République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara – Boumerdes

Faculté des Sciences
Département d'Informatique
Laboratoire d'Informatique Fondamentale et Appliquée

Mémoire

Pour l'obtention du diplôme de

Magister en Informatique
Option : Informatique Fondamentale

Présenté et soutenu publiquement par
ISHAK BOUSHAKI Saida

Thème :

Modélisation et simulation des processus biologiques dans le π-calcul

Cas : Régulation de la transcription de gènes

JURY

M. Nadjib Badache	**Professeur**	**USTHB**	**Président**
M. Mohamed Ahmed Nacer	**Professeur**	**USTHB**	**Examinateur**
M. Djamel Tanjaoui	**Chargé de recherche**	**CERIST**	**Examinateur**
M. Mohamed Mezghiche	**Professeur**	**UMBB**	**Promoteur**

ANNEE -2006/2007-

Remerciements

Je tiens à remercier chaleureusement mon enseignant et promoteur, M^{r} MEZGHICHE de m'avoir dirigé et proposé mon sujet de recherche, pour m'avoir aidée à faire mes premiers pas dans la recherche, pour sa continuelle disponibilité, ses conseils judicieux, ses suggestions et critiques, ses encouragements incessants, son comportement amical, sa cmpréhension. Je le remercie également pour le fait qu'il n'a jamais hésité une seule fois à mettre à ma disposition tous les moyens dont il dispose (livres, articles de recherche, logiciels …) dont j'ai eu grandement besoin pour pouvoir mener à bien ce travail.

Un remerciement tout particulier à mes enseignants : M^{s} Harzallah Abdekarim, Mezghiche Mohamed, Khalifa Mohamed, Ahmed Nacer, Ahmed Ouamer pour leur effort pour nous communiquer leurs savoir et expérience.

Mes vifs remerciements sont aussi dirigés vers :

- M^{r} Nadjib Badache, professuer à l'U.S.T.H.B, pour l'honneur qu'il me fait en acceptant de présider le jury,
- M^{r} Mohamed Ahmed Nacer, professeur à l'U.S.T.H.B et M^{r} Djamel Tandjaoui chargé de recherche au CERIST, pour l'intérêt qu'ils portent à mon travail en acceptant d'être membre du jury.

Je tiens également à remercier tous ceux qui m'ont encouragée et aidée de près ou de loin tout au long de mon travail, et tout particulièrement, ma collègue et amie, M^{elle} Atika Badaoui, pour sa compagnie, ses critiques, ses conseils, sans oublier M^{r} Imache et M^{me} Hadjidj.

Abstract

In this memory we have studied the biologic reaction modelling and their simulation in the stochastic π-calculus.

The biologic processes are considered as parallel processes. First, they are expressed in the stochastic π-calculus languague. Then, they are simulated with tools as BioSpi and SPiM, wich they are based on the algorithm of Gillespie.

To illustrate this method we have considered the exemple of reaction of gene transcription regulation by feed back.

After its modelling in the stochastic π-calculus, we have exécuted the program with the tool SPiM.

The results of the simulation give some useful information on this regulation reaction.

Key words:

Algebra of processes, π-calculus, modal logic, temporal logic, π-logic, logic ACTL, HD-automata, model checking, biologic processes, modelling, simulation, regulation of gene transcription.

ملخص المذكرة

في هذه المذكرة قمنا بدراسة نموذج للتفاعلات البيولوجية و تمثيلها في الحساب π العشوائي.
التفاعلات البيولوجية, اعتبرت كعمليات متوازية, أولا تم التعبير عنها في الحساب π العشوائي, ثم تم و الآلة π العشوائية اللتان تستندان على تمثيلها بمساعدة وسائل مثل الحساب π العشوائي البيوكيميائي خوارزمية جيليسبي.
من أجل إثراء هذه الطريقة قمنا بدراسة حالة و هي تنظيم استنساخ المورثات عن طريق مفعول رجعي. بعد إنجاز نموذج للعملية في الحساب π العشوائي, قمنا بإنجاز البرنامج بواسطة الآداة الآلة π العشوائية.
نتائج التمثيل المتحصل عليها أعطت لنا معلومات قيمة حول تفاعل التنظيم.

كلمات المفتاح:

جبر العمليات, الحساب π, المنطق المميز, المنطق الزمني, المنطق π, منطق حساب الفعل وفق شجرة منطقية,الآلية ذات التاريخ المرتبط, التحقق من النموذج, عملية بيولوجية, تمثيل, تنظيم استنساخ المورثات.

Résumé

Dans ce mémoire nous avons étudié la modélisation des réactions biologiques et leur simulation dans le π-calcul stochastique. Les processus biologiques, considérés comme des processus parallèles, sont d'abord exprimés dans le langage du π-calcul stochastique, ensuite sont simulés à l'aide d'outils comme BioSpi et SPiM, basés sur l'algorithme de Gillespie.

Pour illustrer cette méthode nous avons considéré l'exemple de réaction de régulation de la transcription de gènes par rétroaction.

Après sa modélisation dans le π-calcul stochastique, nous avons exécuté le programme obtenu à l'aide d'outil SPiM. Les résultats de la simulation obtenus donnent des informations utiles sur cette réaction de régulation.

Mots clés :

Algèbre des processus, π-calcul, logique modale, logique temporelle, π-logique, logique ACTL, HD-automate, model checking, processus biologiques, modélisation, simulation, régulation de la transcription de gènes.

Table des matières

Liste des tables

Liste des figures

Liste des abréviations

A	Adénine
ACP	Algebra of Communicating Process
ACTL	Action Computation Tree Logic
ADN	Acide DésoxyriboNucléique
ARN	Acide RiboNucléique
ARN_m	Acide RiboNucléique messager
ARN_r	Acide RiboNucléique ribosomale
ARN_t	Acide RiboNucléique de transfert
BioSpi	Biochemical Stochastic pi-calculus
C	Cytosine
CCS	Calculus of Communicating Systems
CSP	Communicating Sequential Process
G	Guanine
GUI	Graphical User Interface
HAL	HD Automata Laboratory
HD-automate	History Dependent automata
HML	Hennessy Milner Logic
hnf	head normal form
LTS	Labelled Transition System
SPiM	The Stochastic Pi Machine
T	Thymine
U	Uracile
URN	unit-interval Uniform Random Number generator
XML	Extensible Markup Language

Liste des symboles

$\mid$	Parallèle
$+$	Somme
$\xrightarrow{\alpha}$	Transitions
$\overset{\alpha}{\Rightarrow}$	Transitions faibles
$\dot{\sim}$	Relation de bisimilarité tardive
$::=$	Symbole de définition syntaxique
$=$	Egalité
$\neq$	Inégalité
$\equiv$	Congruence structurelle
$\forall$	Quelque soit
$\exists$	Il existe
$\sim$	Relation de congruence
$\wedge$	Le ET logique
$\downarrow$	Observable
$\notin$	N'appartient pas
$\in$	Appartient
$\xrightarrow{}_{E}$	Transition d'une sémantique précoce
$\sim_{E}$	Congruence précoce
$\dot{\sim}_{E}$	Bisimilarité précoce
$\dot{\sim}_{B}$	Bisimilarité barbelée
$\sim_{B}$	Congruence barbelée
$\dot{\sim}_{O}$	Bisimolarité ouverte
$\dot{\approx}$	Bisimilarité faible
$\cap$	Intersection
$\Leftrightarrow$	Equivalence
ø	Ensemble vide
$\sum$	La somme
$\models$	Relation de satisfaction
$\not\models$	Relation de satisfaction n'est pas vérifiée
$\vee$	OU logique
$\neg$	La négation logique
*	Le produit

Introduction générale

Les processus biochimiques mettent en action un ensemble important de protéines responsables des fonctions majeures des cellules vivantes.

La modélisation formelle de ces processus est primordiale pour comprendre le développement des réactions biochimiques et simuler leur comportement. C'est aussi un moyen privilégié permettant de vérifier quelques propriétés de ces réactions.

Plusieurs systèmes formels (base de données, λ-calcul, logique linéaire, réseaux de Petri, diagramme d'états,..) offrent la possibilité de modélisation des phénomènes biologiques.

Le choix d'un modèle est basé sur un ensemble de critères à savoir la pertinence (un modéle pertinent doit capturer deux propriétés éssentielles du système biomoléculaire à modéliser dans une seule vue unifiée : son organisation moléculaire et son comportement dynamique), la calculabilité, la compréhensibilité et l'extensibilité. Cependant, aucun des systèmes cités au paravant n'assurent l'intégrité de ces critères. Les bases de données et le diagramme d'états ne manipulent que les aspects qualitatifs, tandis que les réseaux de Petri manipulent les aspects quantitatifs mais leur modèle n'est pas pertinent. La recherche d'un formalisme répondant au mieux à ces contraintes a permis de montrer l'importance du π-calcul pour la modélisation des processus biologiques.

Dans le cadre de notre travail, nous allons étudier la spécification des processus biologiques et leur simulation dans le π-calcul.

Pour accomplir ce travail nous avons décomposé notre étude en deux parties.

Dans la première partie nous introduirons les algèbres des processus (chapitre1), en présentant l'algèbre élémentaire des processus, particulièrement le CCS (Calculus of Communicating Systems) dont l'extension a été la définition du π-calcul. Nous décrirons la syntaxe, la sémantique, les différentes extensions, et aussi la théorie algébrique du π-calcul (chapitre2). Nous terminerons la première partie par la présentation d'un model checking pour le π-calcul, basé sur HD-automate (History Dependant automata), la π-logique et la logique ACTL (Action Computation Tree Logic) (chapitre3). L'ACTL permet d'inférer les propriétés sur des processus du π-calcul. Nous décrirons aussi un environnement de vérification appelé HAL (History dependant Automata Laboratory).

La deuxième partie de notre étude présente l'application du π-calcul dans le domaine de la biologie.

Comme nous avons besoin des concepts précis, nous avons trouvé nécessaire de rappeler d'abord quelques notions de la biologie, tels que les constituants principaux d'une

cellule (protéine, ADN, ARN, glucides, lipides, …), la synthèse d'une protéine, et aussi quelques notions sur la régulation des protéines (chapitre 4).

Nous présenterons ensuite dans le chapitre 5, le système biomoléculaire dans le π-calcul et aussi les techniques de modélisation des processus biologiques par le π-calcul. Nous décrirons également l'algorithme utilisé pour la simulation d'un programme codé par le π-calcul stochastique. Nous terminerons par la présentation de deux simulateurs (BioSpi et SPiM) implémentés pour exécuter des programmes codés par le π-calcul stochastique.

Dans le dernier chapitre (chapitre 6), nous décrirons le processus biologique de régulation de la transcription de gènes par rétroaction positive, et nous donnerons sa modélisation dans le π-calcul stochastique. Nous utiliserons ensuite l'outil SPiM pour la simulation de notre modélisation de la réaction du processus biologique.

Partie I

Etat de l'art :

Spécification et vérification en π-calcul

Chapitre 1
Algèbre des processus

1 Introduction

L'algèbre des processus est l'étude du comportement des systèmes parallèles et répartis par des moyens algébriques. Elle permet d'exprimer la composition parallèle, la composition alternative et la composition séquentielle (ordonnancement) entre les éléments d'un système donné [5].

Dans une algèbre des processus, nous pouvons raisonner en utilisant les propriétés algébriques de ce système pour vérifier ou pour démontrer si certaines propriétés sont satisfaites ou non [5].

Parmi les algèbres des processus existants, on peut citer CSP (Communicating Sequential Process) définis par Hoare [38], CCS (Calculus of Communicating Systems) définis par Milner [48], ACP (Algebra of Communicating Process) définis par Bergstra et Klop [6] et le π-calcul défini par Robin Milner, Joachim Parrow et David Walker [51].

Ce chapitre concernera l'introduction des algèbres des processus. Nous commencerons par présenter la syntaxe et la sémantique de l'algèbre élémentaire des processus. Nous étudierons ensuite une algèbre des processus fondamentale appelée CCS (Calculus of Communicating Systems) qui est à la base de la définition du π-calcul. Nous terminons ce chapitre par la présentation de la propriété de base de ce dernier calcul.

2 Algèbre élémentaire des processus

Nous présenterons dans ce qui suit la syntaxe et la sémantique de l'algèbre élémentaire des processus [48].

2.1 Syntaxe :

Convention :

Les processus sont représentés par des lettres majuscules et les actions par des lettres minuscules.

P ::= act.P | P|P | P + P

L'interprétation des différents opérateurs est la suivante :

- **act.** est l'opérateur de préfixage. Le processus **act.P** peut exécuter l'action **act** pour se comporter ensuite comme le processus **P**.
- « | » est l'opérateur de composition parallèle. Si on a un processus **P** décrit par **act.P′** alors le processus **P | Q** peut exécuter l'action **act** pour se comporter ensuite comme **P′ | Q**.
- « + » est l'opérateur de choix non déterministe. Le processus **P + Q** se comporte soit comme **P**, soit comme **Q**.

2.2 Sémantique

La sémantique opérationnelle d'une algèbre de processus associe un modèle à chaque terme de cette algèbre, ce modèle se présente sous la forme d'un système de transitions étiquetées (LTS) ou automate à états finis.

La sémantique opérationnelle exploite une relation de transition notée "$\rightarrow$" qui met en relation les éléments du langage. Cette relation de transition est utilisée dans les règles d'inférence qui décrivent l'évolution du système modélisé.

Les règles de transition élémentaires sont :

$$\frac{}{act.P \xrightarrow{act} P}$$

$$\frac{P \xrightarrow{act} P'}{P + Q \xrightarrow{act} P'} \quad \text{et} \quad \frac{Q \xrightarrow{act} Q'}{P + Q \xrightarrow{act} Q'}$$

$$\frac{P \xrightarrow{act} P'}{P|Q \xrightarrow{act} P'|Q} \qquad \frac{Q \xrightarrow{act} Q'}{P|Q \xrightarrow{act} P|Q'} \qquad \frac{P \xrightarrow{act_1} P' Q \xrightarrow{act_2} Q'}{P|Q \xrightarrow{act_1 \wedge act_2} P'|Q'}$$

3 Algèbre CCS pour les processus communiquants

3.1 Présentation

L'algèbre CCS (Calculus of Communicating Systems) [48] décrit les processus communicants qui envoient et reçoivent des messages de manière synchrone.

Le formalisme CCS décrit les systèmes communicants comme des ensembles d'automates non-deterministes (souvent appelés agents ou processus) qui interagissent par le biais de synchronisations.

La syntaxe et la sémantique de CCS englobent celles des algèbres des processus élémentaires. Il faut y ajouter toutefois quelques nouveaux concepts :

- il y a deux types d'actions : les unes sont silencieuses, internes au processus et ne produisent rien de visible depuis l'extérieur du processus. Elles sont notées τ .
  Les autres sont visibles depuis l'environnement du processus : ce sont des actions de communication. Cet ensemble d'actions est noté ***l.***
- toutes les actions appartenant à l'ensemble ***l*** sont couplées à des actions inverses, appartenant aussi à ***l***. L'action a est associée à une action inverse a′ notée aussi $\overline{a}$.
  Ainsi, si a signifie "*émettre le message a*", a′ signifiera "*recevoir le message a*".
  Il est aussi possible d'utiliser la notation suivante : **?** pour recevoir et **!** pour émettre.
- deux processus communicants sont égaux si et seulement si leurs traces[1]
  sont égales et aussi leurs possibilités de synchronisation (par envoie de messages) sont identiques. C'est la notion de bisimulation forte notée « $\sim$ ».

La bisimulation se définit formellement de la manière suivante :

P est relié à Q par une relation de bisimulation, notée « P $\sim$ Q », si et seulement si :

$\left(P \xrightarrow{act} P'\right) \Rightarrow (\exists Q', Q \xrightarrow{act} Q' \text{ et } P' \sim Q')$

$\left(Q \xrightarrow{act} Q'\right) \Rightarrow (\exists P', P \xrightarrow{act} P' \text{ et } P' \sim Q')$

On dit que deux agents sont bisimilaires, s'ils sont reliés par une relation de bisimulation.

En plus des règles de sémantique déjà rencontrées en 2.2, il faut ajouter la règle spécifique à la communication entre deux processus :

$$\frac{P_1 \xrightarrow{a} P_1' \; P_2 \xrightarrow{\overline{a}} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'}$$

[1] Une trace du comportement d'un processus est une séquence finie de symboles qui enregistrent les événements que le processus a pris part jusqu'à moment dans le temps.

R. Milner [51] étend le système CCS en introduisant la notion de mobilité[2]. La problématique étudiée est donc la communication de plusieurs processus au travers d'un réseau dont la topologie change dynamiquement.

Les processus dans le π-calcul communiquent entre-eux par des canaux explicitement nommés. Ils peuvent, tout comme en CCS se synchroniser, mais aussi changer dynamiquement le réseau sur lequel ils communiquent en s'échangeant des noms de canaux.

Le π-calcul est un formalisme très expressif et adapté pour décrire les processus communicants. Il permet d'encoder des structures de données [49], le λ-calcul [50] et les communications d'ordre supérieur (lorsque des processus sont transmis à la place des noms) [71]. En outre, il peut être utilisé comme un outil de raisonnement sur des langages orientés objet [82].

4 Conclusion

L'algèbre des processus permet de modéliser de manière formelle des processus qui s'exécutent en parallèle et qui communiquent entre-eux.

Le réseau de communication qui matérialise les liens entre les processus peut être dynamique.

Le π-calcul peut être ainsi considéré comme le fondement de la programmation parallèle au même titre que le λ-calcul est le fondement de la programmation fonctionnelle.

Le chapitre suivant est consacré pour présenter ce π-calcul.

[2] Des systèmes dont le nombre de processus ainsi que les liens de communication entre processus peuvent varier au cours du temps.

Chapitre 2

Le π-calcul

I- Le π-calcul

1 Introduction

Le π-calcul [49] [51] [58] [22] [45] [62] [74] est une extension de CCS. Il permet de modéliser les systèmes d'agents interagissant et dont la configuration varie continuellement. Dans le présent chapitre nous décrirons sa syntaxe et sa sémantique. Les extensions du π-calcul seront aussi introduites. Nous terminerons ce chapitre par la présentation d'une théorie algébrique pour ce calcul.

2 Syntaxe du π-calcul

On notera dans toute la suite x,y,z,u,v,…les noms des canaux et par N l'ensemble de tous ces noms : N = { x,y,z,u,v,…}.

On notera par A ,B,…des processus ou bien des agents.

La syntaxe du π-calcul [58] est donnée par la grammaire présentée dans la table 1 :

Préfixes	α	::=	$\bar{a}x$	Sortie
			a(x)	Entrée
			τ	Silencieux
Processus	P	::=	0	vide
			α.P	préfixe
			P+P	Somme
			P | P	Parallèle
			If x = y then P	Match
			If x ≠ y then P	Mismatch
			(vx) P	Restriction
			$A(y_1,y_2,...y_n)$	Identificateur
Définitions			$A(x_1,x_2,...x_n) = P$	(où $i \neq j \rightarrow x_i \neq x_j$)
Réplication			!P = P | !P	

Table1 : La syntaxe du π-calcul

Cette syntaxe formelle peut être interprétée par :

1. **Le processus vide 0 :** c'est un processus qui ne fait rien.
2. **Un préfixe de sortie ($\bar{a}x$.P) :** c'est un processus qui envoie le nom x sur le canal a et se comporte ensuite comme P. a est un sujet positif et x est l'objet de l'action , $\bar{a}x$ est appelé *préfixe négatif*.
3. **Un préfixe d'entrée (a(x).P) :** c'est un processus qui reçoit un nom sur le canal a, ensuite se comporte comme P dans lequel x est remplacé par le nom reçu . a est un sujet négatif et x est l'objet de l'action, a(x) est appelé *préfixe positif*.
4. **Un préfixe silencieux (τ.P) :** c'est un préfixe qui fait une transition silencieuse τ (tau) et se comporte ensuite comme P.
5. **Une somme (P+Q) :** c'est un processus qui se comporte comme P ou comme Q.
6. **Une composition parallèle (P\|Q) :** c'est un processus représentant la composition parallèle de P et Q. Les processus P et Q peuvent évoluer séparément, de plus les communications entre P et Q peuvent se produire si l'un des processus émet sur un

canal et l'autre reçoit sur le même canal .Ces communications se traduisent par des transitions silencieuses τ.

7. **Le match (if x=y then P) :** c'est un processus qui se comporte comme P si x=y , sinon ne fait rien .
8. **Le mismatch (if x ≠ y then P) :** c'est un processus qui se comporte comme P si x et y n'ont pas la même valeur, sinon ne fait rien .
9. **Une restriction ((vx)P) :** c'est un processus qui agit comme P mais le nom x est privé à P et ne peut pas être utilisé comme canal dans les communications avec l'environnement du processus (*ie*, avec les autres processus) .x est donc lié dans P (*ie*, x est un nom local dont la portée est P)
10. **Un identificateur de processus (** $A(y_1, y_2, ...y_n)$ **)** : tel que n est l'arité du processus defini par une équation de la forme P= $A(x_1,x_2,...x_n)$, où $x_1,x_2,...x_n$ sont des noms distincts et les seuls noms libres dans P. Le processus $A(y_1,y_2,...y_n)$ se comporte comme P dans lequel chaque x_i a été substitué par y_i . les x_i peuvent être considérés comme des paramètres formels de A et les y_i comme des paramètres d'appel dans A.
11. **Une Réplication (!P) = P| !P** : L'opérateur « ! » est appelé *réplicateur*, définie par !P = P | !P . Le processus P peut être appliqué une infinité de fois.

Définition 2.1 (Forme normale)

On dit qu'un processus P est sous une forme normale de tête s'il est égal à une somme de préfixes,

P = $\sum_i \alpha_i.P_i$ où α_i est un préfixe.

On note par *n(P)* l'ensemble des noms apparaissant dans le processus P .

Les noms *liés* dans un processus P notés *bn(P)* sont les noms *liés* par un préfixe d'*entrée* ou par l'opérateur de *Restriction* . Les noms liés traduisent uniquement les calculs internes.

On écrit P ≡ Q pour exprimer que P et Q sont *alpha-équivalents* (i.e, ils diffèrent seulement par les noms *liés*).

Les noms *libres* dans un processus P notés *fn(P)* sont des noms qui ne sont pas *liés* dans P. Les noms libres d'un processus P représentent l'environnement de ce processus ou sa liaison avec les autres processus de son environnement. Réciproquement, les noms libres d'un processus P représentent ce que l'environnement sait sur ce processus.

Lorsque x, l'objet d'un préfixe de sortie, est lié on dit qu'on a un préfixe de sortie lié et on le note « $\bar{a}(x)$ ».

Les noms libres d'un préfixe α fn(α) et les noms liés bn(α) sont présentés dans la table 2 :

α	type	fn(α)	bn(α)
τ	silencieuse	$\{\}$	$\{\}$
$\bar{x}y$	Préfixe de sortie libre	$\{x, y\}$	$\{\}$
$\bar{x}(y)$	Préfixe de sortie lié	$\{x\}$	$\{y\}$
xy	Préfixe d'entrée libre	$\{x, y\}$	$\{\}$
x(y)	Préfixe d'entrée lié	$\{x\}$	$\{y\}$

Table 2 : noms liés et noms libres

On dit qu'une *action est libre* si tous les noms apparaissant dans cette action sont *libres.*

On note par *P{y/x}* le résultat de la substitution de toutes les occurrences *libres* de x par y dans P, avec un renommage des noms *liés* ceci est nécessaire pour éviter la capture de y (*ie*, la liaison de y) dans P.

Le processus $P\sigma$ est P dans lequel tous les noms libres x sont remplacés par $\sigma(x)$, où σ est une fonction de N vers N.

Considérons le processus ($\bar{x}y.P$ | $(vx)x(z).Q$). Le nom x apparaît dans les deux parties de la composition parallèle, mais ne représente aucune liaison entre $\bar{x}y.P$ et $(vx)x(z).Q$ car x est libre dans $\bar{x}y.P$ mais lié dans $(vx)x(z).Q$. En particulier, ces processus ne peuvent pas communiquer le long du canal x.

Même si le nom x est transmis comme objet à un processus possédant lui-même un nom local x (phénomène appelé scope intrusion), les deux x demeurent distincts comme le montre la transition suivante :

$\bar{x}y.P$ | $(vx)x(z).Q \xrightarrow{\tau}$ P | $(vx')Q\{x'/x\}\{x/z\}$

Le renommage du x local par x' est une conséquence de la définition de la substitution ; les noms liés sont renommés si nécessaire afin d'éviter les captures.

La seule situation où l'environnement de $(vx)P$ peut connaître le nom local x, est lorsque P émet x comme objet dans une communication à l'environnement. La portée de x s'accroit alors en s'étendant de x (mais pas aux autres processus). Ce phénomène est appelé scope extrusion et est illustré par l'exemple suivant.

Supposons que P1 = $\bar{y}x.P1'$ et Q = $y(z).Q'$.

Le canal x est donc transmis à Q le long du canal y par P1. Or, comme x est local à P, sa portée s'étend à Q. Cette nouvelle configuration est illustrée par la partie droite de la figure1 où Q'' représente $Q'\{x/z\}$.

$$(vx)(\text{P1} \mid \text{P2}) \mid \text{Q} \xrightarrow{\tau} (vx)(\ P1' \mid P2 \mid Q'\{x/z\}\ )$$

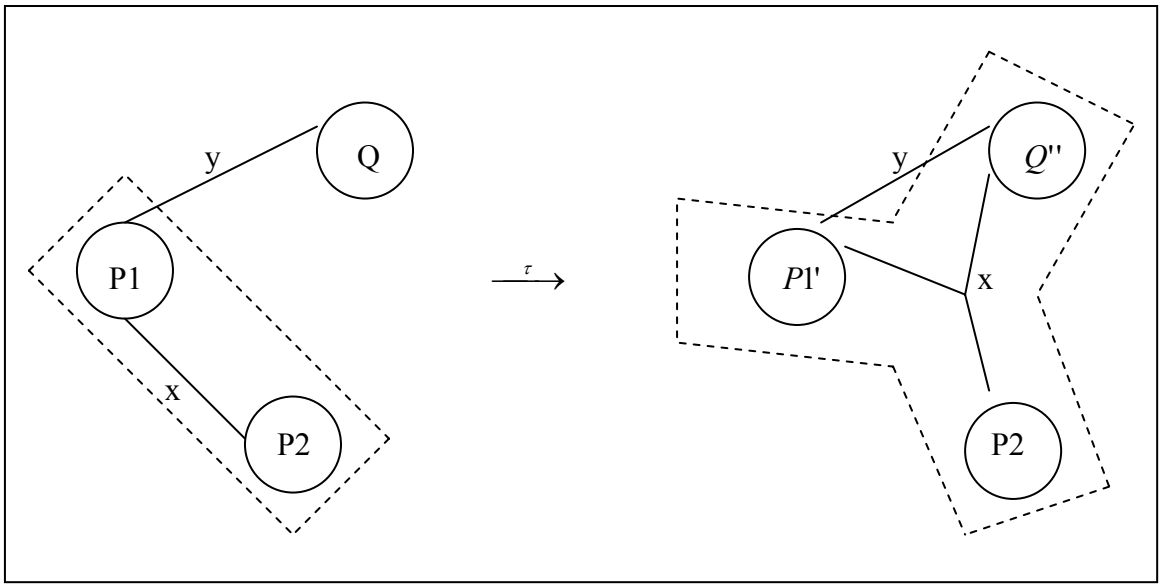

Figure 1 : Scope extrusion

2.1 Congruence structurelle

La congruence structurelle est introduite pour identifier les agents qui intuitivement représentent la même chose, c-à-d qu'ils ont le même comportement.

Donnons quelques définitions :

Définition 2.2 (Monoïde)

Soit E un ensemble, * une opération associative et e un élément neutre pour *. Alors (E,*, e) est un monoïde. Si la loi * est commutative, le monoïde est dit commutatif.

La définition de la congruence structurelle est donnée par la définition suivante :

Définition 2.3 (Congruence structurelle ≡)

La congruence structurelle notée « ≡ » est définie comme étant la plus petite relation sur l'ensemble des processus N vérifiant les propriétés suivantes :

1- Si P et Q sont alpha-équivalents alors P ≡ Q .

2- (N / ≡ , +, 0) et (N / ≡ , |, 0) sont des *monoïdes commutatifs*

3- la loi du développement $A(\tilde{y}) \equiv P\{\tilde{y}/\tilde{x}\}$ si $A(\tilde{x}) \stackrel{def}{=} P$.

4- Les lois de l'extension du scope

$(vx)0 \equiv 0$

$(vx)(P|Q) \equiv P|(vx)Q$ Si x ∉ fn(P)

$(vx)(P+Q) \equiv P+(vx)Q$ Si x ∉ fn(P)

(vx) if u=v then P ≡ if u=v then (vx)P si $x \neq u$ et $x \neq v$

(vx) if u ≠ v then P ≡ if u ≠ v then (vx)P si $x \neq u$ et $x \neq v$

(vx)(vy)P ≡ (vy)(vx)P

3 La sémantique du π-calcul

La sémantique opérationnelle du π-calcul [58] est donnée par la table des transitions suivante:

STRUCT	$\dfrac{P' \equiv P, P \xrightarrow{\alpha} Q, Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$
PREFIX	$\dfrac{}{\alpha.P \xrightarrow{\alpha} P}$
SUM	$\dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$
MATCH	$\dfrac{P \xrightarrow{\alpha} P'}{if x = x then P \xrightarrow{\alpha} P'}$
MISMATCH	$\dfrac{P \xrightarrow{\alpha} P', x \neq y}{if x \neq y then P \xrightarrow{\alpha} P'}$
PAR	$\dfrac{P \xrightarrow{\alpha} P', bn(\alpha) \cap fn(Q) = \{\}}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$
COM	$\dfrac{P \xrightarrow{a(x)} P', Q \xrightarrow{\bar{a}u} Q'}{P \mid Q \xrightarrow{\tau} P'\{u / x\} \mid Q'}$
RES	$\dfrac{P \xrightarrow{\alpha} P', x \notin \alpha}{(vx)P \xrightarrow{\alpha} (vx)P'}$
OPEN	$\dfrac{P \xrightarrow{\bar{a}x} P', a \neq x}{(vx)P \xrightarrow{\bar{a}vx} P'}$

Table 3 : La sémantique opérationnelle

Les différentes valeurs possibles de α sont :

- l'action interne τ .
- l'action de sortie (libre) de type $\bar{a}x$.
- l'action d'entrée de type $a(x)$.
- l'action de sortie liée $\bar{a}(x)$.

II- Extensions du π-calcul

1 Le π-calcul polyadique

Dans le calcul polyadique, on a des objets multiples dans la communication :

Des sorties de type $\bar{a} <y_1 \ldots y_n>.P$ et des entrées de type $a(x_1 \ldots x_n).Q$, où x_i sont deux à deux distincts. Quand n = 0 (il n'y a pas d'objets), donc nous écrirons ā pour ā () et a pour a ().

Le π-calcul polyadique peut être représenté par une séquence d'interactions monadiques à travers une liaison privée.

$\bar{c}(x_1 \ldots x_n).P$ veut dire $(vp)\bar{c}p.\bar{p}x_1 \ldots . \bar{p}x_n.P$

$c(x_1 \ldots x_n).Q$ veut dire $c(p).p(x_1).\ldots . p(x_n).Q$

Une communication dans le π-calcul polyadique est donnée par :

$a(\tilde{\imath}).P \mid \bar{a}<\tilde{o}>.Q \xrightarrow{\tau} P\{\tilde{o} / \tilde{\imath}\} \mid Q$ où $\{\tilde{o} / \tilde{i}\} = \{o_1 / i_1, \ldots\ldots o_i / i_i \ldots\ldots\}$

Dans le cas des processus où la dimension des sorties n'est pas la même que celle des entrées, il y aura une incompatibilité des sortes.

Des systèmes de sorte [49] [58] ont été définis pour résoudre les problèmes de l'incompatibilité de sorte.

2 Recursion

Dans le π-calcul, comme dans la majorité des processus algébriques, la récursivité est le mécanisme permettant la description d'une itération.

Une définition de la récursivité est $A(\tilde{u}) =_{def} P$

Où A apparaît dans P, peut être défini comme une procédure récursive A, avec des paramètres formels ũ, et le processus A(ũ) est invoqué avec les paramètres actuels ũ .

La notion de point fixe a été utilisée pour représenter la recursion et la réplication [58].

3 Le π-calcul asynchrone

Plusieurs définitions sont données pour le π-calcul asynchrone [39] [11] [2] [58] [44] [72] [77] nous donnons la définition citée dans [77].

La syntaxe du π-calcul asynchrone est donnée par la grammaire suivante :

P ::= $\bar{x}z$ | x(y)P | P | P | !P | (vx)P

Où x, y, z sont des noms de canaux et P est un processus.

4 Le π-calcul d'ordre supérieur

Dans le π-calcul d'ordre supérieur les objets transmis dans une communication peuvent être des processus.

La forme d'un préfixe de sortie d'ordre supérieur est ā<P>.Q, ceci signifie « envoyer l'agent P à travers a ensuite continuer comme Q ».

Le préfixe d'entrée d'ordre supérieure est de genre a(X).Q, ce qui signifie « recevoir un agent X ensuite continuer comme Q ».

Dans [49] un système de sorte d'ordre supérieure a été défini pour le π-calcul d'ordre supérieur afin de résoudre les problèmes de sorte.

5 Le π-calcul stochastique

5.1 Introduction

Dans le π-calcul toutes les communications ont des chances égales de se produire (sémantique non déterministe), sa version stochastique [64] introduit la notion de probabilité de réalisation d'une action.

Définition 2.4

Soit N un ensemble dénombrable infini des noms , tel que N = { a,b x,y,z,u,v,…}.avec $\tau \notin$ N. les processus (notés par P,Q,R ,..) sont construits à partir des noms suivant la syntaxe :

P ::= 0 | π.P | (vx)P | if x=y then P | P|P | P+P | P($y_1,\ldots,y_n$)

Où π peut être soit un préfixe d'entrée x(y), un préfixe de sortie $\bar{x}y$, ou bien τ pour l'action silencieuse.

Dans le π-calcul stochastique nous ajoutons aux préfixes π une valeur r qui est le paramètre d'une distribution exponentielle. Ce paramètre est appelé le taux d'activité.

Définition 2.5

Soit r $\in$ R^+. Les Processus de π-calcul stochastique sont construits d'après la définition 2-4 en remplaçant π.P par (π, r) .P .

5.2 Probabilités et taux de transitions

Donnons tout d'abord la définition du taux de sortie d'un processus P.

Définition 2.6 (Taux de sortie)

Le taux de sortie d'un processus P est égal à la somme de tous les taux d'activités permis dans P noté « er(P) » . Tel que : $er(P) = \sum_{P \xrightarrow{(u_j, r_j)} P_j \in Ts(P)} r_j$

Où Ts(P) est l'ensemble de transitions permises dans P.

On admet que :

- Chaque activité a une multiplicité égale à un.
- Le taux de sortie est fini.

La condition de la course provient en considérant la probabilité d'une transition $P \xrightarrow{(u,r)} P'$ qui exprime le rapport entre le taux de l'activité de la transition et celui de sortie de P.
On considère que la synchronisation a son propre taux.

Proposition 2.1
Soit $P \xrightarrow{(u_i, r_i)} P_i \in$ Ts (P). Alors, la probabilité d'une occurrence de l'événement $P \xrightarrow{(u_i, r_i)} P_i$ est r_i /er(P).

Exemple : La probabilité que le processus (a, 3) + (b, 4) termine l'activité a est 3/7.

5.3 Calcul du taux de synchronisation

Pour calculer er (P), il est nécessaire de calculer le taux de synchronisation.

Définition 2.7 (Le taux apparent)

Le taux apparent d'une action a noté « r_a (P) » dans un processus donné P est la somme des taux de toutes les activités ayant l'action a et qui sont permises dans P

r_a (P) = $\sum_{P \xrightarrow{(a, r_j)} P_j \in Ts(P)} r_j$

Exemple

Le taux apparent d'une action de type a dans le processus (a, 3) + (a, 4) est 7.

Le taux apparent de synchronisation entre deux activités (a, r_0) dans P et ($\overline{a}$, r_1) dans Q prend la valeur minimale de ces deux activités , il est définit par :

$$\min \quad (r_a(P), r_{\bar{a}}(Q))$$

Le taux apparent nous permet de calculer des probabilités conditionnelles.

La probabilité d'une transition $P \xrightarrow{(a,r)} P'$, sachant que l'action a se produit, est le rapport entre son taux d'activité et le taux apparent de l'action a. Sous les mêmes suppositions faites dans la proposition 2-1, nous donnons la proposition suivante :

Proposition 2.2

Soit $P \xrightarrow{(a,r_i)} P_i \in$ Ts(P). Sachant qu'une action a se produit, la probabilité de l'événement conditionnelle $P \xrightarrow{(a,r_i)} P_i$ est $r_i / r_a(P)$.

Exemple

La probabilité de la transition $(a,3) \mid (a,2) + (b,1) \xrightarrow{(a,3)} 0 \mid (a,2) + (b,1)$, sachant que a se produit, est 3/5.

Corollaire 2.1

Soit $P \xrightarrow{(a,r_i)} P_i \in$ Ts(P) et soit p sa probabilité d'occurrence.

Alors, r_i = p* r_a (P) .

Ce corollaire nous suggère la méthode de calcul de taux de synchronisation. Celui-ci est obtenu en multipliant la probabilité de l'occurrence par et le taux apparent (voir formule (1)) .

Nous supposons que les processus parallèles décident indépendamment l'un de l'autre, les actions à produire. Si P est défini par (a, r_0) et Q défini par ($\bar{a}$, r_1) alors la probabilité que les deux activités produisent une communication est

$$r_0 / r_a(P) * r_1 / r_{\bar{a}}(Q) .$$

Finalement, le taux de synchronisation est donné par la formule (1) ci-dessous :

$$R(P,a,Q,\bar{a},r_0,r_1) = r_0 / r_a(P) * r_1 / r_{\bar{a}}(Q) * \min \quad (r_a(P), r_{\bar{a}}(Q)) \qquad (1)$$

III- Théorie algébrique

Dans cette partie nous présenterons une axiomatisation algébrique pour le π-calcul, c-à-d nous caractériserons l'ensemble des processus égaux en utilisant la notion de bisimilarité [58] [52] [53] [73].

1 Bisimilarité et congruence

On dit que deux agents sont équivalents si on ne peut pas distinguer les évolutions de leurs comportements. La notion de bisimilarité permet de comparer à tout moment le comportement de deux processus.

1.1 Bisimilarité

Définition 2.8 (Bisimulation forte)

La bisimulation forte est une relation binaire $\mathcal{R}$ symétrique sur des agents et qui vérifie les conditions suivantes :

P $\mathcal{R}$ Q et $P \xrightarrow{\alpha} P'$, où bn($\alpha$) $\notin$ fn (P,Q) implique que :

1. si $\alpha = a(x)$ alors $\exists Q' : Q \xrightarrow{a(x)} Q' \wedge \forall u : P'\{u / x\} \mathcal{R}\ Q'\{u / x\}$.
2. si α n'est pas un préfixe d'entrée, alors $\exists Q' : Q \xrightarrow{\alpha} Q' \wedge P' \mathcal{R}\ Q'$.

On note « $\sim$ » pour exprimer une relation de bisimulation entre deux agents.

Proposition 2.3 :

La relation de bisimulation forte noté « $\sim$ » est préservée par tous les opérateurs (Parallèle, Somme, Restriction, préfixe de sortie, l'action τ) sauf par le préfixe d'entrée.

1.2 Congruence

Le fait que la bisimilarité n'est pas préservée par un préfixe d'entrée, il serait donc intéressant de définir un ensemble des termes congruents le plus grand possible qui soit inclu dans l'ensemble des termes bisimilaires.

Définition 2.9 (Congruence)

Soient deux agents P et Q (fortement) congruent, noté « P $\sim$ Q », si Pσ $\sim$ Q σ pour toutes les substitutions σ .

2 Variantes de bisimilarité

La définition d'une équivalence peut varier suivant plusieurs méthodes. Nous présenterons ici quelques variantes particulières du π-calcul.

2.1 Bisimulation précoce (early)

Définition 2.10 (La sémantique précoce)

L'ensemble des règles de la sémantique précoce est obtenu à partir de la table 03 en remplaçant les règles INPUT et COM par les deux règles suivantes :

$$E-INPUT\ \frac{}{a(x).P \xrightarrow{au} P\{u/x\}} \qquad E-COM\ \frac{P \xrightarrow{au} P', Q \xrightarrow{\bar{a}u} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

On note « $\longrightarrow_E$ » pour indiquer une sémantique précoce.

Une définition de bisimilarité précoce en utilisant la sémantique précoce est donnée par :

Définition 2.11 (Bisimulation précoce)

Une bisimulation précoce (forte) est une relation binaire symétrique $\mathcal{R}$ sur les agents et qui vérifie la condition suivante :

P $\mathcal{R}$ Q et $P \xrightarrow{\alpha}_E P'$ où $\mathrm{bn}(\alpha) \notin$ fn (P,Q) implique que $\exists Q' : Q \xrightarrow{\alpha}_E Q' \wedge P' \mathcal{R} Q'$

Deux agents P et Q sont reliés par une relation de bisimilarité précoce forte notée « P $\sim_E$ Q », s'il existe une bisimulation précoce $\mathcal{R}$, tel que P $\mathcal{R}$ Q.

La bisimilarié citée dans la définition 2-8 est dite une bisimulation tardive et la congruence définie dans la définition 2-9 est dite une congruence tardive.

Définition 2.12 (Bisimulation précoce avec une sémantique tardive)

Une bisimulation précoce avec une sémantique tardive est une relation $\mathcal{R}$ binaire symétrique sur les agents et qui vérifie les conditions suivantes : P $\mathcal{R}$ Q et $P \xrightarrow{\alpha} P'$ où $\mathrm{bn}(\alpha) \notin$ fn (P, Q) implique que :

1. si $\alpha = a(x)$ alors $\forall u \exists Q' : Q \xrightarrow{a(x)} Q' \wedge P'\{u/x\}\ \mathcal{R}\ Q'\{u/x\}$.
2. si α n'est pas un préfixe d'entrée alors $\exists Q' : Q \xrightarrow{\alpha} Q' \wedge P' \mathcal{R} Q'$

Définition 2.13 (Congruence précoce)

P et Q sont deux agents congruents précoce noté « $P \sim_E Q$ » si et seulement si, pour toute substitution σ nous avons $P\sigma \simeq_E Q\sigma$.

2.2 Congruence barbelée (barbed)

Définition 2.14 (Nom observable)

Un nom a est dit observable à l'agent P noté « $P \downarrow a$ » si a est le sujet d'une certaine action à partir de P.

Définition 2.15 (Bisimulation barbelée (barbed))

La bisimulation barbelée est une relation $\mathcal{R}$ binaire symétrique sur les agents vérifiant les conditions suivantes :

$P \mathcal{R} Q$ implique que

1. si $P \xrightarrow{\tau} P'$ alors $\exists Q' : Q \xrightarrow{\tau} Q' \wedge P' \mathcal{R} Q'$
2. si $P \downarrow a$ alors $Q \downarrow a$

Deux agents P et Q sont reliés par une relation de bisimulation barbelée, noté « $P \simeq_B Q$ », s'ils existe une relation $\mathcal{R}$ tel que $\mathcal{R}$ est une relation de bisimulation barbelée.

Définition 2.16 (Contexte d'un processus)

Un contexte est une expression d'un terme donné dans le π-calcul et contient une variable.

Contexte : C :: = [ ] | α.C | P+C | C+P | P|C | C|P | (vx)C | !C

Etant donné un contexte C et un processus P, C (P) est le terme obtenu en substituant la variable qui apparaît dans le contexte C par le processus P.

Définition 2.17 (Congruence barbelée (barbed))

Deux agents P et Q sont reliés par une relation de congruence barbelée, noté « $P \sim_B Q$ », si pour tout contexte C la propriété suivante est vérifiée : C (P) $\simeq_B$ C (Q).

2.3 Bisimulation ouverte

La bisimulation ouverte nous permet d'obtenir directement une congruence à travers la bisimulation.

Nous considérons un sous π-calcul sans les règles de Restriction et de Mismatch.

Définition 2.18 (Bisimulation ouverte)

Une bisimulation ouverte (forte) est une relation binaire symétrique $\mathcal{R}$ sur les agents et qui vérifie pour toutes les substitutions σ, la condition suivante :

P $\mathcal{R}$ Q et $P\sigma \xrightarrow{\alpha} P'$ implique que $\exists Q' : Q\sigma \xrightarrow{\alpha} Q'$ et $P' \mathcal{R} Q'$.

Deux agents P et Q sont reliés par une relation de bisimulation ouverte, notée « P $\sim_O$ Q » s'il existe une bisimulation ouverte $\mathcal{R}$ tel que P$\mathcal{R}$Q.

A partir de la définition 2-18 on peut déduire que la bisimulation ouverte est fermée sous les substitutions dans le sens où P $\sim_O$ Q implique Pσ $\sim_O$ Qσ, donc P $\sim_O$ Q implique P $\sim$ Q.

2.4 Bisimulation faible

Dans la bisimilarité faible les actions τ sont ignorées.

On définit $\Rightarrow$ pour exprimer $\left(\xrightarrow{\tau}\right)^*$, c à d zéro ou plus de transitions τ. Le symbole $\overset{\alpha}{\Rightarrow}$ exprime $\Rightarrow \xrightarrow{\alpha} \Rightarrow$: si $\alpha \neq \tau$ alors on note « $\overset{\hat{\alpha}}{\Rightarrow}$ », sinon on note « $\Rightarrow$ ».

Définition 2.19 (Bisimulation faible)

Une bisimulation faible (tardive) est une relation binaire symétrique $\mathcal{R}$ sur les agents et qui vérifie les conditions suivantes :

P $\mathcal{R}$ Q et $P \xrightarrow{\alpha} P'$ où $\text{bn}(\alpha) \notin$ fn (P, Q) implique que

1. si $\alpha = a(x)$ alors $\exists Q'' : Q \Rightarrow \xrightarrow{a(x)} Q'' \wedge \forall u \exists Q' : Q''\{u/x\} \Rightarrow Q' \wedge P'\{u/x\} \mathcal{R} Q'$
2. si α n'est pas un préfixe d'entrée alors $\exists Q' : Q \overset{\hat{\alpha}}{\Rightarrow} Q' \wedge P' \mathcal{R} Q'$

Deux agents P et Q sont reliés par une relation de bisimulation faible (tardive), notée « P $\dot{\approx}$ Q » s'ils sont reliés par une bisimulation faible. La relation $\dot{\approx}$ n'est pas une congruence. La congruence faible tardive est définie comme suit :

Définition 2.20

P et Q sont reliés par une relation de congruence faible (tardive) si, pour toutes les substitutions σ , $P\sigma \xrightarrow{\alpha} P'$ où $\text{bn}(\alpha) \notin$ fn (P, Q) implique que :

1. si $\alpha = a(x)$ alors $\exists Q'' : Q\sigma \Rightarrow \xrightarrow{a(x)} Q'' \wedge \forall u \exists Q' : Q''\{u/x\} \Rightarrow Q'$
$\wedge\ P'\{u/x\} \dot{\approx} Q'$
2. Si α n'est pas un préfixe entrée alors $\exists Q' : Q\sigma \Rightarrow^{\alpha} Q' \ \wedge P' \approx Q'$

3 Axiomatisation

On considère un système de sous π-calcul fini, c-à-d sans les règles de Réplications et d'Identificateur, noté π_f -calcul.

3.1 Axiomes de la bisimilarité

Les axiomes pour la bisimilarité tardive forte sont donnés par le système algébrique exprimé dans la table 04.

STR	si $P \equiv Q$ alors $P=Q$
CONGR1	si $P=Q$ alors $\bar{a}u.P = \bar{a}u.Q$
	$\tau.P = \tau.Q$
	$P + R = Q + R$
	$(vx)\,P = (vx)\,Q$
CONGR2	si $P\{y/x\} = Q\{y/x\}$ pour tout $y \in fn(P,Q,x)$ alors $a(x).P = a(x).Q$
S	$P + P = P$
M1	if $x = x$ then $P = P$
M2	if $x = y$ then $P = 0$ si $x \neq y$
MM1	if $x \neq x$ then $P = 0$
MM2	if $x \neq y$ then $P = P$ si $x \neq y$
R1	$(vx)\,\alpha.P = \alpha.(vx)\,P$ si $x \notin \alpha$
R2	$(vx)\,\alpha.P = 0$ si x est le sujet de α
R3	$(vx)\,(P + Q) = (vx)\,P + (vx)\,Q$

EXP soient $P = \sum_i \alpha_i.P_i$ et $Q = \sum_j \beta_j.Q_j$

Avec

- $bn(\alpha_i) \cap fn(Q) = \emptyset$ et $bn(\beta_j) \cap fn(P) = \emptyset$, $\forall$ i , j .
- α_i et β_j ne sont pas des préfixes liés de sortie.

Alors

$$P \mid Q = \sum_i \alpha_i.(P_i \mid Q) + \sum_j \beta_j.(P \mid Q_j) + \sum_{\alpha_i comp \beta_j} .\tau.R_{ij}$$

Où $\alpha_i comp \beta_j$ et R_{ij} sont définis comme suit :

1- $$\alpha_i comp \beta_j = \begin{cases} \alpha_i = a(x) \\ \beta_j = \bar{a}u \end{cases} \text{ et } R_{ij} = P_i\{u/x\} \mid Q_j$$

2- $$\alpha_i comp \beta_j = \begin{cases} \alpha_i = \bar{a}u \\ \beta_j = a(x) \end{cases} \text{ et } R_{ij} = P_i \mid Q_j\{u/x\}$$

Table 4 : Les axiomes pour la bisimilarité tardive forte

3.2 Axiomes de la congruence

Définition 2.21 (Opérateur de Match généralisé)

L'opérateur de Match généralisé est noté par « if M then P » où M est une conjonction de conditions de type x = y et $x \neq y$ et P est un processus.

Tel que: if $m_1 \wedge ... m_k$ then P = if m_1 then (… (if m_k then P)…)

où chaque m_i est de type x = y ou $x \neq y$.

Définition 2.22 (Implication logique du match généralisé)

Soient M et N deux conjonctions de conditions de type x = y et $x \neq y$.

On dit que M logiquement implique N si toutes les substitutions qui rendent les conditions dans M vraies rendent aussi les conditions dans N vraies, et qu'elles sont logiquement équivalentes (noté « $M \Leftrightarrow N$ ») si chacune implique l'autre.

Définition 2.23 (Conjonction de conditions completes)

Une conjonction de conditions M est complète sur un ensemble de noms V si M implique la formule ((x = y) $\vee$ ($x \neq y$)) pour tous les noms x, y dans V.

Définition 2.24 (Forme normale de tête sur un ensemble fini des noms)

Soit un processus P, on dit que P est sous une forme normale de tête sur un ensemble fini des noms V (V - hnf) si

$$P = \sum_i if M_i then \alpha_i P_i$$

Où pour tout i, $bn(\alpha_i) \notin V$, et chaque M_i est complet sur V.

Les axiomes de la congruence forte sont donnés dans la table 5 :

STR si P ≡ Q alors P=Q

CONGR « = » est préservé par tous les opérateurs

S P + P = P

MM1 if x ≠ x then P = 0

GM1 if M then P = if N then P si M ⇔ N

GM2 if x = y then P else P = P

GM3 if M then ($P_1 + P_2$) = if M then P_1 + if M then P_2

GM4 if M then $\alpha.P$ = if M then (α . if M then P) si bn$(\alpha) \notin M$

GM5 if x = y then $\alpha.P$ = if x = y then (α {x/y}.P)

GM6 (vx) if x = y then P = 0 si x ≠ y

R1 (vx) α . P = α . (vx) P si x ∉ α

R2 (vx) α . P = 0 si x est le sujet de α

R3 (vx) (P + Q) = (vx) P + (vx) Q

EXP2 Soient $P = \sum_i$ if M_i then $\alpha_i.P_i$ et $Q = \sum_j$ if N_j then $\beta_j.Q_j$

Avec

- bn(α_i) ∩ fn(Q) = ø et bn(β_j) ∩ fn(P) = ø , ∀ i , j
- α_i et β_j ne sont pas des préfixes liés de sortie

Alors

$$P \mid Q = \sum_i \text{ if } M_i \text{ then } \alpha_i . (P_i \mid Q) + \sum_j \text{ if } N_j \text{ then } \beta_j (P \mid Q_j)$$

$$+ \sum_{\alpha_i opp \beta_j} \text{ if } M_i \wedge N_i \wedge a_i = b_j \text{ then } \tau.R_{ij}$$

Où $\alpha_i comp \beta_j$ et R_{ij} sont définis comme suit :

$$1\text{-}\ \alpha_i comp \beta_j = \begin{cases} \alpha_i = a_i(x) \\ \beta_j = \overline{b}_j u \end{cases} \text{ et } R_{ij} = P_i\{u/x\} \mid Q_j$$

$$2\text{-}\ \alpha_i comp \beta_j = \begin{cases} \alpha_i = \overline{a}_i u \\ \beta_j = \overline{b}_j(x) \end{cases} \text{ et } R_{ij} = P_i \mid Q_j\{u/x\}$$

Table 5 : Les axiomes de la congruence tardive forte

4 Conclusion

Nous avons présenté le π-calcul une des algèbres des processus qui possède la mobilité comme primitive de base.

Nous avons défini sa syntaxe, sa sémantique, ses extensions et enfin sa théorie algébrique.

Afin de répondre à la question « la spécification du π-calcul est-elle convenable pour un système donné ? », nous présenterons dans le chapitre suivant un model checking pour ce calcul. Ce modèle est défini pour vérifier si un système spécifié en π-calcul, satisfait ou non certaines propriétés.

Chapitre 3

Un model checking pour le π-calcul

1 Introduction

Dans ce chapitre nous présenterons les pricipales parties d'un model checking pour le π-calcul. Nous commençons par la définition d'un automate ordinaire pour le π-calcul en passant par un nouveau type d'automate, appelé HD-automate (History Dependant automata) décrit dans [24] [63]. Nous présenterons ensuite, les logiques définies pour les processus mobiles dont la π-logique qui est une logique temporelle. Le mécanisme de traduction d'une formule de π-logique en une formule de la logique ACTL sera aussi abordé. Nous terminons le chapitre par la description d'un environnement de vérification des processus mobiles : le HAL.

2 Le principe du model checking

Le model-checking est une technique automatique de vérification formelle d'un système fini. Il est très utilisé pour étudier les systèmes réactifs, les systèmes critiques, les systèmes temps réel et les systèmes embarqués complexes.

Le principe de cette technique est le suivant : étant donné un automate à états finis ou bien un système de transition étiqueté et une propriété désirée, formalisée dans une logique temporelle, l'algorithme explore l'ensemble des états de cet automate afin de vérifier que la propriété désirée est bien satisfaite (figure : 2).

Si ce n'est pas le cas, la séquence de transitions d'état menant à la violation du système est générée en guise de contre-exemple, ce qui montre que le système est incorrect.

L'algorithme permettant de vérifier si un automate ordinaire satisfait la propriété s'appelle un model-checker.

Ces propriétés sont de plusieurs types : les logiques temporelles peuvent exprimer les notions d'accessibilité d'un état, de sûreté (le comportement spécifié ne se produira jamais), de vivacité (le comportement spécifié aura lieu), d'équité (le comportement spécifié pourra se produire une infinité de fois) et des propriétés temporelles.

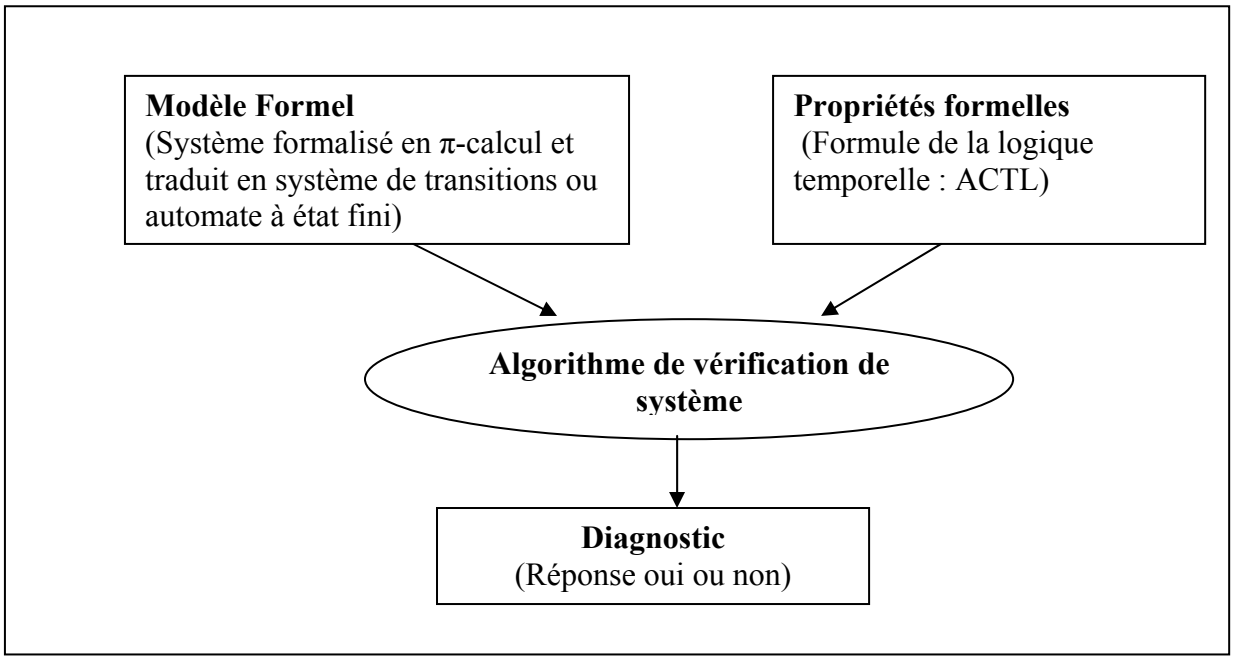

Figure 2 : Le model-checking

3 Les automates ordinaires pour le π-calcul

Un système simple en π-calcul peut générer un automate à état infini puisque le préfixe d'entrée de la sémantique opérationnelle du π-calcul nécessite un nombre infini d'états et la création d'un nouveau nom engendre un ensemble infini de transitions: un pour chaque choix du nom nouveau. Pour résoudre ce problème, la génération d'un automate ordinaire pour un agent du π-calcul se fait en deux étapes. La première étape consiste à associer le HD-Automate [23] [24] [25] [54] [56] [57] [63] [13] correspondant à la spécification du processus en π-calcul. La deuxième étape consiste à associer l'automate ordinaire correspondant du HD-automate.

Nous commençons par une définition de l'automate ordinaire.

Définition 3.1 (Automates ordinaires)

Un automate ordinaire A est un quadruplets A= (Q, q^0, L, R) où :

- Q est un ensemble fini d'états ;
- q^0 est l'état initial ;
- L est un ensemble fini d'étiquettes d'actions;
- $R \subseteq Q * Act * Q$ est une relation de transition. Si $(q, \lambda, q') \in$ R on écrira $q \xrightarrow{\lambda} q'$.

Définition 3.2 (Bisimulation sur les automates ordinaires)

Soient A_1 et A_2 deux automates sur le même ensemble L d'étiquettes. Une relation binaire $R \subseteq Q_1 * Q_2$ est une simulation pour A_1 et A_2 si chaque fois que :

Pour tout $t_1 : q_1 \xrightarrow{\lambda} q_1'$ de A_1, il existe $t_2 : q_2 \xrightarrow{\lambda} q_2'$ de A_2 tel que $q_1' R q_2'$.

La relation R est une bisimulation si les deux relations R et R^{-1} sont des simulations.

Deux automates A_1 et A_2 sont bisimilaires, noté $A_1 \sim A_2$, si leurs états initiaux q_1^0 , q_2^0 sont bisimilaires, c à d, q_1^0 R q_2^0 pour une certaine bisimulation R.

3.1 Transformation d'un agent du π-calcul à un HD-automate

Dans le HD-automate des noms locaux paraissent explicitement dans les états, les transitions et les étiquettes. L'usage des noms locaux permet de modéliser l'exécution d'un préfixe d'entrée par un nombre fini de transitions. Ceci est fait en considèrant comme valeur de l'objet du préfixe d'entrée tous les noms libres de l'état source plus un seul nouveau nom appelé nom frais. Donc il sera inutile d'avoir plus de transitions qui diffèrent seulement dans le choix du nom frais. Un seul état du HD-automate peut être utilisé pour représenter tous les états du système qui différent juste par un renommage bijectif. Chaque transition est requise pour représenter explicitement les correspondances entre les noms de source, les noms de cible et les noms de l'étiquette.

Définition 3.3 (HD-Automate)

Un HD- automate est une structure A=(Q, q^0,L,w, $q \xrightarrow[\sigma]{\lambda} q'$) , où :

- Q est un ensemble fini d'états;
- q^0 est l'état initial;
- L est un ensemble d'étiquettes de l'action;
- w est une fonction qui associe des ensembles finis de noms locaux aux états:

  $w : Q \longrightarrow P_f(N)$;

- $q \xrightarrow[\sigma]{\lambda} q'$ est la relation de transition où σ : w (q ') $\longrightarrow$ w(q) U {*} est une fonction (Injective), et * est un nom distingué.

La fonction σ fait correspondre les noms de l'état de la cible aux noms de l'état de la source de la transition. Le symbole distingué * est utilisé pour manipuler la création d'un nom nouveau: le nom créé pendant la transition est associé à *. Les noms qui paraissent dans la source de la transition mais pas dans la cible seront éliminés.

Un HD-automate à états finis peut être construit pour une classe d'agents finis.

La figure 3 illustre le HD-automate correspondant de l'agent : P(in, out) := in(x). $\overline{out}$ x

Les bisimulations ne peuvent pas être définies simplement comme des relations symétriques sur des états mais elles doivent être traitées aussi avec les correspondances des

noms. Un HD-bisimulation[3] est un ensemble de triplets de la forme $\langle q_1,\delta,q_2\rangle$ où q_1 et q_2 sont des états de HD-automate et δ une bijection partielle entre les noms des états.

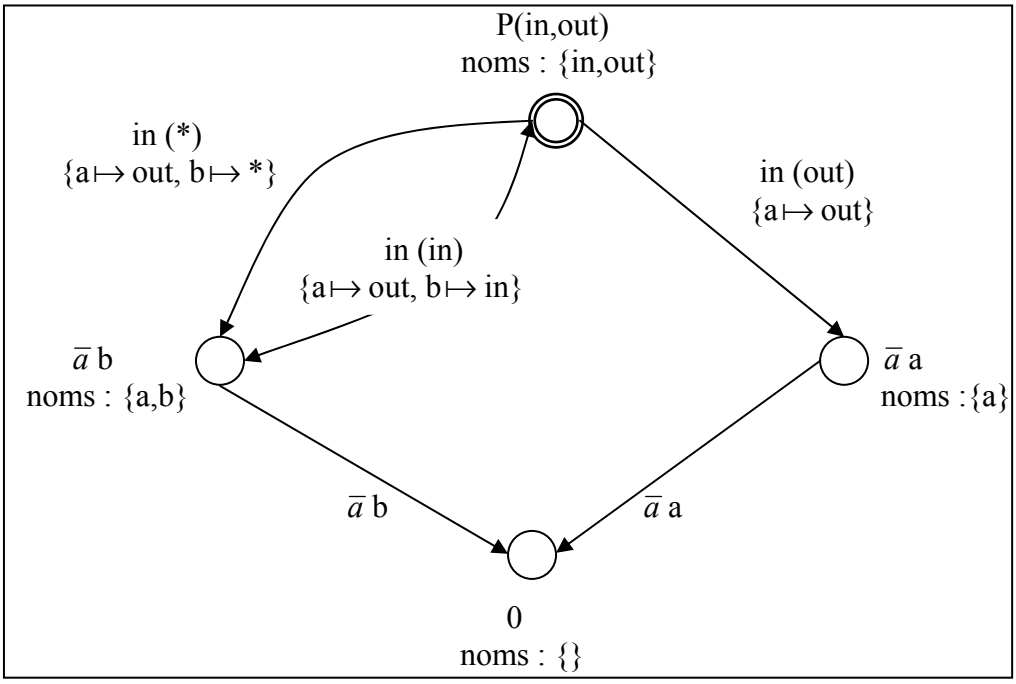

Figure 3 : Le HD-automate qui correspond à l'agent P(in, out) := in(x).$\overline{out}$ x

Définition 3.4 (HD-bisimulation)

Soient A_1 , et A_2 deux HD-automates sur le même ensemble L d'étiquettes. Une HD-simulation pour A_1 , et A_2 est un ensemble de triplets

R $\subseteq$ { $\langle q_1,\delta,q_2\rangle | q_1\in Q_1,Q_2\in q_2,\delta$ est une bijection partielle de $w_1(q_1)$ et $w_2(q_2)$ }tel que, pour tout $\langle q_1,\delta,q_2\rangle\in R$ nous avons:

- pour chaque $t_1 : q_1 \xrightarrow{\lambda_1} q_1'$ il y a quelque $t_2 : q_2 \xrightarrow{\lambda_2} q_2'$, et:

  - $\lambda_2 = \delta^*(\lambda_1)$ où δ^* est une bijection partielle entre $w_1(q_1)\cup\{*\}$ et $w_2(q_2)\cup\{*\}$ tel que
    $\delta^*(x)=\delta(x)$ si $\delta^*(x)\in w_2(q_2)$;
  - $\langle q_1',\delta',q_2'\rangle \in R$, où $\delta'=\sigma_2^{-1}o\delta^* o\sigma_1$.

Une Relation R est une HD-bisimulation si tout R et $R^{-1}=\{\langle q_2,\delta^{-1},q_1\rangle | \langle q_1,\delta,q_2\rangle \in R\}$ sont des HD-simulations.

[3] On note HD-bisimulation pour désigner la bisimilarité pour le HD-automate.

Deux HD-automates A_1 , et A_2 sont HD-bisimilaires et on note $A_1 \simeq A_2$, si leurs états initiaux sont bisimilaires d'après la bijection partielle qui est l'identité sur $w_1(q_1^0) \cap w_2(q_2^0)$.

La fonction δ* nous permet d'étendre la fonction δ en associant au symbole spécial * dans t_1 le symbole * dans t_2, ou bien en associant au symbole * un nom de $w_1(q_1)$ qui ne soit pas couvert par δ. Ce deuxième cas est nécessaire puisque q_1 et q_2 peuvent avoir des ensembles des noms libres différents d'où des ensembles de transition d'entrée différents.

3.2 Transformation d'un HD-automate à un automate ordinaire

Il est possible d'extraire du HD-Automate d'un agent du π-calcul sa sémantique opérationnelle précoce. Cela est fait d'une manière simple : Quand on a un nom frais dans une transition du HD-Automate, une instanciation globale doit être choisie pour ce nom. Nous obtenons une transition pour tous les choix possibles du nom frais. Donc la procédure mène à un automate à état infini. Pour obtenir un automate à état fini il suffit de prendre comme nom frais le premier nom qui n'a pas déjà été utilisé. De cette manière, un automate à état fini est obtenu pour chaque HD-automate fini.

L'automate ordinaire obtenu à partir du HD-automate de la figure 3 est représenté dans la figure 4.

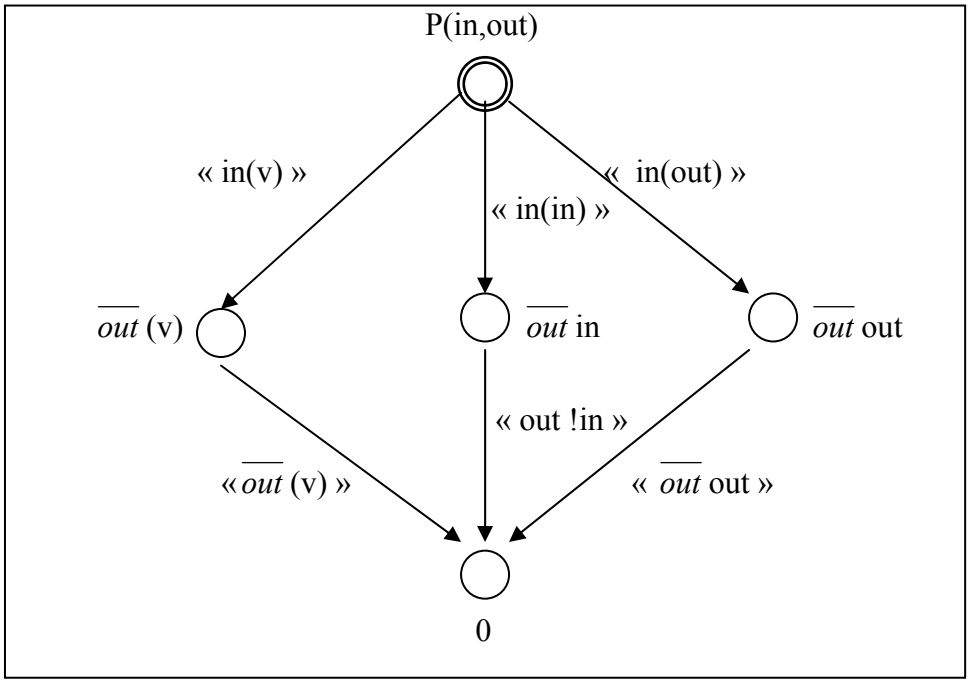

Figure 4 : L'automate ordinaire qui correspond au HD-automate de la Figure 3

La procédure transforme un sous ensemble d'agent de π-calcul à un automate à état fini.

Des agents du π-calcul équivalents (c-à-d bisimilaire dans le sens bisimilarité forte ou faible) sont associés à des automates ordinaires équivalents (c-à-d bisimilaire dans le sens forte ou faible). Ceci est vrai seulement si nous pouvons garantir que deux agents bisimilaires ont les mêmes ensembles des noms libres. Pour ce but, le HD-automate doit être transformé en un HD-automate non redondant dans une phase de prétraitement. La transformation d'un HD-automate à un automate non redondant jette tous les noms qui paraissent dans les états du HD-automate mais qui ne jouent aucun rôle actif dans les calculs de cet état.

Dans [55] un algorithme simple et effectif est présenté pour transformer un HD-automate à un HD-automate non redondant pour une classe d'agent du π-calcul sans l'opérateur de Matching. Le HAL exploite une extension de cet algorithme qui manipule une forme limitée de l'opérateur de Matching.

4 La logique pour les processus

Plusieurs logiques ont été définies pour les processus afin de décrire leurs propriétés. On présentera ici la logique modale et la logique temporelle.

4.1 La logique modale

Les formules de la logique modale [52] [78] [7] sont interprétées en utilisant des connecteurs booléèns et deux opérateurs appelés opérateurs modaux :

$[\alpha]$ (« boite α ») et $\langle\alpha\rangle$ (« diamant α »).

Pour les processus concurrents Hennessy-Milner proposent une logique simple notée « HML » pour exprimer les propriétés des systèmes concurrents.

4.1.1 La logique de Hennessy-Milner

Définition 3.5 (HML)

Soient A l'ensemble de toutes les actions possibles, α une action telle que $\alpha \in A$ et Φ dénote les formules dans HML.

$\Phi ::= \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [\alpha]\Phi \mid \langle\alpha\rangle\Phi \mid \text{true} \mid \text{false}$

Le processus P satisfait la formule Φ est dénoté par « $P \models \Phi$ » et sa sémantique :

- $P \models \text{True}$
- $P \not\models \text{False}$

- $P \models \Phi_1 \wedge \Phi_2 \Leftrightarrow P \models \Phi_1$ et $P \models \Phi_2$
- $P \models \Phi_1 \vee \Phi_2 \Leftrightarrow P \models \Phi_1$ ou $P \models \Phi_2$
- $P \models [\alpha]\Phi \Leftrightarrow \{ \forall \alpha.P \xrightarrow{\alpha} P' \text{ et } P' \models \Phi \}$
- $P \models \langle\alpha\rangle\Phi \Leftrightarrow \{ \exists \alpha.P \xrightarrow{\alpha} P' \text{ et } P' \models \Phi \}$

Exemple :

- $P \models \langle\alpha\rangle$ True (le processus P peut exécuter α).
- $P \models [\alpha]$ False (P est interbloqué si α est exécuté).
- $\langle\alpha\rangle$ True Exprime la capacité d'exécuter α.
- $[\alpha]$ False Exprime l'incapacité d'exécuter α.

$[-]$ Est une méthode pour dénoter n'importe quelle action observable dans l'ensemble des actions, (à l'exception de τ). Dans le cas où les actions τ sont considérées, les opérateurs modaux peuvent être écrits comme :

$$P \models [[\alpha]]\Phi \Leftrightarrow \{ \forall \alpha.P \Rightarrow^{\alpha} P' \text{ et } P' \models \Phi \}$$

$$P \models \langle\langle\alpha\rangle\rangle\Phi \Leftrightarrow \{ \exists \alpha.P \Rightarrow^{\alpha} P' \text{ et } P' \models \Phi \}$$

Où $\Rightarrow^{\alpha}$ veut dire $\left(\xrightarrow{\tau}\right)^* \xrightarrow{\alpha} \left(\xrightarrow{\tau}\right)^*$.

Propriété

Soient α une action et ϕ une formule logique écrite dans la HML alors :

$[\alpha]\phi = \neg\langle\alpha\rangle\neg\phi$ et $\langle\alpha\rangle\phi = \neg[\alpha]\neg\phi$

4.1.2 La modalité précoce et tardive

Dans la logique modale on définit deux modalités : précoce et tardive. Ces deux modalités diffèrent dans la définition de l'opérateur de diamant d'un préfixe d'entrée noté « $\langle x(y)\rangle^L$ » qui permet d'exprimer la modalité tardive et « $\langle x(y)\rangle^E$ » la modalité précoce.

En terme de la logique modale, la bisimilarité tardive est caractérisée par la modalité tardive, et la bisimilarité précoce par la modalité précoce.

Définition 3.6

Soient P un processus et φ une formule logique écrite dans la HML alors :

$P \models \langle x(y) \rangle^{L} \varphi \Leftrightarrow \{ \exists P'. \forall z P \xrightarrow{x(y)} P' \text{ et } P'\{z/y\} \models \varphi \{z/y\} \}$

$P \models \langle x(y) \rangle^{E} \varphi \Leftrightarrow \{ \forall z. \exists P', P \xrightarrow{x(y)} P' \text{ et } P'\{z/y\} \models \varphi \{z/y\} \}$

4.1.3 Limitation

Bien que la logique modale soit puissante, elle ne peut pas exprimer les propriétés de durabilité. Seulement les propriétés « immédiates » sont possibles à formuler dans la logique modale. les déclarations comme « la propriété P est toujours possible » ou bien « l'action a se passera finalement » sont en dehors de sa portée, ceci est dû au fait que les formules de la logique modale sont toujours finies, alors qu'un processus peut être infini. Ce qui implique la nécessité d'un outil plus efficace qui est la logique temporelle.

4.2 La logique temporale

La logique temporelle [78] [7] peut être vue comme la logique modale avec récursion.

4.2.1 La π-logique

La π-logique [10] [34] [35] [24] est une logique temporelle. Elle est basée sur la logique modale. Elle permet d'exprimer les propriétés de vivacité et de sûreté en introduisant l'opérateur temporel EF.

La syntaxe de π-logique est donnée par (ϕ est une formule de π-logique) :

$\phi ::= \text{true} \mid \neg\phi \mid \phi \wedge \phi' \mid \text{EX}\{u\}\phi \mid \text{EF}\,\phi$

L'interprétation d'une formule de la π-logique est comme suit :

- P $\models$ true valable toujours.
- P $\models \neg\phi$ si et seulement si, $\neg$ (P $\models \phi$).
- P $\models \phi \wedge \phi'$ si et seulement si, P $\models \phi$ et P $\models \phi'$.
- P $\models$ EX $\{u\}\phi$ si et seulement si, il y a $P \xrightarrow{u} P'$ et $P' \models \phi$.
- P $\models \langle u \rangle \phi$ si et seulement si, il existe $P_0, \ldots P_n$ $n \geq 1$ tel que $P = P_0 \xrightarrow{tau} P_1 \ldots \xrightarrow{tau} P_{n-1} \xrightarrow{u} P_n$ et $P_n \models \phi$.

- P ⊨EF ϕ si et seulement si, il existe $P_0,....P_n$ $u_1,....u_n$ avec n≥0 tel que

$$P = P_0 \xrightarrow{u_1} P_1 ..... \xrightarrow{u_n} P_n \text{ et } P_n \models \phi .$$

Les opérateurs dérivés suivants peuvent être définis:

- $\phi \mid \phi'$ est représenté par $\neg(\neg\phi \wedge \neg\phi')$
- AX $\{u\}\phi$ est représenté par $\neg EX\{u\}\neg\phi$.
- $[u]\phi$ est représenté par $\neg\langle u\rangle\neg\phi$.
- AG ϕ est représenté par $\neg EF\neg\phi$.

Le développement d'un model checker pour la π-logique exploite et réutilise le model checker implémenté pour la logique ACTL

4.2.2 ACTL (Action Computation Tree Logic)

La logique ACTL [18] [20] [16] [47] [17] [21] [15] [27] [28] peut décrire les propriétés de la sûreté et de la vivacité. Elle exploite l'idée de "évolution dans le temps par des actions" et les formules logiques prennent leur signification sur les systèmes de transition étiquetés (LTS) ou les automates ordinaires.

Définition 3.7 (Formules de l'action)

Etant donné un ensemble Act d'actions observables, le langage AF(Act) de la formule de l'action x sur Act est défini par la syntaxe suivante:

x ::= true | b | ¬ x | x ∧ x

Où b ∈ Act.

Conventions

L'opérateur booléèn faux est abrégé à ¬ true et la disjonction $x \vee x'$ sera abrégée à $\neg(\neg x \wedge \neg x')$.

Définition 3.8 (Syntaxe ACTL)

La syntaxe de la formule ACTL est donnée par la grammaire ci-dessous:

ϕ ::= true | $\phi \wedge \phi'$ | $\neg\phi$ | $E\ \pi$ | $A\ \pi$

π ::= $X\{x\}\phi$ | $X\{tau\}\phi$ | $[\phi\{x\}U\phi]$ | $[\phi\{x\}U\{x'\}\phi]$

Où :

- ϕ et π sont respectivement une formule d'état et une formule de chemin.
- x , x' sont des formules de l'action.
- E et A sont respectivement des quantificateurs de chemin.
- X et U sont les opérateurs Next et Until respectivement.

De même manière, faux est abrégé à $\neg true$ et $\phi \vee \phi'$ est abrégée à $\neg(\neg\phi \wedge \neg\phi')$.

De plus, nous définissons les opérateurs dérivés suivants :

- $EF\phi$ est représenté par $E[true\{true\}U\phi]$
- $AG\phi$ est représenté par $\neg EF\neg\phi$
- $\langle a\rangle\phi$ est représenté par $E[true\{false\}U\{a\}\phi]$
- $\langle tau\rangle\phi$ est représenté par $E[true\{false\}U\phi]$

Pour présenter la sémantique ACTL, nous avons besoin d'introduire la notion de chemins sur un automate ordinaire.

Définition 3.9 (Chemins)

Soit $A=(Q,q^0,Act\cup\{tau\},R)$ un automate ordinaire.

- σ est un chemin de $r_0\in Q$ si soit $\sigma=r_0$ (le chemin vide de r_0) ou bien σ est une séquence (peut-être infini) $(r_0,\alpha_1,r_1)(r_1,\alpha_2,r_2)\ldots$ tel que $(r_i,\alpha_{i+1},r_{i+1}) \in$ R.
- La concaténation de chemins est dénotée par juxtaposition. La concaténation $\sigma_1\sigma_2$ est une opération partielle: elle est définie seulement dans le cas où σ_1 est fini et son dernier état coïncide avec l'état initial de σ_2. La concaténation de chemins est associative et a des identités. C-a-d $\sigma_1(\sigma_2\sigma_3)=(\sigma_1\sigma_2)\sigma_3$ et si r_0 est le premier état de σ et r_n est son dernier état, alors nous avons $r_0\sigma=\sigma r_n=\sigma$.
- Un chemin σ est appelé maximal s'il est infini ou s'il est fini avec le dernier état n'ayant pas d'états successeur. L'ensemble des chemins maximaux de r_0 sera dénoté par $\prod(r_0)$.
- Si σ est infini, alors $|\sigma|=w$.

  Si $\sigma=r_0$, alors $|\sigma|=0$.

  Si $\sigma=(r_0,\alpha_1,r_1)(r_1,\alpha_2,r_2)\ldots(r_i,\alpha_{i+1},r_{i+1})$ $n\geq 0$, alors $|\sigma|=n+1$.

  De plus, nous dénoterons le i ème état dans la séquence c.-à-d r_i , par $\sigma(i)$.

Définition 3.10 (sémantique de la formule de l'action)

La relation de satisfaction $\models$ pour la formule de l'action est définie comme suit:

- a $\models$ true toujours
- a $\models$ b si et seulement si a=b
- a $\models \neg x$ si et seulement si non a $\models$ x
- a $\models x \wedge x'$ si et seulement si a $\models$x et a $\models x'$

Définition 3.11 (sémantique ACTL)

Soit $A=(Q,q^0,Act\cup\{tau\},R)$ un automate ordinaire. Soient $s\in Q$ et σ un chemin. La relation de satisfaction pour une formule ACTL est définie comme suit:

- s $\models$ true toujours
- s $\models(\phi\wedge\phi')$ si et seulement si s $\models\phi$ et s $\models\phi'$
- s $\models\neg\phi$ si et seulement si $\neg$ s $\models\phi$
- s $\models E\ \pi$ si et seulement si, il existe $\sigma\in\prod(s)$, tel que $\sigma\models\pi$
- s $\models A\ \pi$ si et seulement si pour tout $\sigma\in\prod(s)\ \sigma\models\pi$
- $\sigma\models X\{x\}\phi$ si et seulement si, $\sigma=(\sigma(0),\alpha_1,\sigma(1))\sigma'$, et $\alpha_1\models$ x et $\sigma(1)\models\phi$
- $\sigma\models X\{tau\}\phi$ si et seulement si, $\sigma=(\sigma(0),tau,\sigma(1))\sigma'$ et $\sigma(1)\models\phi$
- $\sigma\models[\phi\{x\}U\phi']$ si et seulement si, il existe $i\geq0$ tel que $\sigma(i)\models\phi'$ et pour tout $0\leq j<i:\sigma=\sigma'(\sigma(j),\alpha_{j+1},\sigma(j+1))\sigma''$ implique $\sigma(j)\models\phi$ et $\alpha_{j+1}=tau$ ou bien $\alpha_{j+1}\models$x
- $\sigma\models[\phi\{x\}U\{x'\}\phi']$ si est seulement si, il existe $i\geq1$ tel que $\sigma=\sigma'(\sigma(i-1),\alpha_i,\sigma(i))\sigma''$ et $\sigma(i)\models\phi'$ et $\sigma(i-1)\models\phi$ et $\alpha_i=x'$ et pour tout $0<j<i:\sigma=\sigma'_j(\sigma(j-1),\alpha_j,\sigma(j))\sigma_j''$ implique $\sigma(j-1)\models\phi$ et $\alpha_j=tau$ ou bien $\alpha_j\models$x

La logique ACTL vérifie la propriété suivante : deux automates ordinaires A_1 et A_2 sont fortement bisimilaires si et seulement si :

$F_1=F_2$ où $F_i=\{\psi\in ACTL:A_i satisfie\ \psi\}$ i= 1,2. Où ACTL est l'ensemble des formules décrites par la syntaxe ACTL.

4.2.3 La traduction d'une formule de π-logique en une formule de la logique ACTL

La traduction d'une formule de π-logique en une formule de la logique ACTL a été définie en respectant la propriété suivante :

une formule de π-logique est satisfaite par un agent P de π-calcul si et seulement si l'automate ordinaire à état fini associé à P satisfait la formule ACTL correspondante par la fonction de traduction .

La traduction d'une formule n'est pas unique, mais dépend de S l'ensemble des noms frais de l'automate ordinaire associé à l'agent correspondant.

Définition 3.12

Soit $\theta = \{\alpha/y\}$. On définit $\mu\theta$ l'action u' obtenue en remplaçant les occurrences du nom y dans μ par le nom α. On définit aussi $\text{true}\,\theta = \text{true}$, $(\phi_1 \wedge \phi_2)\theta = \phi_1\theta \wedge \phi_2\theta$, $(\neg\phi)\theta = \neg\phi\theta$, $(EX\{\mu\}\phi)\theta = EX\{\mu\theta\}\phi\theta$, $(\langle\mu\rangle\phi)\theta = \langle\mu\theta\rangle\phi\theta$ et $(EF\phi)\theta = EF\phi\theta$.

Définition 3.13 (Fonction de traduction)

Etant donnés une formule de π-logique ϕ et un ensemble de noms S, la traduction ACTL de ϕ noté «Ts (ϕ) » est la formule ACTL définie comme suit:

- Ts(true) = true
- Ts $(\phi_1 \wedge \phi_2)$ = $Ts(\phi_1) \wedge Ts(\phi_2)$
- Ts(EX{tau} ϕ) = EX{tau}Ts(ϕ)
- Ts(EX{$\bar{x}$ y} ϕ) = EX{$\bar{x}$ y}Ts(ϕ)
- Ts(EX{$\bar{x}$ (y)} ϕ) = $\bigvee_{\alpha \in S}$ EX{$\bar{x}$ (α)} Ts($\phi\theta$), où $\theta = \{\alpha/y\}$
- Ts(EX{x(y)} ϕ) = EX{x(y)}Ts(ϕ) V $\bigvee_{\alpha \in S}$ EX{x(α)} TS($\phi\theta$), où $\theta = \{\alpha/y\}$
- Ts(<tau> ϕ) = <tau>Ts(ϕ)
- Ts(<$\bar{x}$ y> ϕ) = <$\bar{x}$ y>Ts(ϕ)
- Ts(<$\bar{x}$ (y)> ϕ) = $\bigvee_{\alpha \in S}$ <$\bar{x}$ (α)> TS($\phi\theta$), où $\theta = \{\alpha/y\}$
- Ts(<x(y)> ϕ) = <x(y)>Ts(ϕ) V $\bigvee_{\alpha \in S}$ <x(α)>TS($\phi\theta$), où $\theta = \{\alpha/y\}$
- Ts(EF ϕ) = EFTs (ϕ)

Dans la définition précédente quand S = { } alors $\bigvee_{\alpha \in S}$ ϕ = false.

5 L'Architecture du HAL

Le HAL [24] [25] [26] est un environnement expérimental pour vérifier les automates à état fini des systèmes mobiles représentés dans le π-calcul. Il est intégré dans l'environnement JACK[4], qui fournit un ACTL model checker.

L'architecture de HAL est représentée dans la figure 5. L'implémentation courante contient cinq modules principaux qui sont tous intégrés à l'intérieur de l'environnement JACK. Trois de ces modules manipulent les traductions d'agents de π -calcul à un

[4] Pour plus d'information concernant JACK consulter : http://matrix.iei.cnr.it/projects/JACK.

HD-automate, de HD-automate à un automate ordinaire, et de formule de π-logique à une formule ACTL. Le quatrième module fournit plusieurs routines qui manipulent la représentation interne de HD-automate. La routine qui rend un HD-automate non redondant est contenue dans ce module. Le dernier module fournit au HAL une interface d'utilisateur graphique conviviale (GUI) Graphical User Interface. L'interface de l'utilisateur de HAL est divisée en deux côtés: le côté de l'agent et le côté logique.

HAL est écrit en C++ et compilé avec le GNU compilateur C++. La GUI est écrite en Tcl/Tk. HAL fonctionne actuellement sur une station SUN (sous SUN-OS) et sur les postes PC (sous Linux).

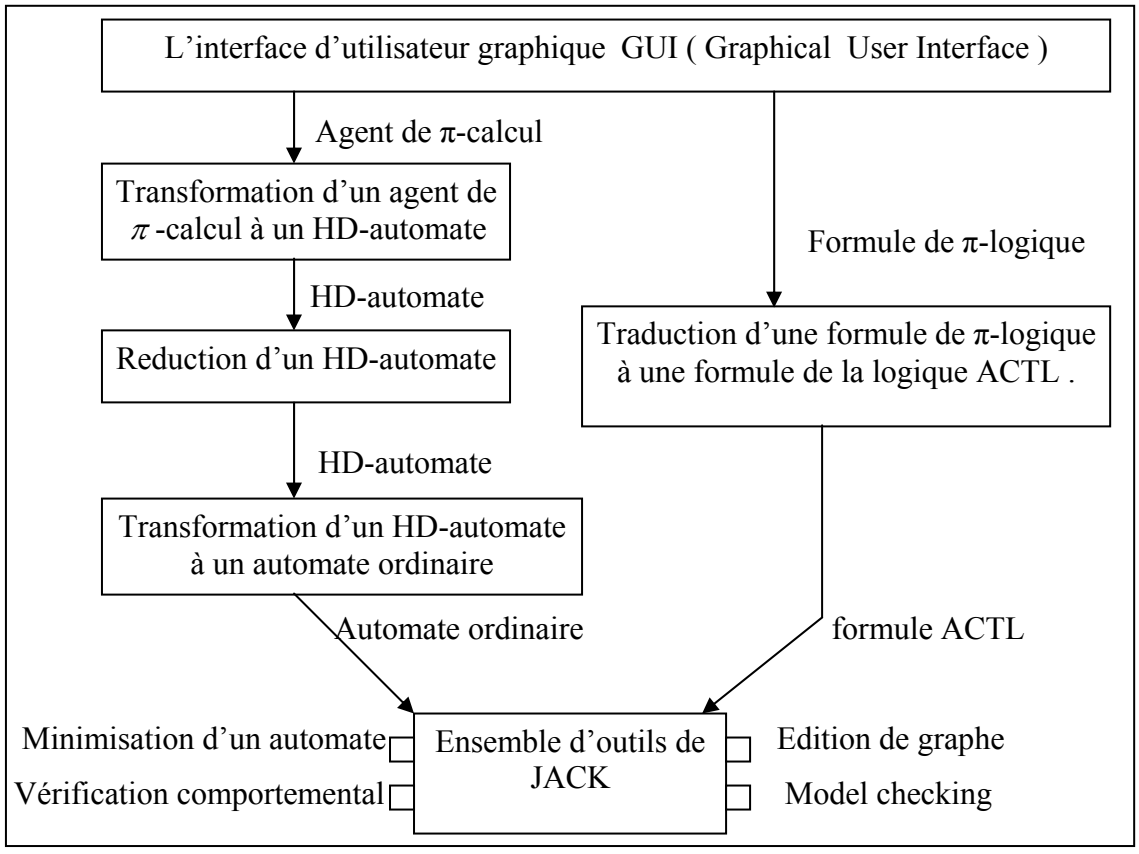

Figure 5 : L'architecture logique de l'environnement HAL

6 Conclusion

Dans ce chapitre nous avons présenté un model checking pour le π-calcul.
Nous avons présenté l'automate ordinaire d'un agent de π-calcul qui se fait grâce à un nouveau type d'automate appelé le HD-automate. Nous avons présenté la logique modale, la π-logique, et la logique ACTL. La traduction d'une formule de π-logique à une formule de la logique ACTL est aussi présentée. Enfin nous avons décrit le HAL : un environnement de vérification des processus mobiles.

Dans le chapitre suivant nous montrerons que le π-calcul peut être appliqué à un domaine plus spécifique, celui de la biologie.

Partie II

Application du π-calcul à la biologie

Chapitre 4

Notions de biologie

1 Introduction

Dans ce chapitre, nous présenterons les constituants moléculaires d'une cellule (protéines, ARN, ADN, glucides, lipides, acides aminés, ..).

Nous présenterons ensuite succinctement, pour le besoin de notre étude les étapes et le mécanisme de la synthèse des protéines. Nous introduisons aussi la régulation de la synthèse des protéines.

2 Constituants moléculaires d'une cellule

L'eau représente en moyenne 70% du poids cellulaire [32]. Elle contient les éléments suivants : glucides, lipides, acide aminé, protéine, ADN, ARN, etc.

2.1 Les glucides

- Les glucides sont utilisés comme une source d'énergie (glucose, fructose, ..).
- Ils sont un stockage d'énergie (exemples : amidon dans les cellules végétales et le glycogène dans les cellules animales).
- Ils sont des éléments de structure et de soutient des parois cellulaires et tissus **(**exemples : cellulose dans les cellules végétales et le peptidoglycanne pour les bactéries).
- Ils Participent à l'adhésion entre cellules (exemple glycoprotéines).

2.2 Les lipides

Les lipides caractérisés par leur faible solublité dans l'eau. Ils sont les constituants principaux des corps gras alimentaires et du tissu adipeux de l'organisme. L'oxydation des lipides fournit une grande quantité d'énergie.

2.3 Les acides aminés

Ce sont des petites molécules, solubles dans l'eau et cristallisables [32].

Il existe 20 acides aminés différents utilisés pour fabriquer les protéines. L'enchaînement et la succession des acides aminés formes les peptides.

2.4 Les protéines

Les protéines sont composées des enchaînements d'acides aminés [32].

Ces fonctions sont très variées et permettent de classer les protéines comme suit :

- les « protéines de structure » sont comparables à des briques cellulaires (ex : le collagène).
- les « protéines de transport » sont chargées du transport d'autres molécules dans la cellule ou entre les cellules d'un organisme (ex : l'hémoglobine transporte l'oxygène).
- Les enzymes permettent d'accélérer les réactions chimiques nécessaires à la vie (ex : la glucose 6 phosphatase initie la dégradation du glucose, notre principale source de l'énergie cellulaire).
- Les protéines de l'immunité (ou anticorps) contribuent à la défense de notre organisme.

2.5 L'Acide DésoxyriboNucléique (ADN)

L'ADN est une macromolécule polynucléotique. Elle constitue le matériel héréditaire de la plupart des êtres, à l'exception des ribovirus dont le matériel génétique est composé d'ARN [32] [83].

2.5.1 Structure

L'ADN a une structure en hélice double. La macromolécule d'ADN est composée de deux chaînes polynucléotidiques, formées chacune de nombreux nucléotides. Chaque nucléotide comprend lui-même un acide phosphorique lié à un sucre, qui est combiné à une base [32]. Il existe quatre sortes fondamentales de bases : A (Adénine), C (Cytosine), G (Guanine) et T (Thymine).

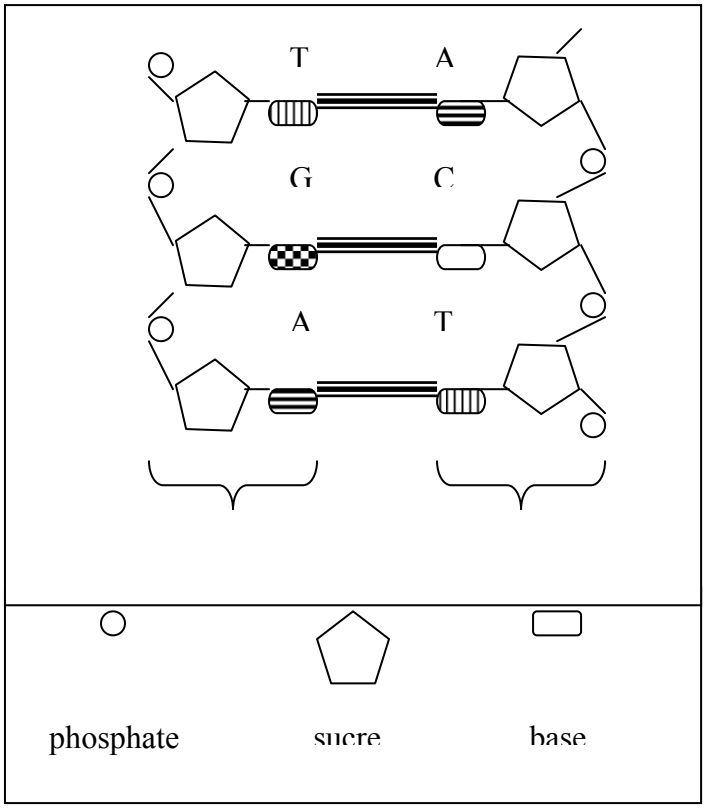

Figure 6 : Composition de l'ADN

Le squelette de la double hélice est constitué d'associations deux à deux des bases azotées (« lettre ») selon l'ordre imposé pour un codon. L'adénine (A) se lie toujours avec la thymine (T), la cytosine (C) toujours avec la guanine (G). Ces molécules seront face à face et se stabiliseront par des interactions appelées liaisons hydrogènes.

2.5.2 Fonctions

Les biologistes attribuent à l'ADN deux rôles fondamentaux : il est le support de l'information génétique, et permet la transmission des informations génétiques de cellule en cellule et de génération en génération.

2.5.2.1 Support de l'information génétique

L'ordre de la succession des bases dans l'ADN constitue le code génétique. Il contient l'information génétique au travers des gènes. Ce code génétique permet en particulier la synthèse des protéines.

2.5.2.2 Transmission de l'information : la réplication

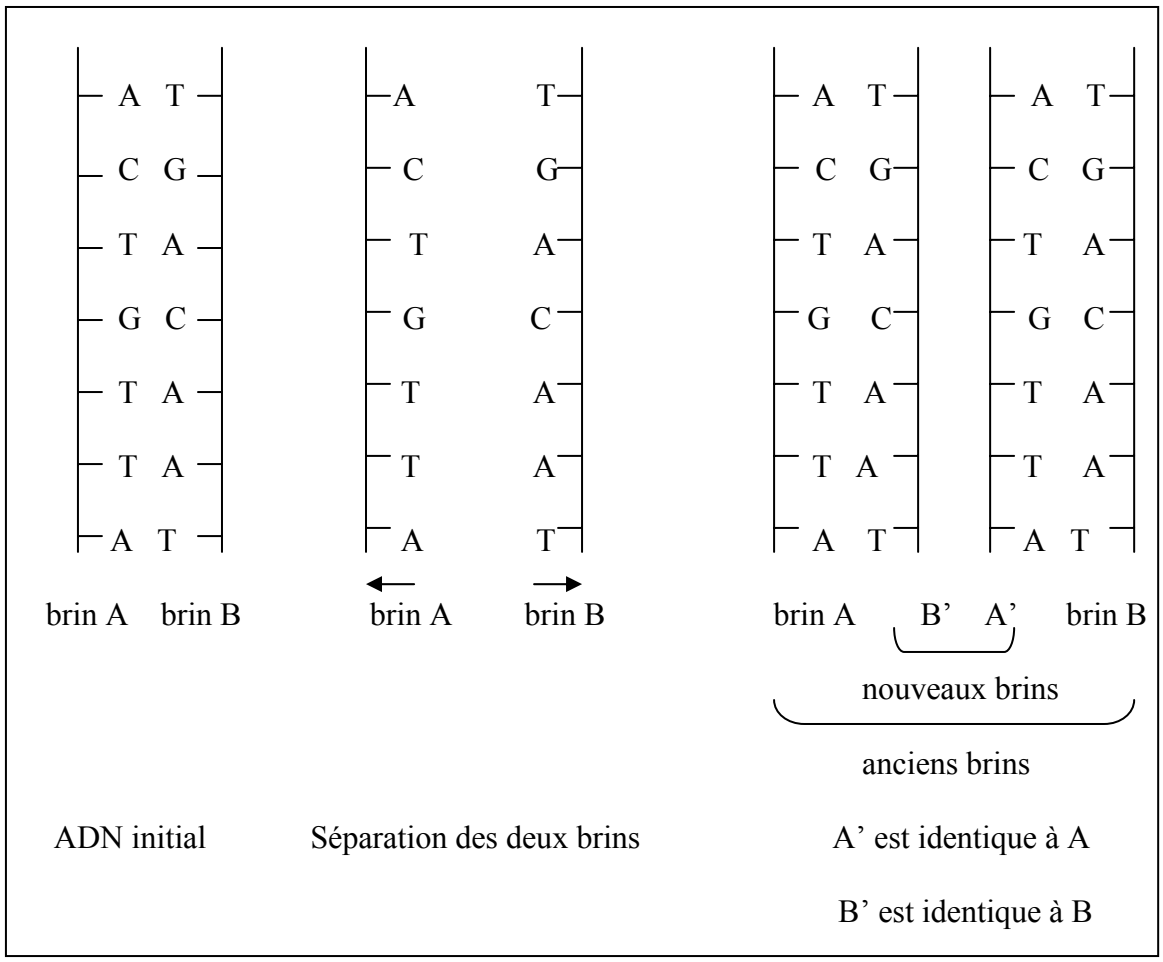

Figure 7 : Réplication de l'ADN

Lorsqu'une cellule se divise pour donner deux cellules filles, il faut que l'ADN de ces cellules – filles soit une copie exacte de l'ADN de la cellule - mère, d'où le nom « réplication » [19].

Pour que la réplication puisse avoir lieu, les deux brins de la molécule d'ADN s'ouvrent, un peu comme une fermeture à glissière. Chaque brin est alors parcouru par une molécule d'une enzyme spécifique, l'ADN polymérase. Cette dernière synthétise un nouveau brin, complémentaire du premier, en accolant bout à bout des bases libres. A la fin du processus, au lieu d'une, il y a désormais deux molécules d'ADN : chacune est constituée d'un brin nouvellement formé et d'un ancien, la réplication est dite semi conservative.

2.6 L'Acide RiboNucléique (ARN)

La molécule d'ARN est plus petite que celle de l'ADN. On en trouve dans le noyau beaucoup plus que dans le cytoplasme cellulaire : ARN messager, ARN de transfert, ARN ribosomal. Ces différentes formes d'ARN notées respectivement ARN_m , ARN_t et ARN_r

Correspondent à des fonctions précises de l'ARN [37].

2.6.1 Structure

Il y a analogie entre l'architecture des molécules de l'ARN et de l'ADN. Les molécules d'acide phosphorique sont liées entre elles et avec un sucre qui est ici le ribose (dans l'ADN, c'est le désoxyribose). Sur ce dernier est fixée une base azotée : adénine (A), guanine (G), cytosine (C) ou uracile (U) [37]. L'ARN est constitué d'une seule chaîne de nucléotides.

2.6.2 Fonctions de l'ARN

ARN messager :

- L' ARN_m se fabrique dans le noyau en contact d'un brin codant d'ADN par complémentarité des bases, l'uracile remplace la thymine.
- La transcription s'effectue grâce à une enzyme : l'ARN polymérase capable d'agencer les nucléotides en molécules d'ARN.
- ARN_m transmet l'information génétique aux organites cytoplasmiques [37].

ARN de transfert :

- Sert à transporter chacun des acides aminés vers le lieu où se fabriqueront les protéines [79].
- Il existe 20 ARN_t différents, correspondant chacun à l'un des 20 acides aminés. À l'une des extrémités de l' ARN_t se trouve l'anticodon dont le rôle est de se positionner face à un codon de l' ARN_m pour présenter au ribosome l'acide aminé qui sera fixé à la chaîne peptidique [37].

ARN ribosomale :

- Les ribosomes sont des petits « grains » situés dans le cytoplasme où s'effectue la synthèse des protéines. Elles sont constituées d'ARN et de protéines [37].
- Ce sont les véritables « usine à protéines » de la cellule.
- Les ribosomes ont un rôle comme une tête de lecteur vis-à-vis de l'ARN_m .

3 Gènes et code génétique

Le code génétique ne peut pas être utilisé, on dit transcrit, que lorsqu'il est présent dans un fragment très particulier de l'ADN, appelé gène.

On appelle gène un ensemble de nucléotides qui contient toute l'information necessaire pour produire , ou transcrire , un ARN messager susceptible, dans un second temps appelé traduction, de fabriquer une protéine [79].

3.1 Structure générale d'un gène

Un gène est un segment d'ADN qui contient toutes les informations nécessaires à la fabrication d'un ARN messager. La longueur d'un gène est très variable, de même sa structure.

Un gène comprend trois séquences d'ADN distincts :

1. Celle qui contient les éléments qui vont en régler la transcription, c'est la zone régulatrice.
2. Celle qui comprend les portions codantes du gène (celles qui possèdent le code génétique).
3. Enfin la structure d'un gène inclut également des nucléotides qui n'ont aucun rôle connu particulier et qui sont simplement inclus dans la structure probablement par hasard.

4 La synthèse des protéines

La synthèse des protéines [37] consiste à transposer une séquence de nucléotide en une suite d'acides aminés. Le processus s'effectue en deux étapes :

1. La transcription.
2. La traduction.

4.1 La transcription

La molécule d'ADN, qui contient sous forme codée les instructions nécessaires pour la synthèse des molécules protéiques spécifiques, s'ouvre par fissuration au niveau des liaisons hydrogènes entre bases azotées. Ceci démasque les séquences d'ADN codant. Des nucléotides (phosphore – désoxyribose – base) présents dans le noyau cellulaire se placent face au brin

codant qui sert de modèle. Ces nucléotides ne se disposent pas au hasard mais selon des règles de complémentarité de bases : la cytosine face à la guanine, l'uracile face à l'adénine, l'adénine face à la thymine, la guanine face à la cytosine.

Une enzyme assemble ensuite les nucléotides en un ARN messager complémentaire de l'ADN codant ou instructeur constituant le gène.

4.2 La traduction

Cet ARN messager quitte le noyau cellulaire vers le cytoplasme où il porte le message, l'information génétique nécessaire pour synthétiser la protéine spécifique. Cet ARN_m a une durée de vie courte (de quelques minutes à quelques heures). Pendant ce laps de temps, son message est lu et traduit en tenant compte du code génétique qui attribue à chaque suite de trois bases de l'ARN_m un acide aminé précis. Ce sont les ribosomes qui assument la lecture et la synthèse.

	2éme nucléotide				
1er nucléotide	U	C	A	G	3éme nucléotide
U	phe	ser	tyr	cys	U
	phe	ser	tyr	cys	C
	leu	ser	stop	stop	A
	leu	ser	stop	try	G
C	leu	pro	his	arg	U
	leu	pro	his	arg	C
	leu	pro	gln	arg	A
	leu	pro	gln	arg	G
A	ileu	thr	asn	ser	U
	ileu	thr	asn	ser	C
	ileu	thr	lys	arg	A
	met	thr	lys	arg	G
G	val	ala	asp	gly	U
	val	ala	asp	gly	C
	val	ala	glu	gly	A
	val	ala	glu	gly	G

Table 6 : Le code génétique

Il y a 20 acides aminés différents qui sont amenés par des ARN de transfert.

Par exemple, la séquence AUG de l'ARN_m appelle l' ARN_t complémentaire UAC, correspondant à la tyrosine. Ces acides aminés sont branchés en une chaîne qui s'allonge progressivement pour former un peptide, un polypeptide puis une protéine. Plusieurs ribosomes entreprennent simultanément la lecture d'une même molécule d' ARN_m .

La lecture de ARN_m par les ribosomes débute au triplet AUG qui marque le début du message et amorce ou initie la synthèse. La lecture se termine à l'un des triplets UAA , UAG ou UGA qui n'ont pas de signification en termes d'acide aminé et indiquant que le message est terminé.

5 Principe de la régulation de l'activité des gènes

Le nombre de molécules protéiques produites par unité de temps varie d'un gène à l'autre, afin de satisfaire les besoins de la cellule sans conduite à des synthèses inutiles.

L'expression de gènes particuliers est controlée par des mécanismes dénommés régulation des génes [31].

Les mécanismes de régulation peuvent etre regroupés en deux catégories : la régulation positive et la régulation négative (figure 8).

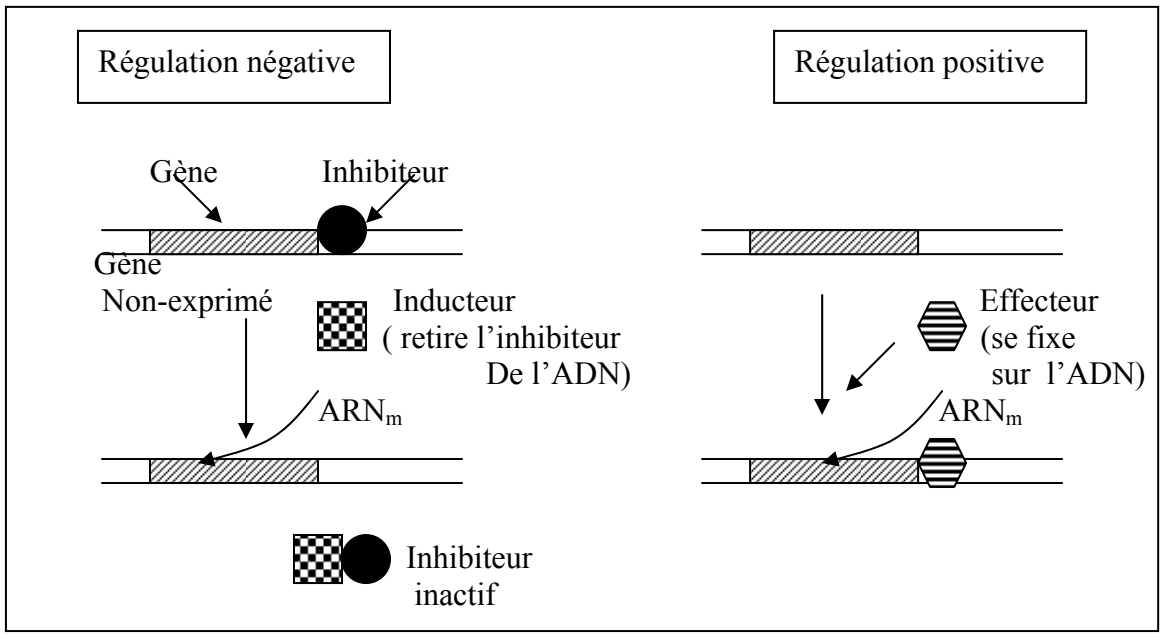

Figure 8 : Comparaison entre régulation négative et régulation positive

Dans le cas d'une régulation négative, l'initiation de la transcription nécessite le départ d'un inhibiteur fixé sur l'ADN. Dans le cas d'une régulation positive, l'initiation de la transcription requiert la fixation d'une molécule effectrice. Un système peut etre régulé à la fois positivement et négativement. Dans ce cas, l'intiation de la transcription ne se fait que lorsque l'effecteur positif est fixé à l'ADN et que l'inhibiteur n'y pas fixé.

Dans un système de régulation négatif, un inhibiteur, présent dans la cellule, bloque la transcription. Un antagoniste de l'inhibiteur, généralement appelé inducteur, est alors nécéssaire pour initier la transcription. Dans un système de régulation positif, une molécule effectrice (qui peut etre une protéine, une petite molécule ou un complexe moléculaire) active un promoteur[5].

Les systèmes de régulation positive et négative ne sont pas exclusifs et quelques systèmes sont à la fois positivement et négativement régulés.

6 Conclusion

La cellule est l'unité structurale de tous les êtres vivants et l'ADN est l'un des constituants du noyau de cette cellule.

L'ADN est une molécule d'importance biologique fondamentale, car elle constitue le support de l'information génétique.

La synthèse des protéines se fait selon un mécanisme bien précis, suivant deux étapes : la transcription et la traduction.

Les réseaux de régulations régissant l'activité de la cellule afin de s'adapter en permanence aux variations de son environnement.

Le chapitre suivant montre une modélisation formelle des processus biologiques par le π-calcul stochastique.

[5] C'est une courte séquence spécifique d'ADN, située au début des gènes, sur laquelle se fixe l'enzyme qui effectue la transcription (l'ARN polymérase). Etant nécessaire pour que la transcription débute, le promoteur est indispensable au fonctionnement d'un gène.

Chapitre 5

Modélisation et simulation des processus biologiques par le π-calcul stochastique

1 Introduction

Dans ce chapitre nous présenterons la modélisation et la simulation des processus biomoléculaires par le π-calcul stochastique.

Nous commencerons le chapitre par la présentation des modeles existants pour la modélisation des processus biologiques. Nous présenterons aussi le système biomoléculaire dans le π-calcul. L'algorithme utilisé pour la simulation d'un programme codé en π-calcul stochastique sera aussi présenté. Nous terminons le chapitre par la description de deux utilitaires implémentés pour la simulation d'un programme codé en π-calcul stochastique.

2 Les modèles existants pour les processus biologiques

Plusieurs approches ont été proposées dans le cadre de la modélisation formelle des processus biomoléculaires. Elles peuvent être divisées en quatre catégories :

2.1 Les modèles cinétiques chimiques

Les modèles cinétiques chimiques appartiennent à la théorie des systèmes dynamiques qui modélise les systèmes biomoléculaires en se basant sur l'abstraction[6] « *la cellule est considérée comme une collection d'espèce moléculaire*s ». De tels modèles décrivent les systèmes moléculaires d'une vision biochimique pure [1] [3] [8]. Bien que cette approche est claire et extrêmement puissante avec une origine théorique étendue et une variété de méthodologies et outils, mais elle manque de la pertinence et de la compréhension quand nous manipulons des systèmes biologiques.

2.2 Les modèles généralisés de régulation

Les modèles du réseau booléèns utilisent l'abstraction « *une molécule comme expression logique* » pour décrire et simuler les circuits de régulateurs de gène. Ils ont été présenté par Kaufmann et appliqué par la suite sur plusieurs systèmes [80] [70].

Bien que cette approche est prouvée utile quand nous étudions des propriétés générales de grand réseaux et quand nous manipulons un système où seulement les connaissances qualitatives sont disponibles, cependant elle ne peut pas être considérée comme un modèle général de système biomoléculaire, car elle manque de pertinence, de calculabilité et d'extensibilité.

2.3 Les bases de données orienté objet fonctionnel

Les bases de données du chemin sont basées sur l'abstraction « *molécule comme objet* ». Plusieurs bases de données ont été implémentées [4] [81] qui fournissent des outils de requêtes.

Bien que ces bases de données nous permettent de bien organiser et manipuler et quelques fois visualiser les données de chemin, mais leurs capacités dynamiques (exemple simulation) et leurs outils de requêtes sont limités. Des langages d'échanges ont été développés pour minimiser ces limitations en fournissant des moyens permettant d'intégrer des modèles et des outils de plusieurs sources. La plupart sont basés sur un langage de majoration XML [29].

[6] Une abstraction est une correspondance d'un domaine de monde réel à un domaine mathématique qui conserve quelques propriétés essentielles du domaine de monde réel en ignorant d'autres propriétés, considérées superflues.

2.4 Les langages des processus abstraits

Cette approche est basée sur l'abstraction « *molécule comme calcul* ». Elle a été fondée par Fontana et Buss [30] qui ont utilisé le λ-calcul et la logique linéaire.

Certains systèmes biomoléculaires ont été modélisés en utilisant des formalismes de calculs concurrents. Parmi eux on cite les réseaux de Petri qui ont été utilisé pour la représentation, la simulation et l'analyse des chemins métaboliques [36] [46]. Bien qu'ils soient clair et utilisent des outils analytiques, cependant ils manquent de pertinence et sont utilisés seulement pour l'étude des chemins métaboliques avec une vision chimique.

Certaines études plus récentes tentent de surmonter les limitations des réseaux de Petri en les combinant avec une présentation orientée objet.

D'autres études, utilisent le diagramme d'états [40] pour construire des modèles graphiques qualitatifs essentiellement pour les systèmes cellulaires avec un composant moléculaire.

Le diagramme d'états est un langage des processus riche et expressif avec une sémantique claire. Il permet la représentation qualitative d'un système biologique complexe, mais il ne manipule pas les aspects quantitatifs.

Aviv Regev, William Silverman et Ehud Shapiro ont défini une approche basée sur l'abstraction « *molécule comme calcul* », qui utilise le π-calcul et ses extensions [69] [65] [67] (celle-ci sera présentée dans les paragraphes ci-après).

L'usage du π-calcul dans la biologie moléculaire a été aussi discuté par Fontana et Ciobanu [14].

3 Le π-calcul : Modèle abstrait pour les systèmes biomoléculaires

La construction de l'abstraction nécessite trois étapes:

1- *Organisation* informelle des connaissances sur le domaine du monde réel, en identifiant les entités essentielles dans ce domaine, leurs propriétés et leurs comportements.

2- *Sélection* d'un domaine mathématique en établissant une correspondance informelle entre les propriétés ou les comportements essentiels dans le domaine du monde réel et ceux du système mathématique.

3- *Conceptions et exécutions* de l'abstraction en utilisant des directives *pragmatiques* pour construire la représentation des entités du monde réel par le modèle mathématique choisi.

3.1 Le domaine du monde réel: Propriétés essentielles des systèmes biomoléculaires

Les aspects essentiels d'un système biomoléculaire sont présentés dans la figure 9.

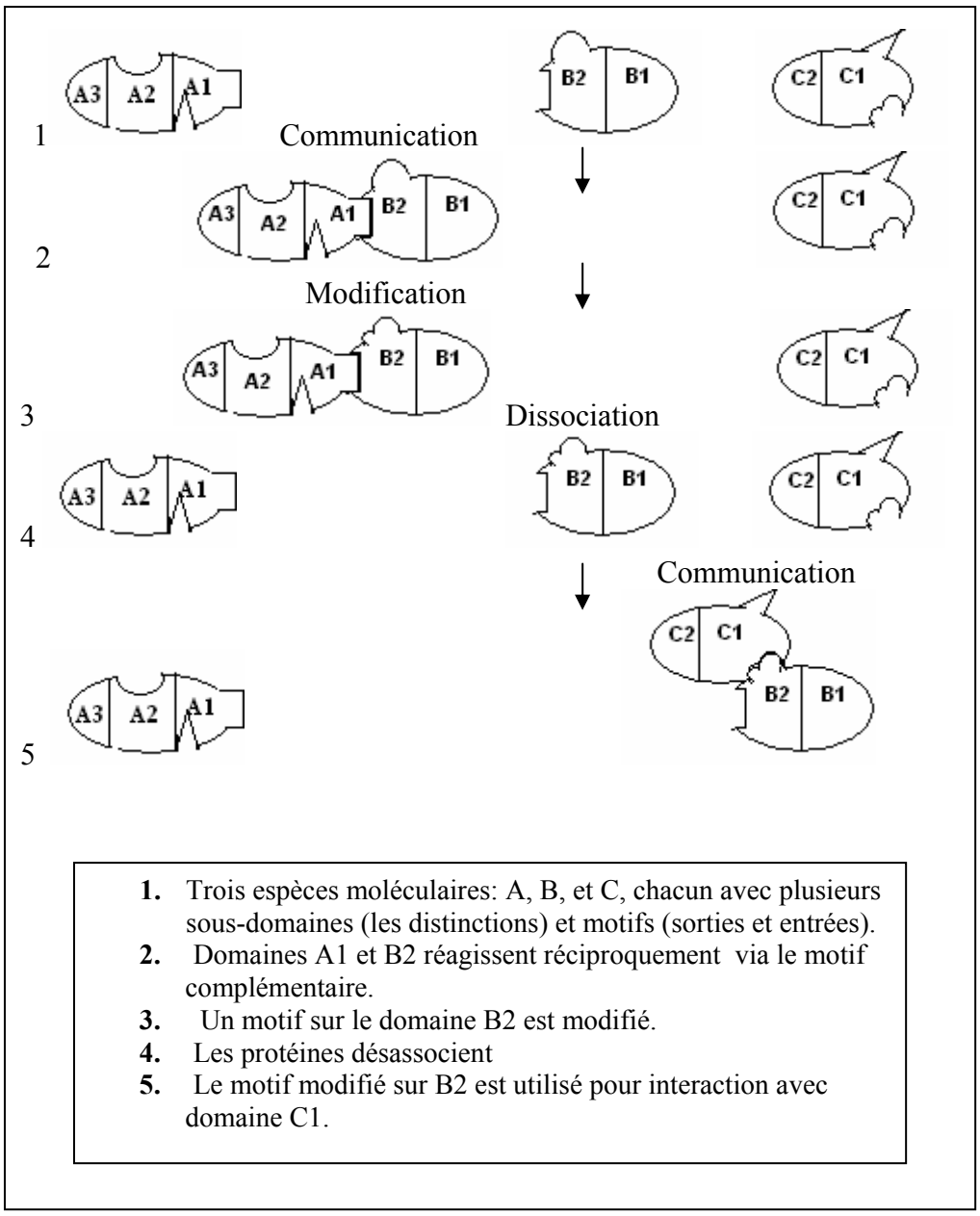

Figure 9 : Un système biomoléculaire

1- Composition Moléculaire

Chaque molécule de la protéine peut être composée de plusieurs domaines structurellement indépendants et de parties fonctionnelles (1 de la figure 9). Chaque domaine peut avoir un ou plusieurs motifs structurels chimiques (1 de la figure 9).

2- Complémentarité et interaction spécifique

Des domaines différents interagissent entre eux à travers des motifs complémentaires. (Figure 9 - 1, 2).

3- Les résultats de l'interaction: Reconstitution, modification et changement dans l'état moléculaire

Une molécule peut être reconstituée sans aucun changement ou peut être changée (figure 9-3,4). Ce changement peut être attribué à une modification chimique spécifique d'un des motifs dans la molécule ou bien par un changement plus général de l'état de la molécule, tel qu'un changement d'un état inactif à un état actif.

4- Interactions Alternatives

Dans beaucoup de cas une molécule peut être capable de participer à plus d'une interaction.

3.2 Le domaine mathématique: Calcul concurrent

Les programmes du π-calcul spécifient des réseaux de processus communicants. La communication se produit sur des canaux complémentaires qui sont identifiés par des noms spécifiques (figure 10).

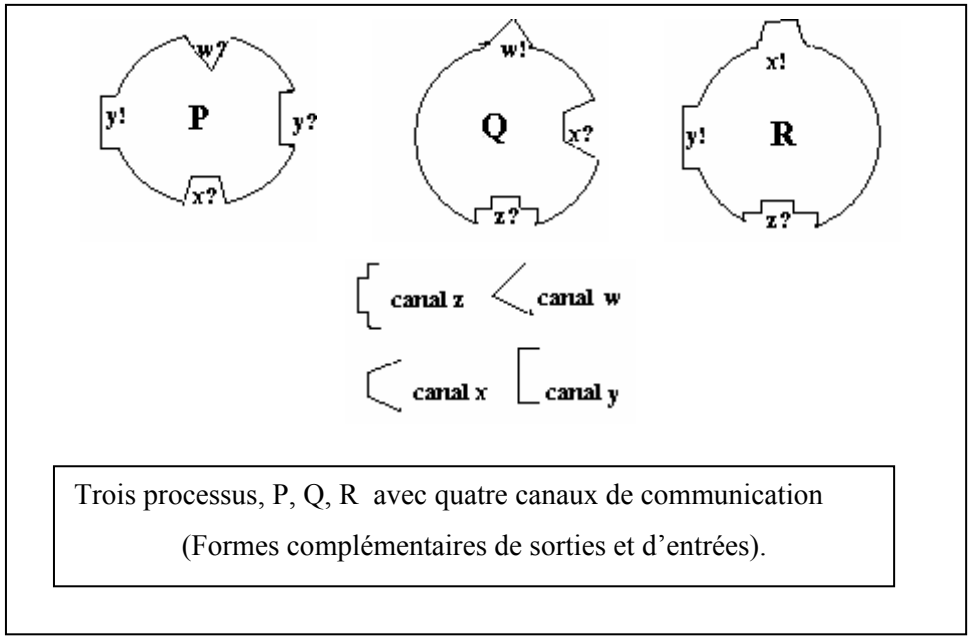

Figure 10 : Les processus et les canaux de π-calcul : Une vue intuitive

Il y a deux types de communication dans le π-calcul : non mobile et mobile (Figure 11).
Dans la communication non mobile, le processus envoie seulement une alerte (message vide « *nil* ») aux autres processus. Tandis que dans le cas d'une communication mobile les processus peuvent envoyer des canaux entre eux. Le passage des canaux comme des messages permet aux processus de modifier dynamiquement leurs capacités de communication.

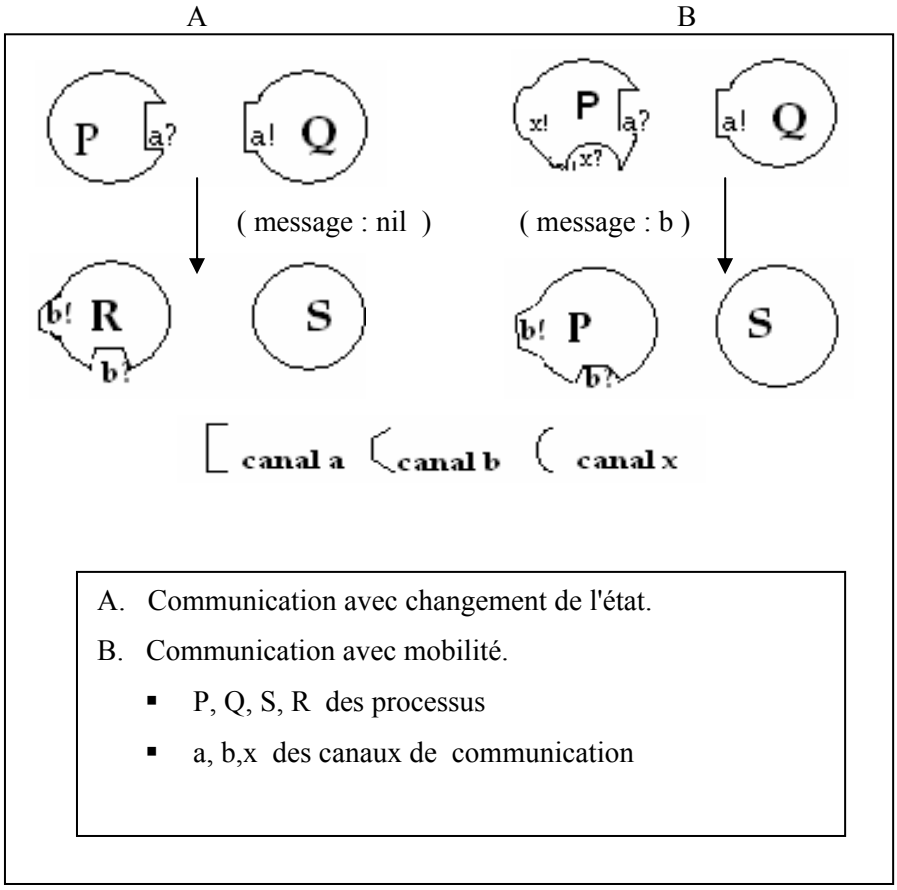

Figure 11 : Communication du π-calcul: Une vue schématique

3.3 L'abstraction « *molécule comme calcul* »

Les directives générales de la construction de la représentation d'un système biomoléculaire dans le π-calcul sont données dans la table 7.

Entité biomoléculaire	**Entité de π-calcul**
Espèce moléculaire	Espèce de processus
Population moléculaire	Système des processus concurrents
Types de motif complémentaires	Entrée et sortie complémentaire du même nom de canal
L'occurrence du motif dans la molécule ou dans le domaine	La communication se passe sur des canaux dans les processus
Événement biomoléculaire	**Evénement du π-calcul**
Interaction spécifique sur motif complémentaire	Communication spécifique sur des canaux complémentaires
Résultats qui suivent l'interaction	Préfixe de la communication qui précède la création du processus ou autre communication
Reconstitution qui suit l'interaction	Préfixe de la communication qui précède la récréation du processus
État moléculaire changé qui suit l'interaction	Préfixe de la communication qui précède la création de nouveau processus
Modification de motifs pendant l'interaction	Approche non-mobile: ré-création d'un type différent de processus avec un ensemble de canaux différents
	Approche mobile: Message (un tuple de noms de canaux) à envoyer dans une communication pour remplacer des canaux dans le processus de la réception

Table 7 : Directives pour l'abstraction de système biomoléculaire dans le π-calcul

4 Le système biomoléculaire dans le π-calcul

On va présenter les détails des techniques de l'abstraction de processus biomoléculaires dans le π-calcul.

1- Molécules et domaines comme processus concurrents

- Chaque système biomoléculaire est représenté par un processus, dénoté par un nom capitalisé.
- Chaque molécule constituante dans le système biomoléculaire ainsi que ses domaines sont représentés par des processus.
- Un système de processus est une collection de processus de molécule parallèles. Cette concurrence est dénotée par l'opérateur PAR (|), inséré entre les noms des processus.
- D'une manière analogue, nous définissons une molécule de protéine comme plusieurs processus de domaine composés en parallèle.

2- La complémentarité moléculaire comme canaux de communication

- Deux molécules (ou domaines) interagissent entre eux suivant leur complémentarité structurelle et chimique. Ceci est représenté en π-calcul par la complémentarité offerte sur les canaux de communication.
- Les motifs sur les molécules peuvent varier souvent en se basant sur des modifications biochimiques différentes. Un tel motif est souvent représenté comme un canal paramétrique du processus.

3- Événements séquentiels et mutuellement exclusifs

- Les événements biochimiques peuvent se produire en séquence, ceci est représenté dans le π-calcul par le processus qui suit l'exécution d'une action.
- Ils peuvent être exécutés dans un mode mutuellement exclusif, ceci est représenté dans le π-calcul par l'opérateur de la somme (+).

4- Compartiments

- Les molécules qui partagent un compartiment commun peuvent réagir réciproquement l'un avec l'autre, cependant les molécules exclues du compartiment ne le peuvent pas. Ceci est représenté en π-calcul en introduisant des canaux "privés" avec les scopes de communications restreintes.

5- Interaction et modification biochimiques comme communication

- L'interaction et la modification biochimiques sont représentées par des communications dans le π-calcul. On peut utiliser soit une communication mobile ou non mobile.

Communication non mobile :

Dans une communication non mobile, nous présentons une modification ou un changement d'état d'un processus par un nouveau processus créé dans le système.

Communication mobile :

Dans une communication mobile, nous présentons une modification d'une molécule comme un ou plusieurs canaux envoyés entre les processus. Ce qui permet au même processus de continuer de représenter la molécule avec des modifications faites à ce dernier. Malgré que l'approche mobile est plus difficile que l'approche non mobile, mais elle est fermée sur le raisonnement biologique.

6- Changement de compartiment comme scope d'extrusion d'un canal privé

- Nous utilisons aussi le mécanisme de la mobilité pour faire l'abstraction de changement de compartiment, tel que la formation de complexe.

7- Objets moléculaires comme processus paramétriques

- L'usage de définitions du processus paramétriques nous permet de faire l'abstraction de l'identité constante d'une molécule (processus) comme sa structure (canaux publics) et son compartiment (canaux privés) modifié par l'interaction.

8- Compétition comme choix

- Une molécule peut participer souvent à un ou plusieurs interactions mutuellement exclusives. Nous faisons l'abstraction de ceci par l'opérateur de choix (la somme).

Exemple

Si on considère le système biomoléculaire représenté dans la figure 9, on peut le modéliser par le π-calcul par le code suivant :

La population moléculaire est représentée par un seul processus noté « Pop_mol », ce processus est composé par trois processus A, B et C exécutés en parallèle :

Pop_mol = A | B | C

Les trois espèces de molécules sont composées par des sous-domaines représentés par des processus parallèles :

A = A1 | A2 | A3

B = B1 | B2

C = C1 | C2

Les motifs dans les sous domaines sont représentés par des canaux d'entrée ou de sortie, tel que :

A2 = x( ).A2

A1 = y( ).A1 + $\bar{z}$ w.A1

B2(m) = z(m).B2(m) + $\bar{m}$ .B2(m)

C1 = w( ).C1 + $\bar{y}$ .C1

La communication effectuée dans le cas 2 est représentée par la réaction suivante :

$$B2(x) \mid A1 = (z(x).B2(x) + \bar{x}.B2(x)) \mid (y( ).A1 + \bar{z}\, w.A1) \xrightarrow{\tau} A1 \mid B2(w)$$

Ce qui correspond à un changement du motif x de B2 à un motif w.

Donc, la molécule B deviendra capable d'effectuer une communication avec la molécule C.

La communication effectuée dans le cas 5 est représentée par la réaction suivante :

$$B2(w) \mid C1 = (z(w).B2(w) + \bar{w}.B2(w)) \mid (w( ).C1 + \bar{y}.C1)$$

En utilisant ces techniques de modélisation, on peut représenter le système biomoléculaire dans le π-calcul. Les opérations sémantiques de ce calcul définissent le comportement dynamique du système modélisé.

On peut avoir plusieurs communications possibles à un instant donné. La sélection de la communication suivante est non déterministe, c à d que toutes les communications ont une chance égale de se produire. Ce qui ne correspond pas à la réalité, où il y a des communications qui sont plus probables que d'autres.

Pour résoudre ce problème, la version stochastique de ce calcul a été exploitée pour représenter les informations quantitatives.

La notion « taux de base » correspond à la même définition que « le taux d'activité » qui a été déjà défini dans le chapitre 2.

5 Simulation d'un programme codé en π-calcul stochastique :

La version stochastique du π-calcul a été exploitée pour représenter les informations quantitatives. L'évolution du temps dans la simulation suit l'algorithme de Gillespie décrit ci-dessous. Il permet de déterminer le temps d'apparition du prochain évènement et le canal de la réaction suivante. BioSpi [67] et SPiM [60] [61] sont deux simulateurs implémentés pour la simulation d'un programme codé par le π -calcul stochastique.

5.1 L'algorithme de Gillespie

5.1.1 Introduction

Dans [33] l'auteur présente une formulation stochastique d'un système chimique cinétique basé sur la théorie de collisions et qui suppose une constante stochastique de la réaction c_μ pour chaque réaction chimique R_μ. La constante de la réaction c_μ est tel que $c_\mu dt$ est la probabilité moyenne qu'une combinaison particulière de réactifs de R_μ interagissent entre eux dans un intervalle de temps infinitésimal *dt*. La probabilité qu'une réaction R_μ se produise dans la solution dans un intervalle de temps infinitésimal *dt* est donnée par $c_\mu dt$ multiplié par le nombre de combinaison moléculaire de réactif distinct de la réaction R_μ. Par exemple la réaction citée dans [33] :

$R_1 : S_1 \mid S_2 \longrightarrow 2S_1$ (1)

Se produira dans la solution avec un nombre X_1 de molécules S_1 et avec X_2 de S_2.

Et la probabilité sera $X_1 X_2 c_1 dt$. Tandis que la réaction inverse :

$R_2 : 2S_1 \longrightarrow S_1 \mid S_2$ (2)

Se produira avec une probabilité $\frac{X_1(X_1-1)}{2!} c_2 dt.$

Le nombre de combinaisons de molécule de réactif est dénoté h_μ par Gillespie, la probabilité que R_μ se produira pendant le temps *dt* est noté $a_\mu dt$ tel que : $a_\mu dt = h_\mu c_\mu dt$.

5.1.2 Formulation de problème

Si on a N espèces chimiques $S_1, S_2, ..., S_N$ chacune avec une quantité $X_{1,}, X_2, .., X_N$ respectivement dans un volume V. Les S_i (i=1,..,N) peuvent interagir à travers M réactions chimiques R_i (i=1,..,M) . Chaque réaction chimique R_i (i=1,..,M) possède une constante stochastique c_i (i=1,..,M) .

Quelle est l'évolution de cette population dans le futur ?

Pour la simulation stochastique de temps d'évolution d'un système chimique nous avons besoin de définir les deux paramètres suivants :

1- Le temps dans lequel la prochaine réaction se produira noté « τ ».

2- μ qui représente le type de la réaction R_μ (μ=1,.., M) qui se produira dans l'intervalle de temps τ .

La fonction $P(\tau,\mu)dt$ représente la probabilité que la prochaine réaction se produira dans la solution dans un intervalle de temps infinitésimal $[t+\tau, t+\tau+d\tau]$ et elle sera R_μ.

$P(\tau,\mu)dt = P_0(\tau).a_\mu dt.$

Où $P_0(\tau)$ représente la probabilité qu'aucune réaction ne se produira dans l'intervalle $[t, t+\tau]$.

$$P_0(\tau) = exp\left(-\sum_{v=1}^{M} a_v \tau\right)$$

Donc $P(\tau,\mu)dt = exp\left(-\sum_{v=1}^{M} a_v \tau\right) a_\mu dt$.

Finalement on peut choisir τ et μ comme suivant :

$$\tau = \left(\frac{1}{\sum_{v=1}^{M} a_v}\right) ln\left(\frac{1}{r_1}\right) \qquad \text{et } \mu \text{ est l'entier lequel } \sum_{v=1}^{\mu=1} a_v < r_2 \sum_{v=1}^{M} a_v \leq \sum_{v=1}^{\mu} a_v .$$

Où r_1 , r_2 sont deux nombres réel générés par un générateur de nombre aléatoire .

5.1.3 Les étapes de l'algorithme de Gillespie

L'algorithme de Gillespie est résumé dans la figure 12. Les étapes de l'algorithme sont les suivantes :

Etape 0 : (Initialisation)

- Entrer les valeurs de M constantes $c_1, c_2, ..c_M$ de M réactions chimiques.
- Entrer les N nombres $X_1, X_2, ..X_N$ de la population moléculaire initiale.
- Initialiser la variable temps t et le compteur de la réaction n à zéro.
- Initialiser l'intervalle de l'unité générateur du nombre aléatoire uniformément (URN).

Etape 1 :

- Calculer et enregistrer les M quantités $a_1 = h_1 c_1, .. a_M = h_M c_m$ pour les nombres de la population moléculaire courante, où h_μ (μ=1,..,M) sont calculés en fonction ($X_1, X_2, .. X_N$) ($\mu = 1, .. M$) comme décrit précédemment .
- calculer et enregistrer a_0 qui est égal à la somme des valeurs a_v (v=1...M) :

$$a_0 = \sum_{v=1}^{M} a_v = \sum_{v=1}^{M} h_v c_v$$

Etape2 :

- Générer deux nombres aléatoires r_1 et r_2 en utilisant l'unité de générateur du nombre aléatoire uniformément.
- calculer τ le temps d'apparition de la prochaine réaction et μ le type de réaction ($\mu = 1, .. M$) tel que :

  $\tau = (1/a_0)\ln(1/r_1)$

  Et soit μ l'entier lequel $\sum_{\nu=1}^{\mu=1} a_\nu < r_2 a_0 \leq \sum_{\nu=1}^{\mu} a_\nu$

Etape3 :

- Utiliser les valeurs du τ et μ obtenues dans l'étape 2 en augmentant t par τ, et en ajustant les niveaux moléculaires de la population pour refléter l'occurrence d'une réaction R_μ .
- augmenter le compteur de la réaction n par 1 et revenir à l'étape 1.

La boucle 1-2-3 peut être arrêtée si t ou n atteint certaines valeurs prédéterminées.

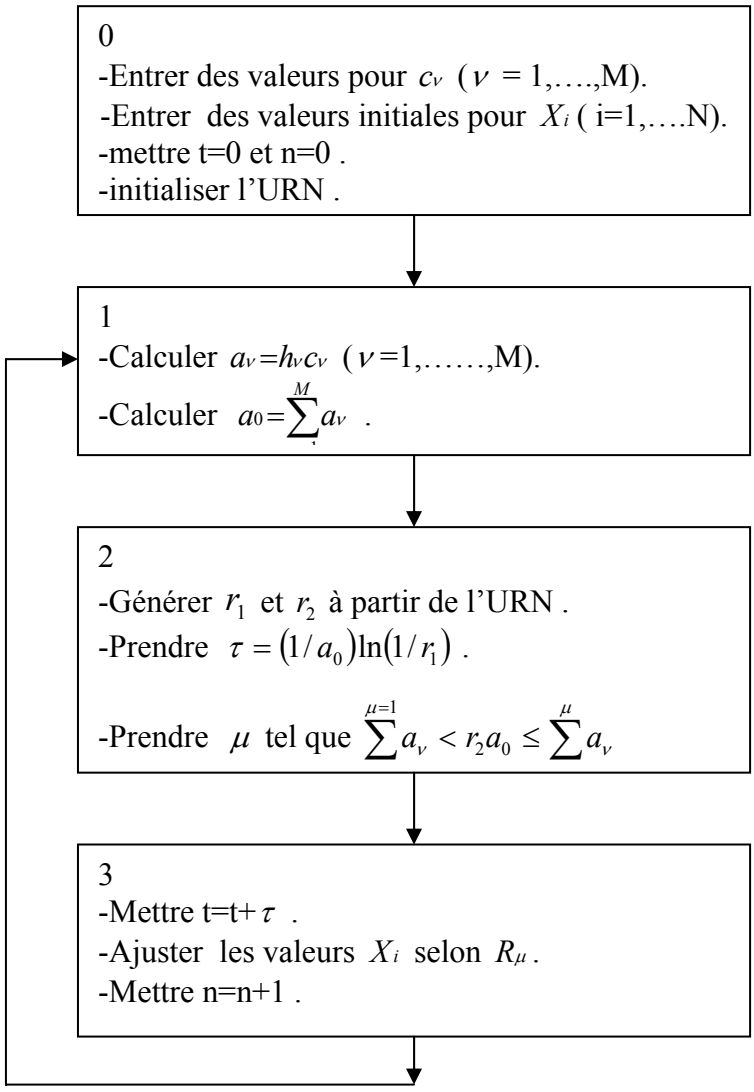

Figure 12 : L'algorithme de la simulation stochastique

5.2 Le π-calcul stochastique biochimique

La sémantique originale du π -calcul stochastique nécessite quelques modifications dans le but de décrire correctement les réactions chimiques. Le π-calcul stochastique biochimique est inspiré de l'algorithme de Gillespie. Il est défini par :

1- Canaux avec taux de base comme réactions élémentaires avec des taux mesoscopiques constants

On associe à chaque canal une constante stochastique, appelé le taux de base. Ce taux de base est identique au taux mesoscopique constant de la réaction élémentaire correspondante.

2- Les taux réels des canaux comme taux réels des réactions

A chaque état du système de π-calcul nous déterminons à un instant donné le taux réel d'un canal en se basant sur le taux de base de ce canal et le nombre d'entrée et de sortie trouvé sur ce canal à cet état (lesquels représentent le nombre de molécule du réactif correspondant).
Le taux réel d'un canal est estimé au taux réel de la réaction correspondante.
On distingue deux types de réactions :
1) Une réaction où on a deux molécules différentes de réactifs. Soient |P| et |Q| le nombre de molécules de type P et Q respectivement .Alors, le taux réel de la réaction est donné par :

Le taux de base*|P|*|Q|

Où le taux de base est le taux de base du canal où s'effectue la réaction.
2) Une réaction où on a deux molécules de réactifs et qui sont identiques, réagissant entre eux, ce type de réaction est appelée homodimerization. Le taux réel de la réaction est donné par :

$$\frac{1}{2} \text{*le taux de base*|Q|*(|Q| -1)}$$

Où le taux de base est le taux de base du canal où s'effectue la réaction.

3- Le pas du temps du π -calcul comme évolution du temps de système chimique

Le pas de temps est déterminé d'après l'algorithme de Gillespie décrit -ci-dessous.

4- Sélection stochastique de communication d'après la probabilité d'une réaction

La sélection stochastique de communication est identique à celle spécifiée dans l'algorithme de Gillespie.

5.3 BioSpi

Biospi [67] est un simulateur d'un programme codé en π-calcul stochastique en utilisant l'algorithme de Gillespie. Il a été Développé par Regev et Shapiro. Biospi est basé sur le système logix [76] et est implémenté en FCP (Flat Concurrency Prolog) [75]. Plusieurs modèles ont été simulés en utilisant cet utilitaire [68] [43] [41] [42] [66].

Biospi reçoit comme entrée un programme codé en π-calcul, ensuite l'exécute.
Biospi a plusieurs versions : dans Biospi 1.0, la selection de communication est non déterministe, dans Biospi 2.0 l'algorithme de Gillespie est implémenté pour la simulation d'un programme codé en π-calcul stochastique.

5.4 SPiM (the Stochastique Pi-Machine)

SPiM [60] [61] [9] est aussi un simulateur d'un programme codé en π-calcul stochastique utilisant l'algorithme de Gillespie. Il a été implémenté par andrew Philips et luca Cardelli, c'est une description formelle (machine abstraite) de la façon dont les processus de π-calcul stochastique peuvent être exécutés. Cela n'est pas attaché à une plate-forme particulière, il a une implémentation propre à lui.

La machine abstraite est prouvée correcte et complète. Une présentation graphique pour le π-calcul a été aussi définie [59].

Le simulateur a aussi été utilisé pour simuler une variété de systèmes chimiques et biologiques[7]. Dans quelque cas le simulateur SPiM s'exécute au-dessus de 7 fois plus rapide que Biospi.

6 Conclusion

Dans ce chapitre, nous avons présenté la modélisation des processus biologiques par le π-calcul stochastique. Nous avons présenté aussi l'algorithme de Gillespie utilisé pour la simulation d'un programme codé par le π-calcul stochastique. Nous avons terminé notre chapitre par la description de deux simulateurs (BioSpi et SPiM) implémentés pour la simulation d'un programme codé par le π-calcul stochastique.

Dans le chapitre suivant nous allons présenter un exemple d'un processus biologique qui a été modélisé en utilisant le π-calcul stochastique.

[7] Détails des résultats de simulation sont disponibles à partir : http://www.doc.ic.ac.uk/~anp/spim/

Chapitre 6

Etude d'un cas :

Régulation de la transcription de gènes par rétroaction

1 Introduction

Nous présenterons ici un processus biologique c'est la régulation de la transcription de gènes par rétroaction étudié dans [65] [12]. Ce processus a été modélisé par le π -calcul stochastique.

Nous commencerons par la description biologique de ce processus, puis nous présenterons la modélisation de ce processus biologique en π-calcul stochastique. Enfin nous donnerons la simulation de cette modélisation en utilisant le simulateur SPiM.

2 Description biologique

Nous présentons ici un exemple d'un système biomoléculaire : « la régulation de la transcription de gène par rétroaction positive ».

Le système biomoléculaire contient deux brins d'ADN sur lesquels sont codés deux gènes (le gène A et le gène TF). Ces deux gènes sont transcrits en ARN messager (ARN_m de A et ARN_m de TF). Les deux brins d'ARN_m sont traduits en protéines ou détruits. Les deux protéines formées (protéine de A et protéine de TF) vont soit se dégrader soit interagir l'un avec l'autre. La protéine A est constituée de deux parties : partie Kinase et un Site d'attache qui va s'unir avec la protéine TF. Quand le Site d'attache de protéine A est lié avec la protéine TF la partie Kinase de protéine A va alors activer la protéine TF puis le complexe protéine A-protéine TF va désassocier. La protéine TF activée et libérée va pouvoir catalyser la transcription des gènes A et TF. La figure 13 décrit ce système biomoléculaire.

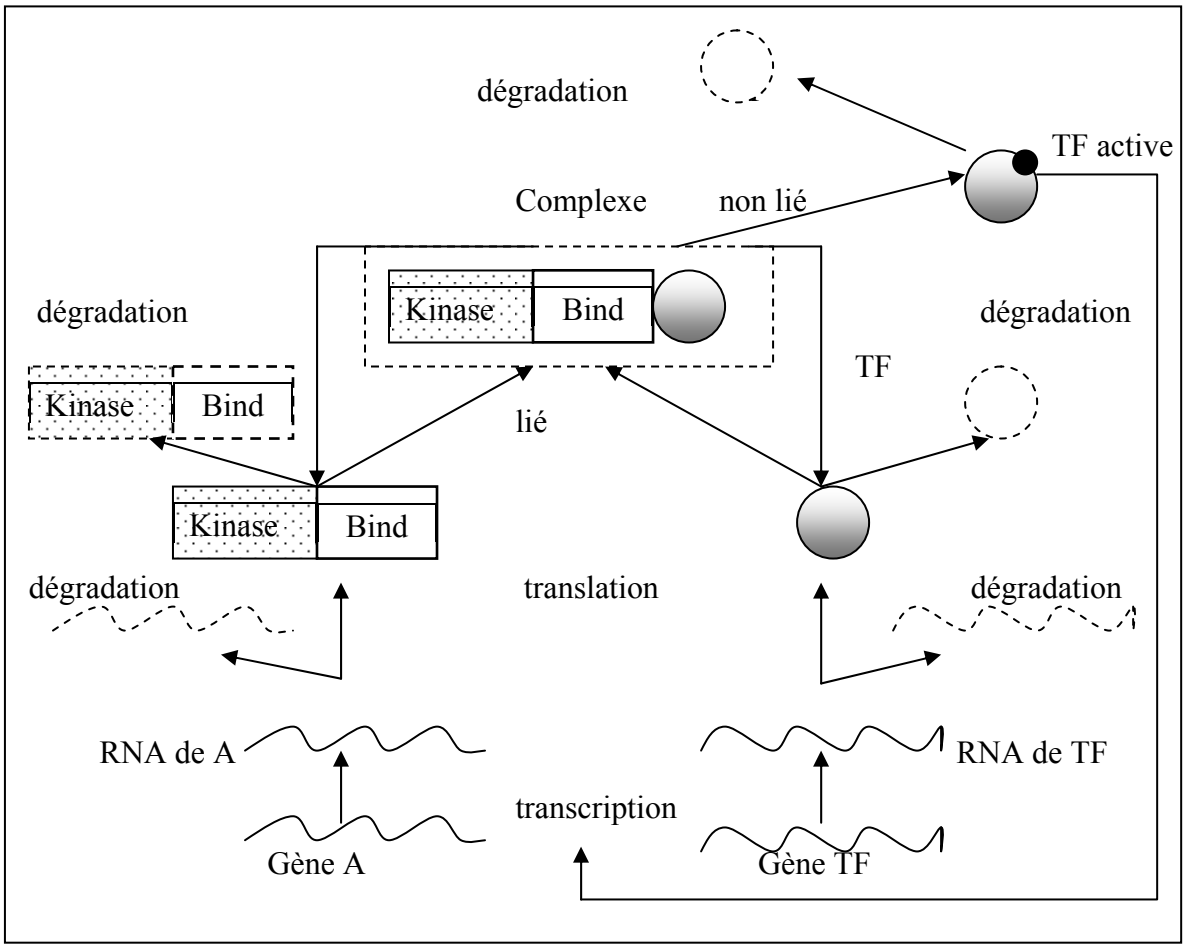

Figure 13 : Régulation de la transcription de gènes par rétroaction

3 Spécification en π-calcul stochastique

Ce code présente la modélisation de ce système biomoléculaire en π -calcul stochastique.

Sys = Gene_A | Gene_TF | Transcr | Transl | RNA_Deg | Protein_Deg……..(1)

Gene_A = (basal(), 4).(Gene_A | RNA_A)……...(2)

+ (pA(), 40).(Gene_A | RNAA)……....…(3)

RNA_A = (utr(), 1).(RNA_A | Protein_A)……...(4)

+ (degm(), 1)……..…(5)

Protein_A = (vbb1), (vbb2), (vbb3)(Binding_Site | Kinase)……...(6)

Binding_Site = ($\overline{bind}$ (bb1, bb2, bb3), 0.1).Bound_Site……...……(7)

+ (degp(), 0.1).($\overline{bb3}$,∞)…...……...…(8)

Bound_Site = ($\overline{bb1}$, 10).Binding_Site……...(9)

+ (degp(), 0.1).($\overline{bb3}$,∞).($\overline{bb3}$,∞)…….……...(10)

Kinase = ($\overline{bb2}$ (ptail), 10).Kinase……(11)

+ (bb3(),∞)……..……(12)

Gene_TF = (basal(), 4).(Gene_TF | RNA_TF)……(13)

+ (pA(), 40).(Gene_TF | RNA_TF)……(14)

RNA_TF = (utr(), 1).(RNA_TF | Protein_TF)……(15)

+ (degm(), 1)…….....(16)

Protein_TF = (bind(c_bb1, c_bb2, c_bb3), 0.1).Bound_TF……(17)

+ (degp(), 0.1)……..(18)

Bound_TF = (c_bb1(), 10).Protein_TF……(19)

+ (c_bb3(),∞)……(20)

+ (c_bb2(tail), 10).((c_bb1(), 10).Active_TF(tail)……...(21)

+ (c_bb3(),∞))……..(22)

Active_TF(tail) = ($\overline{tail}$, 100).Active_TF(tail)……(23)

+ (degp(), 0.1)…….....(24)

Transcr = ($\overline{basal}$, 4).Transcr……(25)

+ (ptail(), 100).($\overline{pA}$, 40).Transcr……...(26)

Transl = ($\overline{utr}$, 1).Transl……..(27)

RNA_Deg = ($\overline{\deg m}$, 1).RNA_Deg……(28)

Protein_Deg = ($\overline{\deg p}$, 0.1).Protein_Deg……...(29)

4 Interprétation

- Le système biomoléculaire composé de cinq processus (*Gene_A*, *Gene_TF*, *Transcr*, *Transl*, *RNA_Deg* et *Protein_Deg*) qui s'exécutent en parallèle (1) .
- Au départ, le processus *Transcr* est le premier à exécuter, il envoie des alértes par le canal *basal* (25). Ces alèrtes sont reçus soit par le processus *Gene_A* (2) soit par le processus *Gene_TF* (13) à travers leur canal *basal*. Ces processus créent le RNA de gène correspondant (*RNA_A* par le *Gene_A* (2) et *RNA_TF* par le *Gene_TF* (13)) . Les processus *RNA_A* et *RNA_TF* peuvent créer les protéines *Protein_A* (4) et *Protein_TF* (15) respectivement par la réception d'une alèrte sur le canal *utr* généré par le processus *Transl* (27) ou bien peuvent être dégradé (5) (16) par la réception d'alèrte sur le canal *degm* envoyé par le processus *RNA_Deg* (28).
- Le processus *Protein_A* crée trois canaux privés : *bb1*, *bb2* et *bb3* réservés à ses deux processus constitutifs : le site d'attache (*Binding_Site*) et la kinase (*Kinase*) (6).
- Le processus *Binding_Site* indique que la *Protein_A* n'est pas attachée avec la *Protein_TF* . Il peut se comporter comme un processus *Bond_Site* qui indique que le processus *Protein_A* est lié avec le processus *Protein_TF* (7) en utilisant le principe de scope extrusion : il envoie ses canaux privés (*bb1*, *bb2* et *bb3*) à travers leur canal *bind* (7) qui sont reçus par le processus *Protein_TF* (17), ou bien il peut se dégrader en recevant des alèrtes à travers le canal *degp* (8) envoyé par *Protein_Deg* (29). Puis il envoie un signal instantané à travers *bb3* au *Kinase* pour qu'il se dégrade aussi (8).
- Le processus *Bond_Site* peut soit émettre un stimuli sur bb1 pour signaler que l'attachement des deux protéines est rompu et il se comportera alors comme *binding_Site* (9) , ou reçevoir une alèrte des processus de dégradation et transmet l'information à *Kinase* et à *Bond_TF* avant de s'éteindre (10).
- Le processus *Kinase* peut soit envoyer le canal *ptail* à travers le canal *bb2* et se comporter ensuite comme *Kinase* (11), soit se dégrader (12).
- Le processus *Protein_TF* après avoir reçu par le biais du canal *bind,* les noms de canaux privés de *Protein_A* , se comporte comme le processus *Bond_TF* (17) , ou bien il peut se dégrader en recevant d'alèrte à travers *degp* (18) envoyé par le processus *Protein_Deg* (29).

- Le processus *Bond_TF* est composé d'un choix d'actions. S'il reçoit un alèrte à travers *bb1* ce qui signifie qu'il y a détachement entre les deux protéines il va redevenir *Protein_TF* (19). s'il reçoit d'alèrte à travers *bb3* alors il se dégrade (20). Enfin, s'il reçoit le canal *ptail* à travers son canal *bb2* envoyé par le processus *Kinase* suivi d'un alèrte sur leur canal *bb1* qui signifie qu'il y a détachement entre les deux protéines alors il peut se comporter comme *Active_TF (ptail)* (21) ou se dégrade s'il reçoit un alèrte à travers *bb3* (22).
- Le processus *Active_TF (ptai)* envoie des alèrtes à travers *ptail* (23) ou se dégrade (24).
- Le processus *Transcr* reçoit l'alèrte envoyé par le processus *Active_TF* à travers son canal *ptail*. Il envoie ensuite d'alèrte sur son canal *pA* (26). Ces alèrtes sont reçus soit par le processus *Gene_A* soit par le processus *Gene_TF* qui à son tour accélère la transcription de ces gènes.

5 Transformation en SPiM

Voici l'interprétation du code précédent en SPiM :

Remarque :

Le texte entre accolades sert aux commentaires et ne fait pas partie du code en SPiM.

directive sample 50.0 1000 *{50 représente la durée de temps de la simulation et 1000 désigne l'intervalle de temps minimal entre les étapes de simulations qui est égal à la durée de temps de simulation divisé par cette valeur , c-à-d 50/1000 est égal à 0.05}*

directive plot Binding_Site() *{ tracer les valeurs de la quantité de processus Binding_Site pendant toutes les étapes de simulations }*

new basal@4.0 : chan *{ declaration d'un canal dont le nom est basal et sa valeur stochastique est égale à 4 et de type chan }*

new utr@1.0 : chan *{ declaration d'un canal dont le nom est utr et sa valeur stochastique est égale à 1 et de type chan }*

new pA@40.0 : chan *{ declaration d'un canal dont le nom est pA et sa valeur stochastique est égale à 40 et de type chan }*

new degm@1.0 : chan *{ declaration d'un canal dont le nom est degm et sa valeur stochastique est égale à 1 et de type chan }*

new bind@0.1 : chan(chan,chan(chan),chan) *{ declaration d'un canal dont le nom est bind et sa valeur stochastique est égale à 0.1 et de type chan(chan,chan(chan),chan) }*

new degp@0.1 : chan *{ declaration d'un canal dont le nom est degp et sa valeur stochastique est égale à 0.1 et de type chan }*

new tail@100.0 : chan *{ declaration d'un canal dont le nom est tail et sa valeur stochastique est égale à 100 et de type chan }*

new ptail@100.0 : chan *{ declaration d'un canal dont le nom est ptail et sa valeur stochastique est égale à 100 et de type chan }*

let Gene_A() = *{ definition de processus Gene_A qui ne reçoit pas de paramètres }*

(do ?basal ; (Gene_A() | RNA_A())

or ?pA ; (Gene_A() | RNA_A()))

and

RNA_A() = *{ definition de processus Gene_A qui ne reçoit pas de paramètres }*

(do ?utr ; (RNA_A() | Protein_A())

or ?degm)

and

Protein_A() = *{ definition de processus Gene_A qui ne possède pas de paramètres }*

(new bb1@10.0 : chan

new bb2@10.0 : chan(chan)

new bb3: chan

run (Binding_Site(bb1,bb2,bb3) | Kinase(bb2,bb3))) *{ execution des processus Binding_Site et Kinase en paralèlle }*

and

Binding_Site(bb1:chan,bb2:chan(chan),bb3:chan) = *{ definition de processus Binding_Site qui possède les canaux bb1, bb2 et bb3 comme paramètres }*

do !bind(bb1,bb2,bb3) ; bound_Site(bb1,bb2,bb3)

or ?degp; !bb3

and

bound_Site(bb1:chan,bb2:chan(chan),bb3:chan) = *{ definition de processus Bound_Site qui possède les canaux bb1, bb2 et bb3 comme paramètres }*

(do !bb1 ; Binding_Site(bb1,bb2,bb3)

or ?degp ; !bb3 ; !bb3)

and

Kinase(bb2:chan(chan),bb3:chan) = *{ definition de processus Kinase qui possède les canaux bb2 et bb3 comme de paramètres }*

```
(     do !bb2(ptail) ; Kinase(bb2,bb3)
      or ?bb3   )
let
  Gene_TF() =         { definition de processus Gene_TF qui ne possède pas de paramètres }

(     do ?basal ; ( Gene_TF() | RNA_TF() )
      or ?pA ; ( Gene_TF() | RNA_TF() )   )
and
      RNA_TF() =        { definition de processus  RNA_TF qui ne possède pas de paramètres }
(     do ?utr ;( RNA_TF() | Protein_TF() )
      or ?degm   )
and
 Protein_TF () =      { definition de processus  Protein_TF qui ne possède pas de paramètres }
(     do ?bind(c_bb1,c_bb2,c_bb3) ; bound_TF(c_bb1,c_bb2,c_bb3)
      or ?degp    )
 and
 bound_TF (c_bb1:chan,c_bb2:chan(chan),c_bb3:chan) = { definition de processus
            Bound_TF  qui possède les canaux c_bb1, c_bb2 et c_bb3 comme  paramètres }
(     do ?c_bb1 ; Protein_TF()
      or ?c_bb3
      or ?c_bb2(tail) ; do ?c_bb1 ; Active_TF(tail)
                  or ?c_bb3    )
 and
Active_TF(tail:chan) = { definition de processus Active_TF qui possède le canal ptail  comme
                                                                          paramètre}
(     do !tail ; Active_TF(tail)
      or ?degp    )
let
Transcr() =                 { definition de processus  Transcr qui ne possède pas de paramètres }
(     do !basal ; Transcr()
      or ?ptail ; !pA ; Transcr ()   )
let
Transl() =                  { definition de processus  Transl qui ne possède pas de paramètres }
(    !utr ; Transl()   )
```

let

RNA_Deg() = *{ definition de processus RNA_Deg qui ne possède pas de paramètres }*

(!degm ; RNA_Deg())

let

Protein_Deg() = *{ definition de processus Protein_Deg qui ne possède pas de paramètres }*

(!degp ; Protein_Deg())

run (Gene_A() | Gene_TF() | Transcr() | Transl() | RNA_Deg() | Protein_Deg())

{ exécution de ces processus en parallèle}

6 Simulation en SPiM

Nous avons simulé le programme précèdent en SPiM. Nous avons calculé la quantité de processus *Binding_Site* de processus *Protein_A* pour une période de 50, nous avons obtenu un graphe croissant jusqu'à atteindre la valeur 431 (illustré dans la figure 14).

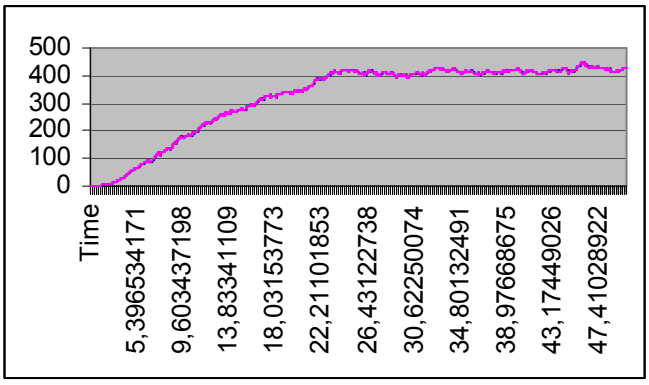

Figure 14 : La quantité de Binding_Site dans la présence de gène TF

Puis, nous avons modifié le programme précédent, nous avons supprimé le gène TF du programme et nous avons fait la simulation pour la même période. Nous avons obtenu une valeur de processus *Binding_Site* qui ne dépasse pas 36. La figure 15 montre les résultats de simulation.

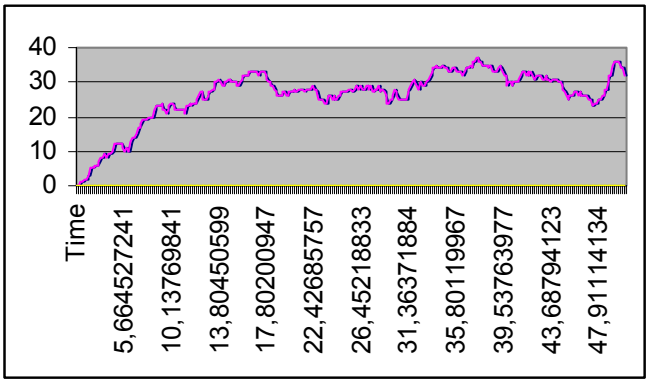

Figure 15 : La quantité de Binding_Site dans l'absence de gène TF

7 Conclusion

Dans ce chapitre, nous avons présenté la modélisation d'un processus biologique, le cas de la régulation de la transcription des gènes par rétroaction positive en utilisant le π-calcul stochastique.

Nous avons présenté aussi la simulation de cette modélisation en modifiant certaines conditions (présence et absence de gène TF). Ce qui nous a permis de déduire certaines propriétés de ce processus biologique.

Conclusion générale

Cette étude nous a permis de présenter la modélisation et la simulation des processus biologiques par un langage formel qui est le π-calcul. Ce dernier a été défini pour analyser et décrire les systèmes parallèles.

L'utilisation d'un langage formel pour la modélisation des processus biologiques offre des moyens rigoureux pour étudier et comparer les processus biomoléculaires. Et permet également la vérification de certaines propriétés du système modélisé.

Ceci rend l'application du π-calcul dans la modélisation et la simulation des processus biologiques très importante : il donne la possibilité de simulation du système modélisé (BoiSpi et SPiM), et permet le raisonnement quantitatif et qualitatif sur les propriétés du système, le modèle est aussi extensible, pertinent et compréhensible permettant ainsi une représentation claire du monde moléculaire. On précise que le degré de pertinence est plus élevé si on utilise l'approche mobile. Malgré ses avantages, le modèle n'échappe pas à quelques limitations de compréhensibilité (utilisation des canaux privés pour la représentation des compartiments, la communication ne peut être effectuée qu'entre deux processus et la spécification ne peut etre que textuelle).

Le π-calcul possède l'avantage de pouvoir être étendu grâce à sa théorie riche. Le passage en π-calcul stochastique est un cas de cette extension. Il est aussi intéressant de développer des outils pour dessiner des modèles graphiques, qui peuvent automatiquement générer le code en π-calcul correspondant. L'utilisation des outils graphiques permet aux non informaticiens de contribuer dans l'activité de recherche et par la suite des nouvelles propositions seront fournies. Il est aussi intéressant d'évaluer le π-calcul avec les autres modèles existants (par exemple les réseaux de Petri).

Chacun des modèles proposés dans le cadre de la modélisation formelle des processus biologiques a ses avantages et ses inconvénients. Aucun n'est parfait et ne peut être considéré comme une solution finale, mais le π-calcul reste l'un des meilleurs modèles. La recherche à maîtriser et à comprendre la différence entre ces modèles est une tâche primordiale permettant de proposer d'autres solutions et pourquoi pas combiner entre ces modèles pour arriver à un langage unifié des processus biologiques.

Bibliographie

[1] **R. Alves and M. A. Savageau.** *Extending the method of mathematically controlled comparison to include numerical comparisons*. Bioinformatics, 16(9):786-798, 2000.

[2] **R. Amadio, I. Castellani, and D. Sangiorgi**. *On bisimulations for the asynchronous pi-calculus*. In Proc. CONCUR '96, volume 1119 of Lecture Notes in Computer Science. Springer Verlag, 1996.

[3] **A. Arkin, J. Ross, and H.H. McAdams**. *Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells.*Genetics, 149:1633-1648, 1998.

[4] **G. D. Bader, I. Donaldson, C. Wolting, B. F. Ouellette, T. Pawson, and C. W.Hogue.** *Bind-the biomolecular interaction network database*. Nucleic Acids Research, 29(1):242-245, 2001.

[5] **J.C.M. Baeten.** *A Brief History of Process Algebra*.Report CSR, 04-02. Eindhoven, Netherlands: Vakgroep Informatica, Technische Universiteit Eindhoven, 2004 .

[6] **Jan A. Bergstra and Jan Willem Klop.** *Process algebra for synchronous communication*. Information and Computation, 60:109-137, 1984.

[7] **Fredrick B. Beste**. *Model Prover - a sequent-calculus based modal mu-calculus model checker tool for finite control pi-calculus agents.* MastersThesis, Department of Computer Systems, Uppsala University, 1998.

[8] **U. S. Bhalla and R. Iyengar.** *Emergent properties of networks of biological signaling pathways.* Science, 283(5400):381-387, 1999.

[9] **Ralf Blossey, Luca Cardelli and Andrew Phillips.** *A Compositional Approach to the Stochastic Dynamics of Gene Networks*. Transactions in Computational Systems Biology (TCSB), 3939:99–122, January 2006.

[10] **M. Boreale , G.Ferro , G.Ristori.** *A model checking tool for the pi-calculus based on pi-automata.* IEI . C.N.R .Tech. Rep B4-52-12-95

[11] **G. Boudol.** *Asynchrony and the pi-calculus.* Technical Report 1702, INRIA, Sophia Antipolis, 1992.

[12] **Mikael Bourhis.** *Algèbres de processus et modélisation de phénomènes biologiques.* Master Recherche Informatique.Cerv, IFSIC Promotion 2004-2005 .

[13] **Vincenzo Ciancia.** *A temporal logic for HD-automata.*Tesi di laurea specialistica. Università degli studi di pisa. Facoltà di scienze matematiche, fisiche e naturali. Corso di laurea specialistica in informatica. Anno accademico 2002-2003.

[14] **G. Ciobanu and M. Rotaru.** *Molecular interaction*. Journal of Theoretical Computer Science, 289(1):801{827, 2002.

[15] **E.M. Clarke, E.A. Emerson, and A.P. Sistla.** *Automatic verification of finite-state concurrent systems using temporal logic specifications*. ACM Transactions on Programming Languages and Systems, 8(2):244–263, 1986.

[16] **R. De Nicola, A. Fantechi, S. Gnesi, G. Ristori.** *An action based framework for verifying logical and behavioural properties of concurrent systems.*Computer Networks and ISDN Systems, Vol. 25, N. 7, pp. 761-778, North Holland, Febbraio1993.

[17] **Rocco De Nicola, Alessandro Fantechi, Stefania Gnesi, Salvatore Larosa, Gioia Ristori.** *Verifying hardware components within JACK.* CHARME 1995: 246-260.

[18] **E.A. Emerson and J.Y. Halpern.** *"Sometimes" and "Not Never" Revisited: on Branching Time versus Linear Time Temporal Logic.* Journal of ACM, 33(1), January 1986, pp. 151-178.

[19] **J.Etienne.** *Biochemie génétique biologie moléculaire*. masson, paris, 1987,1999.

[20] **A. Fantechi and S. Gnesi.** *Action-based Model Checking (and its applications to distributed, mobile, object-oriented systems)*. published as Technical Report 2003-TR-58, Consiglio Nazionale delle Ricerche, Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', 2003

[21] **Fantechi, S. Gnesi, F. Mazzanti, R. Pugliese, E. Tronci.** *A Symbolic Model Checker for ACTL*. Applied Formal Methods -- FMTrends 98, LNCS 1641, Springer - Verlag, 1999.

[22] **Jérome Feret.** *Informal introduction to mobility*. Cours de DEA. Ecole normale supérieure. February, 2004.

[23] **G. Ferrari, G. Ferro, S. Gnesi, U. Montanari, M. Pistore, and G. Ristori.** *An automata based verification environment for mobile processes*. In Proc. TACAS'97, volume 1217 of LNCS. Springer Verlag, 1997.

[24] **G. Ferrari, S. Gnesi, U. Montanari, and M. Pistore.** *A model checking verification environment for mobile processes.* ACM Transactions on Software Engineering and Methodologies (TOSEM), 2004.

[25] **G. Ferrari, S. Gnesi, U. Montanari, M. Pistore, G. Ristori.** *Verifying Mobile Processes in the HAL Environment*. CAV'98, LNCS 1427, Springer - Verlag, 1998.

[26] **Ferrari G., Gnesi S., Montanari U., Raggi R., Trentanni G., Tuosto E.** *Verification on the Web of mobile systems*. 2nd International Workshop on Verification and Validation of Enterprise Information Systems (VVEIS 2004) (Porto, Portugal 14-17, April 2004). Proceedings, p. 72-74. Juan Carlos Augusto and Ulrich Ultes-Nitsche (Eds.). INSTICC Press, 2004.

[27] **Giovanni Ferro**. *Un model checker linear par la logica temporale ACTL*. Tesi di laurea . Università degli studi di pisa. Facoltà di scienze matematiche, fisiche e naturali. Corso di laurea in scienze dell'informzione. Anno accademico 1993/1994.

[28] **G. Ferro.** *AMC: ACTL Model Checker.* Reference Manual. IEI-Internal Report, B4-47, 1994.

[29] **A. Finney, H. Sauro, M. Hucka, and H. Bolouri.** *An xml-based model description language for systems biology simulations.* Technical report, California Institute of Technology, September 2000. Technical report.

[30] **W. Fontana and L. W. Buss.** *The barrier of objects: From dynamical systems to bounded organizations.* In J. Casti and A. Karlqvist, editors, Boundaries and Barriers, pages 56-116. Addison-Wesley, 1996.

[31] **D.Freifelder**. *Biologie moleculaire*. masson , 1990 .

[32] **L. Genevès**. *Biologie moléculaire*. dunod, Paris, 1996.

[33] **Daniel T. Gillespie**. *Exact stochastic simulation of coupled chemical reactions*. The Journal of Physical Chemistry, 81(25) :2340–2361, 1977.

[34] **S. Gnesi, G. Ristori.** *A model checking Algorithm for pi-calculus agents*. In proc. Second Internal Conference on Temporal logic (ICTL)Kluwer Academic Publishers, 1997 .

[35] **S. Gnesi and G. Ristori.** *A Model Checking Algorithm for pi-calculus agents.* In Advances in Temporal Logic, H. Barringer, M. Fisher, D.Gabbay, G. Gough eds,Applied Logic Series, Vol. 16 , Kluwer Academic Publishers, pp.339-358, 2000.

[36] **P. J. E. Goss and J. Peccoud.** *Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets.* Proceedings of the National Academy of Sciences USA, 95(12):6750-6755, 1998.

[37] **A. Harlay**. Biologie. Masson, Paris, 1999.

[38] **C. A. R. Hoare.** *Communicating Sequential Processes*. Prentice Hall, 1985.

[39] **K. Honda et M.Tokoro**. *An object calculus for asynchronous communication*. Proc. ECOOP 91, Geneve, 1991.

[40] **N. Kam, D. Harel, and I.R. Cohen.** *Modeling biological reactivity: Statecharts vs. boolean logic. In Proceeding of the Second International Conference on Systems Biology*. Pasadena, CA, 2001.

[41] **Céline Kuttler and Joachim Niehren.** *Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch.* Transactions on Computational Systems Biology, 2005. Accepted for publication. A first version was published at the second international workshop on concurrent models in molecular biology (BioConcur2004).

[42] **Celine Kuttler, Joachim Niehren, and Ralf Blossey.** *Gene regulation in the pi-calculus: Simulating cooperativity at the lambda switch.* In Workshop on Concurrent Models in Molecular Biology, ENTCS. Elsevier, 2004.

[43] **Lecca, P., Priami, C**. *Cell cycle control in eukaryotes: a biospi model.* In: BioConcur' 03, ENTCS (2003)

[44] **Cédric Lhousaine.** *Réceptivité, Mobilité et pi-Calcul*.These de doctorat, Université D'AIX-MARSEILLE I, le 28 juin 2002.

[45] **Brahim Mammass.** *Une preuve formelle du bounded retransmission protocol dans le pi-calcul*. Rapport de recherche 1998-009. LIP6, équipe SPI Université Pierre et Marie Curie,1998 .

[46] **H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano.** *Hybrid petri net representation of gene regulatory network.* In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, Pacific Symposium on Biocomputing, volume 5, pages 341-352, Singapore, 2000. World Scientific Press.

[47] **Robert Meolic, Tatjana Kapus, Zmago Brezocnik.** *Verification of concurrent systems using ACTL.* In Applied informatics: proceedings of the IASTED international conference AI'2000, M. H. Hamza, ed., Anaheim, Calgary, Zürich, IASTED/ACTA Press, Innsbruck, Austria, pages 663-669, February 14-17, 2000.

[48] **Robin Milner.** *A calculus of communicating systems.* Volume 92 of Lecture Notes in Computer Science. Springer, 1980.

[49] **Robin Milner.** *The Polyadic pi-calculus: a Tutorial*. Laboratory for Fondations of Computer Science, Computer Science Département, University of Edinburgh, October 1991.

[50] **R. Milner.** *Functions as processes*. Journal of Mathematical Structures in Computer Science, 2(2):119-141, 1992.

[51] **R. Milner, J. Parrow and D . Walker.** *A calculus of mobile processes*, part I/II. Journal of Information and Computation, 100:1-77, Sept. 1992.

[52] **R. Milner, J. Parrow, and D. Walker.** *Modal logics for mobile processes.*Theoretical Computer Science, 114(1):149–171, 1993.

[53] **Robin Milner and Davide Sangiorgi.** *Barbed bisimulation.*In Proceedings of 19th ICALP. LNCS 623, pages 685-695, 1992.

[54] **Montanari U., Pistore M**. *History dependent automata*. Technical Report # 0112−14. Istituto Trentino di Cultura. December 2001

[55] **U. Montanari and M. Pistore.** *Checking bisimilarity for finitary pi-calculus.*In Proc. CONCUR'95, LNCS 962. Springer Verlag, 1995.

[56] U. Montanari and M. Pistore. History-dependent automata. Technical Report TR-98-11, Dipartimento di Informatica, Pisa. 1998

[57] **U. Montanari and M. Pistore.** *An introduction to history dependent automata.* In Proc. Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II), volume 10 of ENTCS. Elsevier, 1998.

[58] **Joachim Parrow**. *An introduction to the pi-calculus*. In Handbook of Process Algebra, ed. Bergstra, Ponse, Smolka, pages 479-543, Elsevier 2001

[59] **Andrew Phillips and Luca Cardelli.** *A Graphical Representation for the Stochastic Pi-calculus.* In Proceedings of Concurrent Models in Molecular Biology (Bioconcur'05), affiliated with CONCUR'05, August 2005, San Francisco.

[60] **Andrew Phillips and Luca Cardelli.** *A correct abstract machine for the stochastic pi-calculus.* In Bioconcur'04. ENTCS, August 2004.

[61] **Andrew Phillips and Luca Cardelli.** *A correct abstract machine for the stochastic pi-calculus.* Transactions on Computational Systems Biology, 2005. to appear.

[62] **B.C.Pierce**. *Foundational calculi for programming languages.* In A .B Tucker, editor, Handbook of computer Science and Engineering, chapter 139 .CRC Press, 1996.

[63] **M. Pistore.** *History Dependent Automata*. PhD thesis, Università di Pisa, Dipartimento di Informatica, 1999.

[64] **Corrado Priami.** *Stochastic pi-calculus*. The Computer Journal, 38(7), 1995.

[65] **C. Priami, A. Regev, E. Shapiro, and W. Silverman**. *Application of a stochastic name-passing calculus to representation and simulation of molecular processes.*Information Processing Letters, 80:25-31, 2001.

[66] **Aviv Regev.** *Representation and simulation of molecular pathways in the stochastic pi-calculus.* In Proceedings of the 2nd workshop on Computation of Biochemical Pathways and Genetic Networks, 2001.

[67] **Aviv Regev and Ehud Shapiro**. *The pi-calculus as an abstraction for biomolecular systems.* In Gabriel Ciobanu and Grzegorz Rozenberg, editors, Modelling in Molecular Biology. Springer, 2004.

[68] **Regev, W. Silverman, E. Shapiro.** *Representing biomolecular processes with computer process algebra: π-calculus programs of signal transduction pathways*. in: Proc. Pacific Symp.of Biocomputing, 2000.

[69] **Aviv Regev, William Silverman, and Ehud Shapiro.** *Representation and simulation of biochemical processes using the pi-calculus process algebra.* In Proceedings of the Pacific Symposium of Biocomputing 2001, volume 6, pages 459–470, 2001.

[70] **M. G. Samsonova and V. N. Serov.** *Network: an interactive interface to the tools for analysis of genetic network structure and dynamics.* In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, Paci_c Symposium on Biocomputing, volume 4, pages 102-111, Singapore, 1999. World Scientific Press.

[71] **Davide Sangiorgi.** *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis CST, Department of Computer Science, University of Edingburgh, 1993.

[72] Davide Sangiorgi. Interpreting Functions as pi-calculus Process: a Tutorial. Technical Report, 3470, INRIA, Aout 1998.

[73] **Davide Sangiorgi**. *A theory of bisimulation for the pi-calculus*. Acta informatica, 33:69-97. 1996. Earlier version published as Report ECS-LFCS-93-270, University of Edinburgh. An extended abstract appeared in the Proceedings of CONCUR '98, LNCS 715.

[74] **P. Sewell**. *Applied Pi - A Brief Tutorial.* Technical Report 498, Computer Laboratory, University of Cambridge 2000.

[75] **E. Shapiro**. *Concurrent Prolog*. A progress report, in: E.Shapiro (Ed.), Concurrent Prolog, Vol. I, MIT Press, Cambridge, MA, 1987, pp. 157–187.

[76] **W. Silverman, M. Hirsch, A. Houri, E. Shapiro**. The Logix System User Manual, Version 1.21, Concurrent Prolog, Vol. II, MIT Press, Cambridge, MA, 1987.

[77] **Sylvain Soliman.** *Pi-calculus and LCC, Space odyssey*. Technical Report RR-4855 INRIA Sophia Antipolis, Juin 2003.

[78] **Colin Stirling**. *Modal and Temporal Logics for Processes*. Banff Higher Order Workshop 1995: 149-237.

[79] **Bernard Swynghedauw**. *Biologie et génétique moléculaires*. Dunod , Paris , 2000 pour la 2ème edition .

[80] **D. Thieffry and D. Romero**. *The modularity of biological regulatory networks.*Biosystems, 50(1):49-59, 1999.

[81] **J. van Helden, A. Naim, R. Mancuso, M. Eldridge, L. Wernisch, D. Gilbert D and S. J. Wodak.** *Representing and analysing molecular and cellular function using the computer.* Biological Chemistry, 381(9-10):921-935, 2000.

[82] **David Walker.** *Objects in the pi-calculus*. Information and Computation, 116(2):253-271, 1995.

[83] **P.C. Winter, G.I.Hickey et H.L.Fletcher.** *L'essentiel en génétique*. Edition berti, 2000.