 - 1:** *Arborescence créée par le second script du post-traitement* – **2**: *Détail d'un match*

L'arborescence créée (figure 8.1) rassemble un dossier pour chaque identifiant analysé, des premières statistiques quantitatives sont calculées au niveau de chaque identifiant (StatID.txt ) ainsi qu'un fichier de statistiques plus globales sur l'analyse complète effectuée (StatGlobal.txt). Les matchs sont rassemblés par cluster en fonction de leur position dans la séquence, c'est-à-dire la position de leur fenêtre glissante. Le score est calculé à l'issue du second script tandis que la visualisation du pseudo-nœud au format Dot-Bracket (mise en évidence par le rectangle) est ajoutée par le premier script (figure 8.2) .
Le dernier script complète les statistiques en vérifiant la validité des clusters (i.e. vérification des contraintes de trames : contrôle de la position des codons stop en phase 0 et en phase -1) et en réalisant le classement final des hits selon leur score en tenant compte de la validité des différents sites identifiés.

| *Identifiant* | Nombre de hit total: 16 | Nombre cluster(s): 5<br>**Cluster1 : tttaaac (13376) 5 (31.25%)**<br>Cluster2 : aaatttc (20511) 3 (18.75%)<br>Cluster3 : gggaaat (28715) 1 (6.25%)<br>Cluster4 : aaatttt (122) 4 (25.0%)<br>Cluster5 : tttaaat (22984) 3 (18.75%) |
|---|---|---|

**Figure 9:** *Extrait d'un fichier synthétique crée à l'issue de l'exécution du dernier script du post-traitement (validation des clusters)*

Plusieurs informations sont rassemblées dans ce fichier. Pour chaque identifiant soumis à l'analyse Logol on recense: le nombre de hits totaux trouvés, le nombre de clusters présents avec comme précision la composition de la fenêtre glissante, sa position dans la séquence, le nombre de hits qu'il contient ainsi que le pourcentage que cela représente sur le total des hits pour cet identifiant.
Les clusters verts sont, pour un identifiant donné, les hits que le modèle a trouvé et qui ont été déclarés valides lors de la validation du cluster par le dernier script; ce sont les positifs.
Les clusters rouges sont les sites que le dernier script a déclaré invalides et qui ne peuvent donc être des sites de *frameshift* -1. Leur présence est dûe au fait que cette étape de validation est

intégrée au post-traitement. Lorsque les résultats sont issus d'une analyse Logol sur le jeu de données de référence, un cluster peut être identifié en gras. Il désigne alors l'emplacement du *frameshift* -1 indiqué par Recode2 pour cet identifiant. C'est donc le lieu où s'opère officiellement le décalage de phase.
Pour chaque nouveau modèle établi, nous estimons ainsi sa pertinence en comparant le nombre d'événements *frameshift* correctement prédits (les vrais positifs) et le nombre d'événements parasites. Le classement des hits permet de pousser l'analyse plus en détail en vérifiant à quelle position se trouve le hit exact (celui fournit par Recode2) dans le classement lorsqu'il a été retrouvé lors de l'analyse Logol. A défaut, on vérifie manuellement quel est le hit qui s'en approche le plus et son classement correspondant. Il arrive parfois qu'aucun hit ne soit retrouvé pour le cluster de Recode2, ce qui amène à en étudier la raison. Généralement l'analyse renvoie des hits très proches de celui que fournit Recode2 ce qui permet malgré tout d'en tirer des informations.

# III. Résultats

La modification successive du modèle de base de *frameshift* -1, afin de le rendre le plus efficace possible suivant les objectifs de sensibilité/spécificité habituels, a permis d'aboutir à un modèle final que nous décrivons ici accompagné de ses performances. La réalisation de cette tâche nous a amené à nous interroger sur la façon de traduire en Logol la notion de stabilité des structures, notion qui est habituellement mesurée par un calcul d'énergie libre (les structures présentant l'énergie libre minimum - le plus petit « MFE » - étant les plus stables) et que nous avons tenté d'approcher en terme de composition de séquences. On a également étudié le problème général de la pertinence de la modélisation des pseudo-nœuds en Logol, en comparant nos résultats avec ceux issus de logiciels de prédiction de pseudo-nœuds basés sur un calcul de MFE.
Les ajustements envisagés sur le modèle de *frameshift* -1 en Logol nous ont amené d'une part à utiliser des fonctionnalités de Logol qui n'avaient parfois pas encore été explorées ou véritablement testées, telles que l'autorisation des *mismatchs* en pourcentage ou bien l'application de contraintes de pourcentage nucléotidique. D'autre part cela nous a permis de proposer de nouvelles fonctionnalités pour Logol, ce qui était l'une des visées de cette étude. L'ensemble de ces résultats est présenté dans cette section.

## III.1. Modèle final de détection des événements de *frameshift* -1

Le modèle de base n'étant pas très performant car peu sélectif a donc dû été modifié. Si la modélisation de la fenêtre glissante et de l'espaceur ne présentait pas de réelle marge de manœuvre, il restait la possibilité d'affiner la modélisation du pseudo-nœud. Une plongée dans la littérature a donc été entreprise pour en extraire des caractéristiques fines concernant les pseudo-nœuds. Notre jeu de données de référence a été utilisé en complément, pour confirmer les informations bibliographique et pour aider à calibrer les modèles.
La modélisation finale du pseudo-nœud est présentée en figure 10. De nombreuses modifications ont été effectuées depuis le premier modèle, essentiellement sur les deux tiges: la notion de *mismatch* est exprimée plus fidèlement, la description des branches des tiges se fait désormais en trois variables et une contrainte de contenu à été ajoutée. Nous allons préciser comment ces paramètres ont été définis.

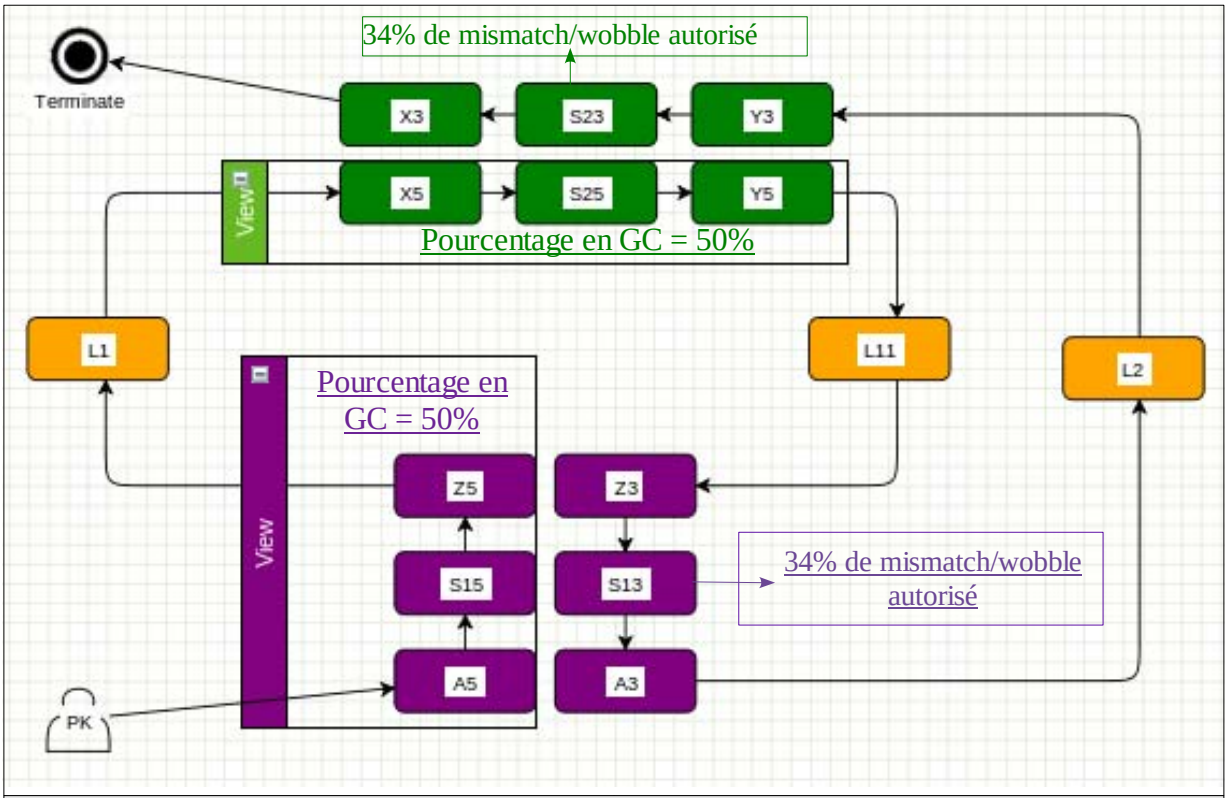



**Figure10:** *Extrait du modèle frameshift Logol finale: pseudo-nœud H-Type (Effectué sous LogolDesigner)*

a) Autorisation des *mismatchs* et prise en compte des liaisons *wobble*

Avec le modèle de base, il est par exemple possible de trouver des hits avec une tige 1 de quatre nucléotides dont deux *mismatchs* accompagnés d'une tige 2 composée de trois nucléotides dont un *mismatch*. Ce match n'est pas pertinent biologiquement, tout comme un hit qui présente un *mismatch* à une des extrémité des tiges du pseudo-nœud. Ces matchs étaient des instances du premier modèle du fait de l'autorisation de *mismatchs* en valeur entière sur l'ensemble de la tige. Une première étape est donc de « fixer » les extrémités des deux tiges afin d'interdire les *mismatchs* à ces emplacements de la tige. Il suffit pour cela de fragmenter chaque branche des tiges en 3 variables; le nucléotide situé à l'extrémité coté 5', le corps de la branche et le nucléotide situé à l'extrémité coté 3' de la tige.

Un autre point s'est révélé d'une grande importance lors de l'étude bibliographique et a été confirmé par le détail du jeu de données de référence. Il s'agit de la nature des appariements présents. Le morphisme de complémentarité inverse de Logol (-wc) n'autorise que les liaisons standards de l'ARN (les liaisons *Watson-Crick*: G-C et A-U). Il n'autorise pas les liaisons *Wobble* (appariement G-U). Nous avons décidé dans un premier temps de les prendre en considération dans le compteur de *mismatchs* autorisés sur le corps de la tige (la tige privée de ses extrémités). Ainsi seront comptabilisés comme *mismatch* les réels *mismatchs* mais aussi les appariements *wobble* (NB: la possibilité d'utiliser un morphisme spécifique, *wcw* pour « *Watson-Crick-Wobble* », a été explorée postérieurement à cette étude, cf l'annexe 2). Le taux de *mismatchs* autorisés est donc assez élevé. La valeur en pourcentage déterminée à l'aide du jeu de données de référence est de 34% sur l'intérieur des tiges ce qui correspond à un *mismatch*/*wobble* pour une tige de cinq nucléotides, deux *mismatchs*/*wobble* pour une tige de 8 nucléotides, trois pour 11, quatre pour 14 etc. L'instauration de ces deux contraintes (*mismatch* en pourcentage et extrémités fixées) participe de manière drastique à la réduction du nombre de sites candidats au *frameshift* -1

obtenus lors de l'analyse Logol sur le jeu de référence et aussi par conséquent à la réduction du nombre de hits en éliminant les hits qui possédaient trop de *mismatchs* dans leurs tiges.
Voici comment on procède en Logol: les branches des tiges sont désormais fragmentées en trois parties, et les *mismatch*/*wobble* sont uniquement autorisés sur l'intérieur des tiges (S13 et S23) afin d'exclure les *mismatchs* sur l'extrémité des tiges.

b) Traduction en Logol de la notion de stabilité

En suivant les informations recueillies lors de l'étude bibliographique, il s'avère que les tiges doivent nécessairement avoir une certaine stabilité pour que la structure formée soit suffisamment stable pour pourvoir initier le décalage. Plus généralement, il faut déjà pour commencer qu'elles puissent effectivement être une conformation plausible de la portion d'ARN en question. Nous avons donc entrepris de traduire cette notion de stabilité en terme de composition nucléotidique des tiges et notamment en terme de pourcentage en GC, facilement exprimable en Logol. Nous avons complété ces informations par une série de tests effectués avec les outils RNAfold et RNAeval en faisant varier la quantité des appariements présents dans une structure de type tige-boucle de taille constante. Nous avons ainsi pu vérifier que la nature et l'agencement des différentes liaisons au sein de la tige est en relation directe avec la valeur d'énergie libre attribuée à cette structure.
Il apparaît d'une part que l'énergie est d'autant plus faible (et donc que la structure est d'autant plus stable) que le nombre de liaisons G-C dans la tige est élevé. D'autre part un positionnement de liaisons G-C à la base de la tige améliore la stabilité (vis-à-vis de leur positionnement « coté boucle »). Ceci nous a également permis de confirmer l'importance de la prise en compte des appariements *wobble*. Toutefois ces liaisons apparaissent relativement moins stables que les liaisons A-U et nettement moins stables que les liaisons G-C (ce qui va dans le sens de la matrice de calcul du score utilisée). Ces tests sont à prendre avec précaution puisqu'ils sont effectués sur des tiges-boucles qui à la différence des pseudo-nœuds ne sont pas des structures nouées.
En détaillant les pourcentage en G-C des tiges S1 et S2 des identifiants de Recode2 de notre jeu de données de référence, on se rend compte qu'ils sont très souvent supérieur à 80% (100% chez de très nombreux identifiants). Nous avons déterminé qu'un pourcentage en G-C minimum de 50% sur chacune des tiges permet à la fois d'être en accord avec une grande majorité des identifiants du jeu de données de référence tout en allant dans le sens des informations de la littérature trouvées sur la composition des tiges des pseudo-nœuds et dans le sens des tests effectués sur les structures non nouées.
En Logol, cela s'effectue avec la création de deux vues qui rassemblent les trois variables qui composent la branche coté 3' des tiges 1 et 2. On applique une contrainte de pourcentage en G-C minimum de 50% sur cet ensemble mais les contraintes individuelles des éléments de la tige sont conservées (cf. le modèle complet en annexe 1).

## III.2. Résultats du premier modèle

En ce qui concerne l'analyse Logol lancée avec le premier modèle sur les 30 identifiants du jeu de données de référence on recense: 359 clusters dont 207 validés à l'issue du post-traitement. Le site de *frameshift* officiel (donné par Recode2) est retrouvé pour 28 identifiants et il est déclaré valide pour 26 d'entre eux. Nous ne disposons pas du nombre total de hits puisque nous avons limité l'analyse à 20 000 matchs par identifiant et que ce chiffre à été atteint pour bon nombre d'entre eux (le nombre de matchs est supérieur à 250 000 hits sur l'ensemble du jeu de référence). Bien sûr, parmi ces hits on retrouve des matchs intéressants qui correspondent à ceux fournis par Recode2 mais ils se retrouvent noyés dans une importante quantité de hits parasites. C'est pour cette raison que rapidement nous avons concentré nos efforts sur le perfectionnement de la modélisation pour améliorer ce ratio. Voici à présent les résultats plus détaillés pour la dernière version du modèle que nous avons élaboré à la fin de cette étude.

## III.3. Résultats du modèle final

a) Analyse du jeu de données de référence

L'analyse Logol effectuée avec le modèle final recense pour le jeu de données de référence 122 clusters dont 83 sont validés à l'issue du post-traitement pour un total de 2534 hits (les hits des clusters non valides sont compris dans ce chiffre). La réduction du nombre de clusters et surtout du nombre de hits est importante.

| Identifiant | cluster(s) valide(s) | Présence du Hit Recode | Rang cluster | Rang global |
|---|---|---|---|---|
| r200015 | 2 | Oui | 1 | 1 |
| r200090 | 1 | Oui | >10 | >10 |
| r200163 | 10 | Non (wobble) | 1 | 1 |
| r200179 | 2 | Oui | 1 | 1 |
| r200191 | 3 | Oui | 1 | 4 |
| r200197 | 2 | Non (wobble) | 1 | >10 |
| r200199 | 4 | Oui | 1 | 1 |
| r200208 | 4 | Non (wobble) | 1 | 1 |
| r200209 | 3 | Non (wobble) | 1 | 1 |
| r200210 | 5 | Non (wobble) | 1 | 1 |
| r200211 | 1 | Non (wobble) | 1 | 1 |
| r200216 | 1 | Oui | 7 | 7 |
| r200217 | 1 | Oui | 1 | 1 |
| r200221 | 4 | Oui | 1 | 1 |
| r200518 | 3 | Oui | 1 | 1 |
| r200525 | 1 | Non (wobble) | 1 | 1 |
| *r200528* | 2* | Non (wobble) | 3 | >10 |
| r200533 | 2 | Oui | 2 | 3 |
| r200550 | 2 | Oui | 1 | 1 |
| r200552 | 3 | Oui | 1 | 1 |
| r200553 | 3 | Oui | 1 | 4 |
| r200558 | 3 | Non (wobble) | 2 | 4 |
| r200562 | 8 | Non (wobble) | 1 | 1 |
| r200564 | 3 | Oui | 1 | 1 |
| *r201052* | 1* | Non (wobble) | 1 | 1 |
| r201056 | 4 | Oui | 2 | 2 |
| *r200200* | 2 | x | x | x |
| *r200203* | 3 | x | x | x |
| *r200219* | 3 | x | x | x |
| *r200517* | 2° | x | x | x |

**Tableau 1:** *Présentation des résultats de l'analyse Logol du modèle final effectué sur le jeu de données de référence.*

Ce tableau présente les résultats de l'analyse Logol lancée sur les 30 identifiants du jeu de données de référence.

La seconde colonne renseigne le nombre de cluster(s) déclaré(s) valide(s) à l'issue du post-traitement. Pour six d'entre eux il est unique et correspond au site de décalage défini par Recode2. 70% des identifiants ont trois sites valides ou moins.

La troisième colonne précise si l'analyse Logol a permis de retrouver le hit exact de Recode2. La mention « Oui » est indiquée si chaque élément d'un hit issu de l'analyse Logol correspond parfaitement au détail du *frameshift* -1 indiqué par Recode2.

Dans les autres cas, l'analyse a trouvé des hits sur ce site (à une exception près détaillée par la suite) mais aucun d'entre eux ne correspond exactement au site défini sous Recode2.

La troisième colonne indique le nombre d'instances du modèle trouvées lors de l'analyse Logol pour le site de décalage de Recode2.

La colonne « rang cluster» mentionne le classement obtenu par les hits issus de l'analyse Logol (dans le cluster de Recode2) qui correspondent au hit « officiel » de Recode2. Si le hit exact n'est

pas retrouvé on indique le rang du hit le plus proche de celui fourni par Recode2.
La colonne « rang total » indique le classement obtenu par le hit exact ou approché parmi l'ensemble des hits valides trouvés pour cet identifiant. Cela permet de prendre en compte le fait qu'il y ait parfois plusieurs sites valides identifiés par l'analyse Logol.

Six identifiants parmi les trente sont non conformes à notre modèle de *frameshift*-1. Le détail en est donné ici.
Pour les identifiants r200528 et r201052, le cluster du site de décalage du cadre de lecture défini par Recode 2 sera déclaré invalide(*) et il s'avère en effet qu'un codon stop est positionné en amont de la fenêtre glissante (après le codon start) pour l'un tandis qu'il n'y pas de stop positionné en phase 0 pour le second, ce qui est contraire à ce que l'on a défini à ce sujet. Cependant, la vérification de trame n'étant actuellement pas intégrée au modèle Logol (pour des raisons expliquées dans le dernier paragraphe de la section II.4) mais deléguée au troisième script du post-traitement (cf II.5.b), l'anomalie n'est pas révélée ici.
Par ailleurs, quatre identifiants entrent directement en conflit avec le modèle Logol et l'analyse ne renvoie pas de résultats probants à leur sujet.
- L'identifiant r200517 (°) possède une fenêtre glissante qui n'est pas conforme au modèle car elle ne respecte pas le consensus X XXY YY Z. L'analyse ne renvoie donc aucun hit pour ce site.
- L'identifiant r200219 ne possède pas visiblement pas de pseudo-nœud de type H. On retrouve de trop nombreux *mismatchs* dans les tiges ainsi que des boucles qui dépassent largement les tailles que nous avons fixées. De plus l'espaceur est vide.
-L'identifiant r200200 est un nouveau cas particulier qui présente des incompatibilités avec le modèle: le pseudo-nœud n'est pas de type H et l'espaceur se compose de 22 nucléotides (le modèle en autorise au maximum 10).
-L'identifiant r200203 est aussi un cas particulier puisque le pseudo-nœud est composé de trois tiges avec une boucle L2 qui dépasse 100 nucléotide. Il n'est pour l'instant pas envisageable de pouvoir le détecter actuellement.
Il ne s'agit pas ici d'occulter ces résultats mais les instances du modèle trouvées lors de l'analyse sont trop vraiment trop éloignées pour décréter que l'on détecte cet événement de *frameshift*.
Ces quatre derniers identifiants ne sont pas pris en compte dans le détail des résultats qui suivent.
Nous retrouvons 15 hits exacts sur 26 qui sont classés pour 11 d'entre eux au rang 1 (pour le cluster « officiel »). Plus globalement ces hits exacts sont classés dans le top 4 à treize reprises que l'on prennent ou non en compte les autres clusters valides de l'identifiant. Ceci permet de dire que notre approche est relativement discriminante puisqu'à de rare exceptions près le rang du hit exact baisse lorsque l'on prend en compte l'ensemble des clusters valides.
La raison pour laquelle le hit exact n'est pas retrouvé (hormis les quatre cas particuliers) c'est qu'une liaison *wobble* (G-U) est positionnée à l'extrémité d'une des tiges du pseudo-nœuds sous Recode2. C'est ce qui explique également que le hit le plus proche est malgré tout positionné au rang 1 puisque la différence entre le hit de Recode2 et ce dernier réside très souvent dans une taille d'une tige inférieur d'un nucléotide (du fait précisément de cet appariement *wobble* en extrémité). Les hits proches obtiennent également un bon classement puisqu'ils sont classés au rang 1 pour le cluster officiel à neuf reprises (sur 11 cas recensés) et au rang 2 et 3 pour les deux identifiants restants. Lorsque l'on prend en compte l'ensemble des clusters valides de l'identifiant, neuf sont situés dans le top 4 ce qui porte à 22/26 prédictions dans le top 4 pour le jeu de référence, un hit se classe septième et les trois autres sont relégués au delà de la dixième place.

b) Analyse du jeu de données aléatoires

En suivant la méthode de validation que l'on a définie, le modèle est ensuite testé sur un jeu de séquences aléatoires. Rappel: chaque identifiant est aléatoirement mélangé 100 fois. On dispose

donc de trente sets de cent séquences aléatoires.
En résultat, 1124 séquences aléatoires sur 3000 possèdent au moins un cluster « valide » (c-à-d ici : syntaxiquement recevable, d'après notre modèle) soit (37%) . Voici le détail des résultats pour l'ensemble des sets (tableau 2):

| 0 – 10 | 10 – 20 | 20 - 30 | 30 - 40 | 40 - 50 | 50 - 60 | 60 - 70 | 70 - 80 | 80 - 90 | 90 - 100 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 12 | 1 | 2 | 1 | 1 | 3 | 4 | 0 |
| 3.33% | 20.00% | 60.00% | 63.00% | 70.00% | 73.33% | 76.67% | 86.67% | 100.00% | 100.00% |

**Tableau 2:** *Détail du nombre de séquences aléatoire présentant au moins un cluster valide par set*

La première ligne est un intervalle: 0 -10 signifie « entre 0 et 10 séquences sur le set présentent un site valide au moins».
La seconde ligne indique comment les trente sets sont répartis dans les différentes catégories (suivant le nombre de séquences pour lesquelles des sites plausibles ont été identifiées). Ainsi par exemple, pour douze des sets, vingt à trente séquences parmi les cent ont un site détecté. Ce qu'on peut lire également : pour ces sets, au moins soixante-dix séquences sur les cent n'ont aucun match.
La dernière ligne cumule en pourcentage la proportion des sets pour les différents intervalles.
Ainsi par exemple, vingt-et-un sets (soit 70%) ont moins de 50% de leurs séquences qui contiennent un cluster valide.

Rappelons que chacune des vingt-six séquences du jeu de données de référence possédaient au moins un cluster valide. Un facteur important s'exprime ici, il s'agit de la taille des séquences des identifiants du jeu de référence qui varie selon les identifiants d'un facteur de 1 à 12. Plus la séquence analysée est longue et plus la probabilité de trouver une instance du modèle est forte. Malgré tout on observe une nette différence entre les résultats sur le jeu de données de référence et ceux obtenus sur les séquences aléatoires.
Un autre élément intéressant est le nombre de hits par cluster valide (i.e pour une même fenêtre glissante, le nombre de pseudo-nœuds trouvés). Ces informations sont visibles dans le tableau 4.

| | Nombre de hits du cluster de Recode2 | Moyenne du nombre de hits par cluster valide | Moyenne nombre de hits par cluster valide pour séquence aléatoire |
|---|---|---|---|
| r200208 | 245 | 70 | 4 |
| r200221 | 144 | 38 | 4 |
| r200015 | 119 | 63 | 3 |
| r200090 | 114 | 114 | 5 |
| r200518 | 110 | 37 | 8 |
| r200525 | 105 | 105 | 7 |
| r200211 | 103 | 103 | 6 |
| r200564 | 101 | 36 | 5 |
| r201056 | 92 | 26 | 7 |
| r200210 | 68 | 15 | 5 |
| r200179 | 64 | 36 | 8 |
| r200209 | 40 | 14 | 6 |
| r200191 | 35 | 24 | 15 |
| r200553 | 35 | 24 | 7 |
| r200163 | 32 | 7 | 5 |
| r200562 | 32 | 11 | 4 |
| r200528 | 28 | **89** | 5 |
| r201052 | 26 | 1 | 10 |
| r200558 | 18 | 7 | 9 |
| r200216 | 17 | 17 | 7 |
| r200197 | 13 | **16** | 4 |
| r200199 | 7 | 6 | 8 |
| r200552 | 6 | 5 | 7 |
| r200217 | 4 | 4 | 7 |
| r200533 | 4 | 3 | 5 |
| r200550 | 4 | **5** | 5 |
| Moyenne | 60,23 | 33,69 | 6,38 |

**Tableau 3:** *nb de hits trouvés dans le jeu de référence c1)pour le cluster « officiel » c2)tous clusters confondus (moyenne) c3) nb de hits trouvés dans le set aléatoire tous clusters confondus (moyenne)*

La première colonne rassemble le nombre d'instances du modèle trouvées lors de l'analyse Logol pour chaque identifiant dans le cluster officiel de Recode2, c'est-à-dire au site précis où l'évènement de *frameshift* -1 est défini pour cet identifiant dans la base de données. Ce chiffre baisse de manière générale (exception faite de trois contre exemples) sur la seconde colonne lorsque l'on prend en compte les autres sites valides recensés pour l'identifiant en question mais non répertoriés comme étant un site de décalage de cadre de lecture en -1 (i.e. hits localisés sur une autre fenêtre glissante). La troisième colonne est une moyenne du nombre de hits par cluster valide mesurée sur le set de séquences aléatoires correspondant à chaque identifiant. Ce chiffre est nettement inférieur dans la majorité des cas (hormis pour les cinq identifiants du jeu de données officiel qui présentent le moins de hits).
A noter qu'il n'est pas forcément mauvais de retrouver peu de hits pour un site donné. Parfois le modèle ne renvoie que peu d'instances pour un identifiant mais retrouve tout de même le hit exact, ce qui est une bonne chose. Néanmoins ces chiffres semblent être une indication sur la potentielle présence d'un événement de *frameshift* -1 dans les séquences en l'état actuel du modèle. Une grande quantité de hits viserait plutôt à confirmer cette tendance.

c) Analyse d'un jeu de données élargi
Une troisième analyse a été lancée sur le jeu de données élargi aux identifiants non validés (identifiants « reviewed ») de Recode2, ce qui représente vingt-six identifiants supplémentaires. Sept d'entre eux présentent des variations dans les trois premiers nucléotides de la fenêtre glissante qui entrent en conflit avec le modèle. Un autre identifiant ne possède pas de pseudo-nœud de type H. Sur les dix-huit restants, les résultats confirment les précédents puisqu'à douze reprises le hit de Recode2 est une instance du modèle Logol, avec un rang supérieur à 5 pour onze d'entre eux (le dernier se situe au delà du dixième rang). Pour quatre identifiants, une liaison *wobble* en extrémité empêche Logol de retrouver le hit exact. Enfin pour les deux identifiants restant, le cluster est retrouvé mais il n'y a pas de hit suffisamment proche pour décréter que l'événement *frameshift* -1 est retrouvé avec succès.

## III.4. Pertinence de la détection des pseudo-nœuds avec Logol

Concernant la détection des pseudo-nœuds, nous avons effectué une étude comparative entre les meilleurs hits trouvés par Logol, les hits indiqué dans Recode2, ainsi que les hits produits par un outil effectuant des calculs de minimisation d'énergie. Le logiciel de calcul d'énergie présent dans les résultats qui suivent est DotKnot[35]. C'est un outil récent et adapté pour la recherche de structures nouées, qui se base sur le calcul de RNAfold pour déterminer la structure présente. Voici comment les résultats de DotKnot présentés dans cette section sont obtenus.
Nous avons fourni à DotKnot en entrée une séquence de 49 nucléotides qui correspond à la portion de séquence qui vient juste après la fenêtre glissante pour chaque identifiant de Recode2 où Logol a trouvé des hits exacts ou proches. Comme l'espaceur dans le modèle Logol mesure au minimum un nucléotide la séquence fournie à DotKnot est les 49 nucléotides suivants la fenêtre glissante auquel s'ajoute un nucléotide. Ceci afin de permettre à DotKnot une certaine liberté de repliement sur une portion de séquence conséquente. Le but n'est pas ici de le contraindre à retrouver le hit Logol mais d'observer ce qu'un outil employant un algorithme heuristique pour le calcul du minimum d'énergie trouve dans la même région.
Note: la visualisation présentée des pseudo-nœuds pour le cadre de Recode2 (Figure 11) est obtenue avec PseudoViewer[36] qui emploie la notation *Dot-Bracket.*

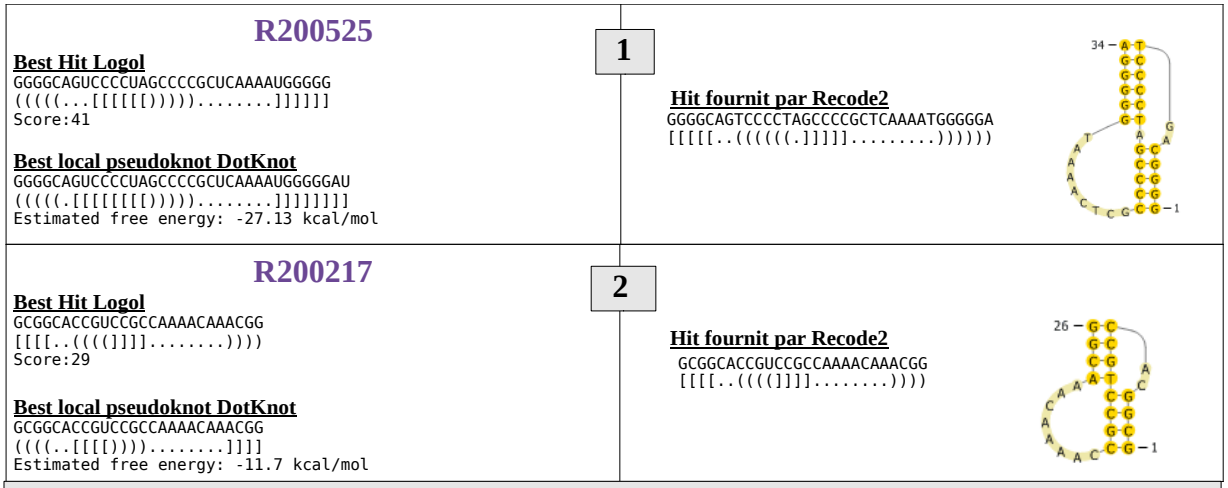



**Figure 11:** *Extrait de l'étude comparative sur les structures initiatrices définies par le modèle Logol, l'outil DotKnot et Recode2*

Pour l'identifiant r200525 (**1**): les trois structures fournies sont légèrement différentes. L'emprise de la tige 1 (GGGGC) est commune mais ce n'est pas le cas pour la tige 2 qui est deux nucléotides plus longue pour DotKnot (avec une liaison *wobble* à l'une des extrémités de S2). Logol et Recode2 ont une tige S2 de même taille mais qui est décalée dans le cas de Logol qui n'autorise pas les liaisons *wobble* à l'extrémité.
Le second identifiant (**2**) présente une correspondance parfaite entre les trois hits.
Au total 9 comparaisons sur 26 présentent la même structure (il a parfois été nécessaire d'aller vérifier non pas avec le meilleur hit Logol mais avec celui correspondant à Recode2 placé dans le top 4). Ce chiffre semble correct d'autant plus qu'à de nombreuses occasions DotKnot semble maximiser la taille de la seconde tige au détriment de la première tige. La présence de liaisons *wobble* dans les extrémités a de nouveau été confirmée avec l'utilisation de cet outil, puisque cet appariement est intervenu à plusieurs reprises (ce qui permet à DotKnot de faire la correspondance avec Recode2, et qui n'est pas le cas de Logol).

## III.5. Limites du score

Les résultats issus du modèle sont encourageants. On retrouve avec une bonne précision les hits fournis par Recode2. Le hit exact est très souvent placé dans les premiers rangs du classement. Dans le cas où l'analyse logol n'est pas parvenue à identifier le hit exact, bien souvent un hit très proche est identifié et obtient là encore un bon classement. La principale différence entre notre approche et les travaux existants c'est que nous n'effectuons pas de véritable calcul de minimisation d'énergie sur la portion de séquence en aval de la fenêtre glissante qui est suspectée contenir une structure d'ARN stable (de type pseudo-nœud). Le calcul du score pour chaque hit va tout de même dans ce sens. Il est relativement simpliste puisqu'il prend uniquement en compte la nature et la longueur des tiges qui composent le pseudo-nœud, mais se révèle malgré tout d'une certaine efficacité. Certaines limites du score semblent provenir du fait qu'il dépende directement de la taille des tiges (le score est d'autant plus élevé que les tiges sont longues). Des anomalies sont ainsi apparues dans le cas de tiges courtes. En effet, certains identifiants de Recode2 définissent un élément stimulateur du *frameshift* -1 avec un pseudo-nœud relativement court (des tiges inférieures à 5-6 nucléotides). On se retrouve dans ces conditions aux bornes inférieures du modèle concernant les limites des tailles des tiges. Les éventuels candidats sur d'autres clusters valides de l'identifiant, qui sont généralement également des pseudo-nœuds courts, entrent donc plus fréquemment en concurrence directe au niveau du classement avec ceux du site du cluster donné par Recode2, ce qui explique pourquoi certains hits « exacts » (ceux conformes aux

données de Recode2) perdent des places dans le classement lorsque l'on prend en compte l'ensemble des clusters valides de l'identifiant. Un autre scénario contribue à pénaliser les petits pseudo-nœuds. Ainsi au sein d'un même cluster, certains hits exacts ne sont pas classés au premier rang, car, comme le modèle est assez lâche au niveau de la boucle L2, il arrive qu'un appariement plus long pour la tige S2 intervienne lorsque l'on allonge la boucle L2, ce qui confère automatiquement un meilleur score pour ce hit. Ce n'est pourtant pas nécessairement le meilleur hit au point de vue de la stabilité énergétique de la structure, mais il faudrait pour cela que le calcul du score prenne en compte les longueurs des boucles comme élément du calcul.
Nous avons, en vue d'améliorer la pertinence du classement, tenté d'utiliser un logiciel de calcul d'énergie mais nous avons été confrontés à plusieurs problèmes : nous avions besoin d'un outil qui prenne en entrée une séquence correspondant au pseudo-nœud (celle trouvée par Logol) ainsi que les contraintes de structures (au format dot-bracket, ajoutée lors du post-traitement) afin de calculer l'énergie de la structure initiatrice instance du modèle Logol. Or certains logiciels ne donnent pas la possibilité de fournir les contraintes en entrée (pknot, pknotRG-fs) ce qui ne permet pas de les utiliser dans cette optique. D'autres le permettent mais n'ont pas la possibilité d'effectuer les calculs sur des structures nouées et donc les pseudo-nœuds (RNAeval). Un outil paraissait répondre à nos attentes, il s'agit de MC-fold[37]. Malheureusement il n'a pas été aisé de le manipuler en ligne de commande pour lui soumettre nos grandes quantités de hits (avec un temps de calcul qui plus est relativement long). De plus cet outil s'est révélé limité dans le cas des pseudo-nœuds courts. Il ne renvoie pas de résultats dès lors que les tiges du pseudo-nœud en entrée sont trop courtes or ce sont justement ces pseudo-nœuds là qui posent le plus de problème dans notre classement et qui auraient mérités d'être affinés en priorité. Une dernière étape a été envisagée mais n'a pas abouti, c'est d'intégrer directement un calcul de ce type dans le post-traitement. Ceci aurait nécessité un temps très important, or notre objectif premier n'était pas de trouver la structure initiatrice précise de chaque site de *frameshift* -1 identifié mais d'utiliser la souplesse de notre outil de modélisation pour pointer de probables sites de décalage de phase en -1, dans des conditions (délais) raisonnables; quitte ensuite à poursuivre, avec des méthodes spécialisées, les investigations sur les candidats mis en évidence.

# IV. Conclusion

Cette étude a consisté à mettre en œuvre la détection d'événements de *frameshift* -1 au moyen de modèles grammaticaux (qualifiés également de « modèles logiques ») avec l'outil de *pattern matching* Logol.
Du point de vue de la pertinence des candidats proposés comme potentiel événements de *frameshift* -1, les résultats obtenus sont encourageants. Les améliorations successives d'un premier modèle Logol ont permis de réduire de manière importante le nombre de candidats dans les séquences analysées. La mise en place additionnelle, en post-traitement, d'un classement basé sur un score jugeant la qualité de la structure initiatrice (le pseudo-nœud) a permis de discriminer les candidats restant et de retrouver avec une assez bonne précision les sites précis où des événements de décalage du cadre de lecture en -1 interviennent réellement (d'après les informations issues de la base de référence Recode2). Le modèle final obtenu apparaît discriminant par rapport aux séquences générées aléatoirement. Cela dit, un certain nombre de points d'améliorations sont perceptibles. Concernant la méthode de validation, le calcul de score, bien que globalement satisfaisant, est imprécis et a montré ses limites pour évaluer des pseudo-nœuds de petite taille. Il pourrait être amélioré, en le rapprochant par exemple du calcul de stabilité énergétique. Le modèle peut également être amélioré. Notamment la prise en compte des liaisons *wobble* fait défaut dans le modèle final proposé, ces liaisons n'étant considérées qu'en

post-traitement au moment du calcul de score. On voit qu'il faut pouvoir les introduire dans le modèle Logol, pour permettre en particulier d'accepter des liaisons *wobble* sur les nucléotides situés en extrémité de tige. Il est possible en Logol de définir de nouveaux morphismes, et donc de définir un morphisme reverse complément « wcw » qui prenne en compte les liaisons *wobble* en plus des liaisons Watson-Crick. Par contre, si on remplace le morphisme Watson-Crick par un morphisme Watson-Crick-Wobble, les proportions de liaisons *wobble*, liaisons plus faibles, devront être contrôlées.

Du point de vue de la pertinence d'utiliser l'outil Logol pour rechercher des motifs biologiques complexes, les résultats obtenus sont également encourageants. Élaborer un modèle pour la détection des événements de *frameshift* -1 représentait une sorte de challenge, dans la mesure où le modèle biologique des *frameshift* -1 est particulièrement riche et que l'outil Logol, en tant qu'outil générique, n'est pas spécifiquement dédié à la recherche de tels événements. Au final, tous les compartiments du modèle biologique ont trouvé leur expression en Logol, même les pseudo-nœuds, bien qu'aucun calcul strict d'énergie libre n'ait été utilisé, ce qui constitue une singularité de l'approche proposée.
Parmi les intérêts de cette approche de modélisation par modèle logique, se situe le fait qu'elle permet de donner une vision globale du modèle, reflétant de façon groupée et structurée l'ensemble des caractéristiques du motif recherché. Les paramètres, rassemblés dans un même descripteur, y sont ainsi plus explicites qu'avec des techniques de « pipelines », et faciles à modifier. Cette propriété peut être particulièrement utile dans le cadre d'une démarche exploratoire où le modèle n'est pas stabilisé. La grande expressivité du langage Logol contribue à limiter l'importance du post-traitement et notamment de l'étape de filtrage -systématiquement présente dans les autres approches de détection de *frameshift* rencontrées.
Lors de cette étude, Logol a également montré ses limites. En particulier, il ne permet pas d'effectuer un calcul de repliement avec minimisation d'énergie, et en l'état actuel des spécifications il n'a pas cette vocation. D'autres limitations de Logol, pointées par ces travaux, ont quant-à elles lieu d'être levées. Une des visées de cette étude était d'ailleurs de participer à la validation et consolidation de Logol. Parmi les points relevés, il est apparu utile de pouvoir de masquer certains détails dans les résultats de l'analyse Logol. En effet, la trace de certaines parties du modèle peut alourdir de manière conséquente le fichier XML de résultat, et rendre son exploitation délicate. Ce fut le cas dans cette application pour la partie du modèle consacrée à la vérification des contraintes de trames (absence de stop dans l'ORF, cf section II.4.b.4). La mémorisation du détail d'analyse de chacun des codons traité par le modèle codonRepeat alourdissant démesurément le fichier de résultat, la vérification de la conformité biologique de la séquence a finalement été reléguée en post-traitement. On gagnerait bien entendu en lisibilité du modèle à laisser cette contrainte au niveau du modèle Logol.
Un autre besoin apparu est la possibilité d'appliquer des contraintes sur un ensemble de variables non consécutives, afin par exemple de demander un pourcentage en GC minimum sur la globalité des deux tiges du pseudo-nœud. Cette possibilité d'appliquer des contraintes globales sur des segments non contigus a été ajoutée à la suite de ces travaux et est désormais opérationnelle.

# Bibliographie


1. Stahl G. & Rousset J.P. *Les surprises du décodage de l'information génétique.* Med Sci (Paris) 1999 ; *15* : 1118-25.

2. Brierley I. *Ribosomal frameshifting on viral RNAs.* J Gen Virol 1995 ; *76* : 1885-92.

3. Belleannée C, Nicolas J. *Logol : Modelling evolving sequence families through a dedicated constrained string language.* Rapport de recherche INRIA n6350 2007 ; *22.1* : 49-62.

4. Baranov P.V, Fayet O, Hendrix R.W. and Atkins J.F. *Recoding in bacteriophages and bacterial IS elements.* Trends Genet. 2006 ; *22* : 174–181.

5. Baranov P.V, Gesteland R.F, Atkins J.F. *Recoding: translational bifurcations in gene expression.* Gene. 2002 ; *286* : 187–201.

6. Jacks T, Madhani H.D, Masiarz H.D, Varmus H.E. *Signals for ribosomal frameshifting in the Rous sarcoma virus gag-pol region.* Cell 1988 ; *55* : 447-58.

7. Bekaert M, Bidou L, Denise A, Duchateau-Nguyen G, Forest JP, Froidevaux C, Hatin I, Rousset JP, Termier M. *Towards a computational model for -1 eukaryotic frameshifting sites.* Bioinformatics. 2003 ; *19(3)* : 327-35

8. Giedroc D.P, Cornish P.V. *Frameshifting RNA pseudoknots: structure and mechanism.*

Virus Res. 2009 ; *139(2)* : 193-208. Review

9. Brierley I, Digard P, Inglis SC. *Characterization of an efficient coronavirus ribosomal frameshifting signal: requirement for an RNA pseudoknot.* Cell. 1989 ; *57(4)* : 537-47.

10. ten Dam E.B, Pleij C.W, Bosch L. *RNA pseudoknots: translational frameshifting and readthrough on viral RNAs.* Virus Genes. 1990 ; *4(2)* : 121-36.

11. Giedroc D.P, Theimer C.A, Nixon P.L. *Structure, stability and function of RNA pseudoknots involved in stimulating ribosomal frameshifting.* J Mol Biol. 2000 ; *298(2)* : 167-85.

12. T. J Macke, D.J Ecker, R.R Gutell, D Gautheret, D.A Case, R Sampath. *RNAMotif, an RNA secondary structure definition and search algorithm. Nucleic Acids* Res.2001 ; *29* : 4724-4735

13. Bairoch A. *PROSITE: a dictionary of sites and patterns in proteins.* Nucleic Acids Res. 1991 ; *19 Suppl* : 2241-5.

14. Nicolas J, Durand P, Ranchy G, Tempel S. & Valin A. *Suffix-tree analyser (STAN): looking for nucleotidic and peptidic patterns in chromosomes.* Bioinformatics 2005 ; *21* : 4408-1610.

15. Searls DB. *String variable grammar: A logic grammar formalism for the biological language of DNA.* The Journal of Logic Programming. 1995 ; *24* : 73-102

16. Theis C, Reeder J, Giegerich R. *KnotInFrame: prediction of -1 ribosomal frameshift events.* Nucleic Acids Res. 2008 ; *18* : 6013-20.

17. Hammell AB, Taylor RC, Peltz SW, Dinman *Identification of putative programmed -1 ribosomal frameshift signals in large DNA databases.* JD.Genome Res. 1999 ; *9(5)* : 417-27.

18. Jacobs JL, Belew AT, Rakauskaite R, Dinman JD. *Identification of functional, endogenous programmed -1 ribosomal frameshift signals in the genome of Saccharomyces cerevisiae.* Nucleic Acids Res. 2007; *35(1)* : 165-74.

19. Moon S, Byun Y, Kim H-J, Jeong S. and Han K. *Predicting genes expressed via 1 and +1*

*frameshifts.* Nucleic Acids Res. 2004 ; *32 :* 4884–4892.

20. Jacobs J.L, Belew A.T, Rakauskaite R. and Dinman J.D. *Identification of functional, endogenous programmed 1 ribosomal frameshift signals in the genome of Saccharomyces cerevisiae.* Nucleic Acids Res. 2005 ; *35* : 165–174.

21. Jacks T, and H.E Varmus. *Expression of the Rous Sarcoma Virus pol gene by ribosomal frameshifting.* Science. 1985 ; *230* : 1237–1242.

22. Dinman J.D, T Icho, and R.B Wickner. *A 11 ribosomal frameshift in a double-stranded RNA virus forms a Gag-pol fusion protein.* Proc. Natl. Acad. Sci. 1991 ; *88* : 174–178.

23. Brierley I.A, Jenner A.J, and Inglis S.C. *Mutational analysis of the "slippery sequence" component of a coronavirus ribosomal frameshifting signal.* J. Mol. Biol. 1992 ; *227*: 463–479.

24. Dinman J.D. and R.B Wickner. *Ribosomal frameshifting efficiency and Gag/Gag-pol ratio are critical for yeast M1 double-stranded RNA virus propagation.* J. Virol. 1992 ; *66* : 3669–3676.

25. Kontos H, Napthine S, Brierley I. *Ribosomal pausing at a frameshifter RNA pseudoknot is sensitive to reading phase but shows little correlation with frameshift efficiency.* Mol Cell Biol. 2001 ; *21(24)* : 8657-70

26. Napthine S, Liphardt J, Bloys A, Routledge S, Brierley I.*The role of RNA pseudoknot stem 1 length in the promotion of efficient -1 ribosomal frameshifting*.J Mol Biol. 1999 ; *288(3)* :305-20.

27. Liphardt J, Napthine S, Kontos H, Brierley I. *Evidence for an RNA pseudoknot loop-helix interaction essential for efficient -1 ribosomal frameshifting.* J Mol Biol. 1999 ;*288(3)* : 321-35.

28. Rivas E. and Eddy S.R. *A dynamic programming algorithm*

*for RNA structure prediction including pseudoknots.* J. Mol. Biol. 1999 ; *285* :2053–2068.

29. Reeder J, Steffen P. and Giegerich R. *pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows.* Nucleic Acids. Res. 2007 ; *35 (Suppl. 2)* : W320–W324.

30. Gruber AR, Lorenz R, Bernhart SH, Neuböck R, Hofacker IL. *The Vienna RNA websuite.* Nucleic Acids Res. 2008 ; *36(Web Server issue)* : W70-4

31. Bekaert M, Andrew E, Zhang Y, Gladyshev V. N, Atkins J.F and P.V Baranov. *Recode-2: new design, new search tools, and many more genes* Nucleic Acids Res. 2010; *Vol. 38* ; D69–D74

32. Moon S, Byun Y, Han K.Comput Biol Chem. *FSDB: a frameshift signal database.* 2007 ; *31(4)* : 298-302.

33. Van Batenburg F.H.D , Gultyaev A.P , Pleij C.W.A, Ng J and J Oliehoek. *Pseudobase: a database with RNA pseudoknots.* Nucl. Acids Res. 2000 ; *28(1)* : 201-204.

34. J.P Forest. *Modélisation et détection automatique de sites de décalage de cadre en -1 dans les génomes eucaryotes.* 2005 ; Thèse.

35. Sperschneider J, Datta A. *DotKnot: pseudoknot prediction using the probability dot plot under a refined energy model.* Nucleic Acids Res. 2010 ; *38(7)* : e103.

36. Byun Y and Han K. *PseudoViewer: web application and web service for visualizing RNA pseudoknots and secondary structures.* Nucl. Acids Res. 2006 ; *Vol.34* : W416-W422,

37. Parisien M, Major F. *The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data.* Nature. 2008 ; *452(7183)* : 51-5.

**Annexe 1 -** Modèle final complet de frameshift -1 – Graphique et grammaire

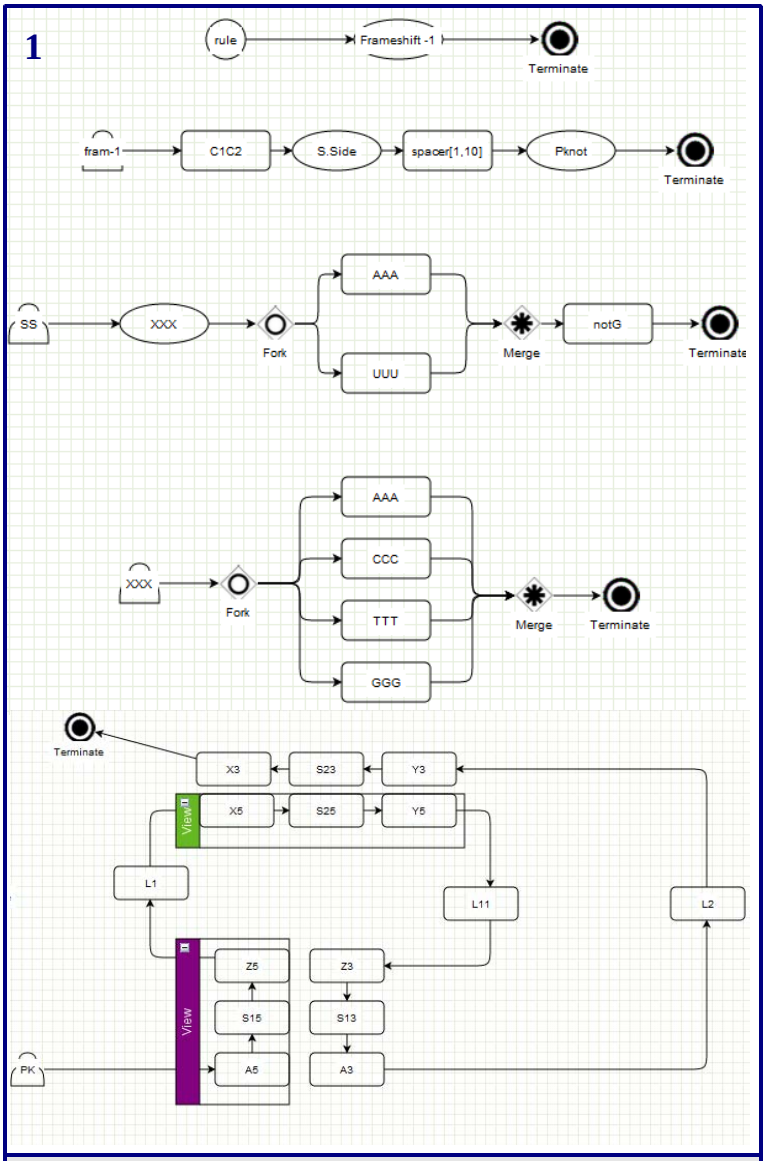
= *modèle final produit lors de cette étude*

1. Modèle graphique LogolDesigner

2. Conversion du modèle en grammaire Logol

3. Modèle épuré, grammaire Logol décrite par l'utiliseur

**1**




**2**

```
mod1(LOGOLVAR1)==>LOGOLVAR2%CC%:{#[2,2]},mod2%SS%
(LOGOLVAR1),LOGOLVAR3%spacer%:{#[1,10]},mod3%PK%(LOGOLVAR1)
mod4(LOGOLVAR4)==>(("aaa")|("ccc")|("ttt")|("ggg"))
mod2(LOGOLVAR5)==>mod4%xxx%(LOGOLVAR5),(("aaa")|("ttt")),! "g":{#[1,1]}
mod3(LOGOLVAR6)==>(LOGOLVAR7%A5%:{#[1,1],_LOGOLVAR8%A5%},LOGOLVAR9%S%:
{#[2,14],_LOGOLVAR10%S15%},LOGOLVAR11%Z5%:{#[1,1],_LOGOLVAR12%Z5%}):
{%"gc":50},LOGOLVAR13%L1%:{#[1,5]},(LOGOLVAR14%X5%:
{#[1,1],_LOGOLVAR15%X5%},LOGOLVAR16%S2%:
{#[1,6],_LOGOLVAR17%S25%},LOGOLVAR18%Y5%:{#[1,1],_LOGOLVAR19%Y5%}):
{%"gc":50},LOGOLVAR20%L11%:{#[0,4]},-"wc" LOGOLVAR21%Z3%:{?
LOGOLVAR12%Z5%},-"wc" ?LOGOLVAR10%S15%:{p$[0,34]},-"wc" LOGOLVAR22%A3%:{?
LOGOLVAR8%A5%},LOGOLVAR23%L2%:{#[4,40]},-"wc" LOGOLVAR24%Y3%:{?
LOGOLVAR19%Y5%},-"wc" ?LOGOLVAR17%S25%:{p$[0,34]},-"wc" LOGOLVAR25%X3%:{?
LOGOLVAR15%X5%}
mod1(LOGOLVAR26)==*>SEQ1
```

**3**

```
mod4()==>(("aaa")|("ccc")|("ttt")|("ggg"))
mod2()==>mod4(),(("aaa")|("ttt")),! "g":{#[1,1]}
mod3()==>(A5:{#[1,1],_SA5},S15:{#[2,14],_STEM15},Z5:{#[1,1],_SZ5}):
{%"gc":50},L1:{#[1,5]},(X5:{#[1,1],_SX5},S25:{#[1,6],_STEM25},Y5:
{#[1,1],_SY5}):{%"gc":50},L11:{#[0,4]},-"wcw" Z3:{?SZ5,_SZ3},-"wc" ?STEM15:
{_STEM13}:{p$[0,34]},-"wcw" A3:{?SA5,_SA3},L2:{#[4,40]},-"wcw" Y3:{?
SY5,_SY3},-"wc" ?STEM25:{_STEM23}:{p$[0,34]},-"wcw" X3:{?SX5,_SX3}
mod1()==>CC1:{#[2,2]},mod2(),SPACER2:{#[1,10]},mod3()
mod1()==*>SEQ1
```

## **Annexe 2 –** Modèle « post-final » *modèle amélioré produit après l'étude*

Postérieurement à l'étude décrite dans ce document, et ayant abouti au « modèle final » donné en Annexe 1, des modifications ont été apportées à Logol. Elles ont permis de mettre en œuvre certaines améliorations envisagées dans la conclusion du présent document.

Ainsi, il est devenu possible en Logol d'appliquer des **contraintes globales** (contraintes sur des variables disjointes). Un **morphisme *wobble*** à également été élaboré (« wcw », pour Watson-Crick-Wobble). Il autorise les appariements G-U en plus de G-C et A-U.

Cela dit, le fait d'autoriser les liaisons *wobbles* sur l'ensemble des deux tiges s'est montré problématique, car il augmente de manière considérable le nombre de matchs (plusieurs milliers de matchs par identifiant). Une solution intermédiaire a été choisie. Il s'agit d'autoriser les liaisons *wobbles* aux extrémités des tiges S1 et S2 et de continuer à les inclure dans les *mismatchs* sur l'intérieur des tiges (les 34% de *mismatch*/*wobble*).

Le pourcentage en GC n'est plus applicable directement avec l'utilisation des « vues » car les G présents aux extrémités peuvent désormais être associés à un U dans le cas d'une liaison *wobble*. Sur une tige de 4 nucléotides, ce nucléotide représente 25% . Il n'est donc pas négligeable. Ceci amène à utiliser la nouvelle fonctionnalité de Logol qui permet d'appliquer des contraintes sur des variables disjointes. Ce calcul, nommé « *control* », est calculé à l'issu de la phase de recherche des instances du modèle et seuls les matchs vérifiant ce contrôle figurent dans le fichier XML de résultats de l'analyse Logol. On peut le voir comme un post-filtre intégré au modèle Logol.

On obtient ainsi le nouveau modèle Logol de Frameshift-1, qu'on appellera « modèle amélioré »:

```
#définition du morphisme wobble
def:{
morphism(wcw,a,t)
morphism(wcw,t,a)
morphism(wcw,c,g)
morphism(wcw,g,c)
morphism(wcw,g,t)
morphism(wcw,t,g)
}
#définition des contrôles
controls:{
% "c"[mod3.SA5,mod3.STEM15,mod3.SZ5,mod3.SZ3,mod3.STEM13,mod3.SA3]>=25
% "c"[mod3.SX5,mod3.STEM25,mod3.SY5,mod3.SY3,mod3.STEM23,mod3.SX3]>=25
}
#définition du modèle Logol

mod4()==>(("aaa")|("ccc")|("ttt")|("ggg"))

mod2()==>mod4(),(("aaa")|("ttt")),! "g":{#[1,1]}

mod3()==>(A5:{#[1,1],_SA5},S15:{#[2,14],_STEM15},Z5:{#[1,1],_SZ5}):{%"gc":50},
         L1:{#[1,5]},
         (X5:{#[1,1],_SX5},S25:{#[1,6],_STEM25},Y5:{#[1,1],_SY5}):{%"gc":50},
         L11:{#[0,4]},
         -"wcw" Z3:{?SZ5,_SZ3},-"wc" ?STEM15:{_STEM13}:{p$[0,34]},-"wcw" A3:{?SA5,_SA3},
         L2:{#[4,40]},
         -"wcw" Y3:{?SY5,_SY3},-"wc" ?STEM25:{_STEM23}:{p$[0,34]},-"wcw" X3:{?SX5,_SX3}

mod1()==>CC1:{#[2,2]},mod2(),SPACER2:{#[1,10]},mod3()

mod1()==*>SEQ1
```

Le pseudo-nœud est modélisé dans mod3. On continue d'appliquer un pourcentage en GC de 50% sur l'aller des tiges S1 et S2 afin de ne pas sélectionner trop de matchs lors de l'analyse Logol. On autorise les liaisons *wobbles* sur les extrémités grâce au morphisme « wcw ». Par contre les éventuels appariements *wobble* continuent d'être pris en compte dans le pourcentage de *mismatch* autorisés sur l'intérieur de S1 et S2. C'est pour cela que c'est le morphisme « wc » qui est

appliqué pour cette portion de la tige retour (STEM13, STEM23).
Les contrôles permettent de définir un pourcentage minimum que l'ensemble de la tige aller + retour doit impérativement satisfaire. Il n'est pas possible de garder un pourcentage en GC puisque les G peuvent être appariés avec des U aux extrémités des tiges. On impose donc que l'ensemble aller/retour de chaque tiges contiennent 25% de C, c'est à dire 50 de GC puisque un C est systématiquement apparié avec un G. Note : il n'est pas pertinent d'imposer 25% de C dans une vue pour les aller ou retour de tiges puisque une tige dont l'aller aurait 0% de C peut très bien être un candidat intéressant si elle possèdent beaucoup de G qui sont appariés avec des C.
Ce modèle permet de récupérer certains candidats fournis par Recode2 que le précédent modèle ne permettait pas de retrouver car il était imprécis au niveau des liaisons *wobbles*.

| Statistiques | Modèle final de fin d'étude | Modèle amélioré (wobble+control) |
|---|---|---|
| Nombre de matchs | 2534 | **3806** |
| Nombre de clusters | 122 | **123** |
| Nombre de cluster valide | 83 | **77** |

Il y a 46 cluster "non valides" (majoritairement pour cause de stop précédent la fenêtre glissante, ou parfois pour cause d'absence de start en amont de la fenêtre glissante).
On retrouve globalement plus de matchs qu'avec le précédent modèle, ce qui est logique du fait que le modèle est plus souple au niveau des extrémités des tiges, mais par contre il génère moins de clusters valides. Ceci est probablement dû à la présence du pourcentage en C appliqué à l'ensemble de chaque tige qui a dû éliminer quelques clusters (les 122 clusters du modèle 1 ne sont pas identiques aux 123 clusters du modèle 2).
Cette modification du modèle permet de récupérer cinq matchs officiel de Recode2. Les sept

| Identifiant | cluster(s) valide(s) | Présence du Hit Recode | Rang cluster | Rang global |
|---|---|---|---|---|
| r200015 | 2 | Oui | 1 | 1 |
| r200090 | 1 | Oui | >10 | >10 |
| r200163 | 10 | Non % C < 25% | 1 | 1 |
| r200179 | 2 | Oui | 1 | 1 |
| r200191 | 3 | Oui | 1 | 4 |
| r200197 | 2 | Non ?? | 1 | >10 |
| r200199 | 4 | Oui | 1 | 1 |
| r200208 | 4 | L1 absente | 1 | 1 |
| r200209 | 3 | Non % C < 25% | 1 | 1 |
| r200210 | 5 | L1 absente | 1 | 1 |
| r200211 | 1 | Oui | 1 | 1 |
| r200216 | 1 | Oui | 7 | 7 |
| r200217 | 1 | Oui | 1 | 1 |
| r200221 | 4 | Oui | 1 | 1 |
| r200518 | 3 | Oui | 1 | 1 |
| r200525 | 1 | Oui | >10 | >10 |
| r200528 | 2* | Oui | 1 | >10 |
| r200533 | 2 | Oui | 2 | 3 |
| r200550 | 2 | Oui | 1 | 1 |
| r200552 | 3 | Oui | 1 | 1 |
| r200553 | 3 | Oui | 1 | 4 |
| r200558 | 3 | Oui | 6 | >10 |
| r200562 | 8 | Non (beaucoup d'erreur) | 1 | 1 |
| r200564 | 3 | Oui | 1 | 1 |
| r201052 | 0 | Non % C < 25% | 1 | 1 |
| r201056 | 0 | Oui | 2 | 2 |
| r200200 | 2 | x | x | x |
| r200203 | 3 | x | x | x |
| r200219 | 3 | x | x | x |
| r200517 | 2° | x | x | x |

restants possèdent d'autres incompatibilités: un pourcentage en GC trop faible, une boucle L1 absente ou bien la présence de trop nombreuses erreurs dans les liaisons des tiges.