N° d'ordre : 452

Année 2006

# THÈSE

présentée en vue de

l'obtention du titre de

DOCTEUR

de

L'ÉCOLE NATIONALE SUPÉRIEURE
DE L'AÉRONAUTIQUE ET DE L'ESPACE

ÉCOLE DOCTORALE : INFORMATIQUE ET TÉLÉCOMMUNICATIONS

SPÉCIALITÉ : Programmation et systèmes

par

**Nil GEISWEILLER**

## Étude sur la modélisation et la vérification probabiliste d'architectures de simulations distribuées pour l'évaluation de performances

Soutenue le 22 mars 2006 devant le jury :

| | | | |
|---|---|---|---|
| MM. | P. | SIRON | Président |
| | G. | ALLÉON | |
| | B. | d'AUSBOURG | Directeur de thèse |
| | J.M. | FOURNEAU | Rapporteur |
| | N. | GIAMBIASI | |
| Mme | B. | PLATEAU | Rapporteur |

Thèse préparée au sein du département traitement de l'information et modélisation (DTIM) de l'ONERA-centre de Toulouse.

# THÈSE

présentée en vue de
l'obtention du titre de

DOCTEUR

de

L'ÉCOLE NATIONALE SUPÉRIEURE
DE L'AÉRONAUTIQUE ET DE L'ESPACE

ÉCOLE DOCTORALE INFORMATIQUE ET TÉLÉCOMMUNICATIONS

Spécialité : Programmation et Systèmes

par

Nil GEISWEILLER

## Étude sur la Modélisation et la Vérification Probabiliste d'Architectures de Simulations Distribuées pour l'Évaluation de Performances

Soutenue le 22 mars 2006 devant le jury :

| | | |
|---|---|---|
| MM. | Pierre Siron | Président |
| | Guillaume Alléon | |
| | Bruno d'Ausbourg | Directeur de thèse |
| | Jean-Michel Fourneau | Rapporteur |
| | Norbert Giambiasi | |
| Mmme | Brigitte Plateau | Rapporteur |

Thèse préparée au sein de l'Office National d'Études et de Recherches Aérospatiales

# Remerciements

Merci à Bruno d'Ausbourg, d'abord pour m'avoir offert l'opportunité de faire une thèse et ainsi réaliser un de mes plus grands rêves d'enfance : devenir chercheur. Merci pour ses nombreux conseils, son aide et son soutien tout au long de ma thèse, pour avoir su me recentrer quand j'en avais besoin tout en me laissant une bonne liberté d'exploration.

Merci également aux rapporteurs de ma thèse, Brigitte Plateau et Jean-Michel Fourneau, pour avoir manifesté leur intérêt et pour leurs remarques pertinentes sur mon mémoire.

Merci aux autres membres du jury, Pierre Siron, Norbert Giambiasi et Guillaume Alléon.

Merci à Jane Hillston et Stephen Gilmore pour l'intéret qu'ils ont porté à mes travaux et pour m'offrir cette opportunité de postdoc en leur compagnie.

Merci à Manuel Samuelides pour ses nombreux éclaircissement sur la théorie des probabilités ainsi qu'à ses deux doctorants Éric Juliani et Laurence Cornez.

Merci à Claire Pagetti pour sa relecture détaillée de mon mémoire et à Alexandre Cortier pour sa relecture et ses remarques.

Merci à mon cousin Joachim Reigle pour son aide dans la correction de mes articles en anglais.

Merci à Laurent Sagaspe pour son aide sur latex.

Merci à Christophe Garion pour les mesures de pings qu'il m'a fournies.

Merci à Rémi Delmas et Alexandre Cortier pour leurs discutions rafraichissantes sur la musique et la philosophie.

Merci à ma jeune femme Yana Zaharieva Kamenova pour son très grand soutien ainsi que sa famille Tonya Goranova, Savka Zehlarova et Zahari Petrov.

Merci à ma sœur jumelle Mohini, mon grand frère Satyavan, ma mère Martine, mon père Gilles, mon grand-père René Leclerc et mon arrière grand-père Louis Dornant pour leur soutien à tous.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Les simulations temps réel sont des systèmes informatiques soumis à de fortes contraintes temporelles. En effet, le temps logique de la simulation doit coïncider au mieux au temps réel du système modélisé. En 1996 le département de la défense des USA propose un standard, nommé HLA [20, 21] (pour High Level Architecture), définissant les spécifications d'un ensemble de services intergiciels permettant la coopération et la réutilisation de simulations en une seule simulation distribuée globale et cohérente. Si un tel procédé possède de nombreux avantages il apporte également de nombreuses difficultés concernant sa mise en œuvre, et particulièrement quand il s'agit de simulations temps réel où les contraintes temporelles exigées deviennent difficile à évaluer off-line.

Pour aider le concepteur à mettre en place une simulation distribuée il peut être avantageux de posséder un modèle de cette simulation et de vérifier, à travers ce modèle, la satisfaction des contraintes temporelles exigées. Ce qui permet d'assurer (ou au moins de conforter) le bon fonctionnement de la simulation lors de sa mise en marche réelle. Une simulation distribuée est un système complexe (machines, réseaux, logiciels, intergiciels, etc) difficile à modéliser dans tous ses détails, et si modélisée, trop conséquente pour être utilisable. Il peut être intéressant d'en donner une abstraction probabiliste. Ceci permet de réduire le niveau de détail tout en conservant les informations pertinentes relatives aux contraintes à évaluer.

## 1.2 Objectif

L'objectif posé par la thèse est d'utiliser les récentes techniques de *model checking probabiliste* afin d'opérer cette vérification. Le model checking est un terme générique regroupant un ensemble de méthodes de preuves formelles basées sur l'exploration des

états d'un modèle. Quand elle est utilisable, elle permet de dispenser d'une recherche mathématique spécifique au problème, l'évaluation étant opérée de manière mécanique par des outils de preuves adaptés, les *model checkers*. Utiliser le model checking probabiliste au lieu du model checking non probabiliste permet :

1. d'appliquer ces méthodes sur des modèles probabilistes et ainsi de bénéficier de l'abstraction probabiliste,
2. de formuler les propositions à vérifier dans des logiques probabilistes. Ce qui permet d'exprimer des contraintes de performances plus souples que celles exprimables dans le contexte du model checking traditionnel, comme par exemple : *le système est fiable à 90%.*

Dès lors, cette procédure pose le problème de la modélisation du système et, notamment, les difficultés liées à la modélisation probabiliste. Elle pose également le problème de la formalisation des exigences à vérifier dans une logique acceptée par un model checker probabiliste.

## 1.3 Mise en œuvre

Dans un premier temps, des contraintes de performances on été formalisées et vérifiées à l'aide du model checker PRISM [39, 53] sur un premier modèle de simulation HLA. Les résultats obtenus ont été confrontés avec des données provenant d'une simulation distribuée réelle. Ceci a permis de révéler une difficulté fondamentale de cette démarche : il est difficile de bien choisir les paramètres du modèle (probabilités ou taux de densité de probabilité) pour que les résultats obtenus par le model checker coïncident avec la réalité. Il est difficile de concevoir un modèle réaliste.

Afin de surmonter cette difficulté on a d'abord utilisé des algorithmes d'approximation de données qui ont permis d'obtenir des modèles plus réalistes mais sans structure, les rendant ainsi difficilement compréhensibles et manipulables. Afin de remédier à ce problème, un de ces algorithmes d'approximation, permettant d'induire une distribution de type phase[1] [49, 7, 57] connaissant un ensemble de temps d'absorption, a été adapté afin de lui permettre d'opérer sur des modèles décrits dans l'algèbre stochastique PEPA [33]. L'intérêt d'utiliser l'algèbre stochastique PEPA est au moins double :

1. Cette algèbre permet de décrire un processus probabiliste de manière compositionnelle et de lui donner une structure plus intelligible qu'avec une description énumérative de ses états et de ses transitions.
2. Les modèles PEPA sont reconnus par le model checker probabiliste PRISM.

L'algorithme a ensuite été étendu afin de fonctionner avec des *exécutions partiellement observables* au lieu de temps d'absorption, permettant ainsi de travailler sur des modèles

[1] c'est-à-dire modélisées par des chaînes de Markov [51] avec état absorbant

PEPA avec ou sans état absorbant. Ces deux algorithmes ont été implantés dans le logiciel EMPEPA, un programme Open-source développé dans le cadre des travaux de thèse.

Le *chapitre* 2 introduit en détail la problématique à laquelle s'attachent ces travaux de thèse. Il présente l'architecture HLA et les difficultés liées à la mise en œuvre d'une simulation distribuée, à sa modélisation et à son analyse.

Le *chapitre* 3 contient des rappels sur la théorie des probabilités, les processus de Markov et les distributions de type phases.

Le *chapitre* 4 fait un point sur l'état des recherches en matière de modélisation et de vérification dans le contexte du model checking probabiliste. Il contient des explications détaillées sur le modèle checking probabiliste des processus de Markov, sur l'algèbre stochastique PEPA [33] et sur l'algorithme EM pour l'approximation de distributions de type phase.

Le *chapitre* 5 présente une tentative de modélisation d'une distribution HLA en PEPA avec la formalisation de deux contraintes de performances dans la logique CSL [1]. La première contrainte est une deadline probabiliste sur le temps de transmission des messages entre les composants de la simulation. La deuxième contrainte permet de vérifier que la probabilité de perdre des messages au cours d'une exécution de la simulation ne dépasse pas un certain seuil. Des expérimentations permettant de comparer les résultats fournis par le model checker et les résultats obtenus à partir d'une simulation HLA réelle sont ensuite présentés. Ces comparaisons on permis de mettre l'accent sur les difficultés liées à la modélisation des processus probabilistes.

Le *chapitre* 6 présente une étude menée afin d'améliorer les techniques de modélisation des processus stochastiques. Dans un premier temps l'utilisation d'algorithmes d'approximations montre comment obtenir des modèles réalistes mais manquant de structure et d'intelligibilité. Dans un second temps on présente une adaptation d'un algorithme permettant d'approcher des distributions de type phase pour l'algèbre PEPA avec état absorbant. Puis on présente une extension de cet algorithme pour l'algèbre PEPA sans état absorbant et dont les données d'observation sont des exécutions partiellement observables.

Une conclusion des travaux est donnée au *chapitre* 7.

# Chapter 2

# Contexte et Problèmes

Dans ce chapitre on présente l'architecture HLA et les difficultés associées à la mise en place d'une simulation distribuée, en particulier temps réel. On présente ensuite les difficultés liées à la modélisation d'un tel système dans le contexte de l'évaluation de ses performances par l'approche du model checking probabiliste.

## 2.1 Évaluer les performances d'une simulation distribuée

### 2.1.1 HLA

HLA pour *High Level Architecture* [20, 21] est un standard, développé par le Département de la Défense des États-Unis (DoD), permettant de combiner plusieurs simulations en une simulation globale et cohérente. La simulation globale est appelée une *fédération* et chacune de ses sous-simulations est appelée un *fédéré*. Tous les fédérés sont interconnectés à travers un intergiciel, le RTI pour *Runtime Infrastructure*, chargé de gérer toutes les communications entre les fédérés. Les fédérés peuvent se trouver sur une seule machine ou être répartis sur un réseau local ou même global comme Internet.

L'intérêt de définir un tel standard est de pouvoir réutiliser ensemble des simulateurs développés séparément, fonctionnant sur des calculateurs éventuellement distants, programmés dans des langages ou sur des plateformes éventuellement différentes. Par exemple, si Airbus possède un bon simulateur de vol et la Snecma un bon simulateur de moteur, ils peuvent mettre en commun leurs deux modèles de simulation pour former un simulateur de vol avec un nouveau type de moteur sans être obligés de fusionner leur deux programmes, de porter l'un ou l'autre sur l'une ou l'autre des plateformes (ce qui, en plus de poser des problèmes techniques, peut poser des problèmes de protection du secret industriel). Il peut s'avérer donc plus efficace de définir une interface de communication commune entre leurs deux simulateurs. De plus la distribution des simulations permet de

répartir la charge CPU et donc de gagner du temps de calcul sur la simulation globale.

### 2.1.2 La complexité du système mis en jeu

Une simulation distribuée est un système complexe car constitué d'un certain nombre d'entités complexes de natures différentes :

- du matériel : réseaux, ordinateurs hôtes,
- du logiciel : intergiciel, composants de simulation,
- et parfois même du naturel : humains ou phénomènes physiques présents dans la boucle de simulation.

De multiples problèmes de nature physique ou logique peuvent se produire lors de l'exécution d'une simulation distribuée. Un message peut être perdu ou retardé, par exemple à cause d'une liaison défectueuse, ou bien d'une saturation de la file d'attente d'un ordinateur, ou même encore à cause d'une erreur dans l'algorithme s'interfaçant avec l'intergiciel. Suivant la qualité des composants matériels et logiciels utilisés, des ressources disponibles, et de la manière dont cet ensemble intéragit, une simulation distribuée peut donner des résultats plus ou moins satisfaisants.

### 2.1.3 Temps Logique, Temps Réel

Le temps logique d'une simulation est une grandeur qui modélise le temps qui s'écoule au sein de l'univers simulé. Si pendant le calcul le temps modélisé avance (ou du moins doit avancer) au même rythme que le temps réel alors la simulation est qualifiée de temps réel. Les simulations temps réel sont en général utilisées quand une entité réelle intéragit avec le reste ou une partie de la simulation. Cela peut être par exemple un simulateur de vol pour former un pilote, la simulation d'un objet physique dont certaines données sont fournies par des capteurs soumis à l'environnement extérieur, etc. Une simulation temps réel est assujettie à de très fortes contraintes temporelles. Un retard trop important de telle ou telle donnée peut perturber la bonne marche de la simulation, sa cohérence et son réalisme.

### 2.1.4 La Conception de Simulations Distribuées Temps Réel

Afin d'obtenir une simulation qui réponde à des contraintes de performances temporelles, il est nécessaire de passer par une étape de conception. Cette conception consiste à choisir le bon matériel (réseau, calculateurs), répartir les calculs de la bonne manière. Par exemple, il pourra être préférable de mettre sur la même machine deux fédérés prenant chacun peu de ressources CPU mais ayant de fortes interactions, et mettre sur deux machines distantes deux fédérés prenant chacun beaucoup de ressources CPU mais ayant peu d'interactions. Il convient également de choisir un intergiciel adapté. Doit-il être réactif ou économe en ressource réseau ? Encore une fois la difficulté des choix provient de la complexité engendrée par les multiples interactions et les dépendances qui existent entre

les parties du système. Un réseau réactif peut dispenser d'un intergiciel réactif mais ceci peut aussi dépendre de la quantité de messages échangés sur le réseau qui dépend à son tour de la manière dont les calculs sont répartis.

Il peut être difficile et coûteux de chercher par tâtonnements, à l'aide d'expérimentations successives sur une simulation réelle, une configuration adaptée au bon déroulement de celle-ci, lui permettant de vérifier les contraintes de performances qui lui sont imposées. Pour faciliter et accélérer cette recherche il est judicieux de disposer d'un modèle de la simulation distribuée et d'obtenir par le calcul (et non par les expérimentations) une évaluation de ses performances. La recherche de la meilleure configuration revient à trouver celle qui optimise la performance estimée par ce calcul.

### 2.1.5 L'évaluation de performances : simulation et vérification

On présente dans cette section deux moyens pour faire de l'évaluation de performance de simulations HLA :

1. par simulation, avec l'outil PERFOSIM,
2. par vérification formelle de propriétés exprimant des contraintes de performances.

#### L'outil PERFOSIM : simuler une simulation

PERFOSIM [18] développé à l'ONERA, permet de construire un modèle d'une simulation HLA pour en évaluer les performances. Le modèle est lui-même une fédération HLA dont l'exécution fournit un évaluation des performances de l'architecture de simulation simulée. Il prend en entrée un ensemble de paramètres caractérisant une simulation distribuée donnée. Parmi les paramètres possibles on trouve par exemple :

- le type et le débit du réseau,
- le nombre de messages à échanger entre les fédérés,
- la taille de ces messages,
- la répartition des fédérés sur les différentes machines.

Seul le comportement de la fédération sur les différentes entités (intergiciel, réseau, fédérés) est simulé, les calculs exacts effectués par la simulation simulée ne sont pas pris en compte. PERFOSIM renvoie plusieurs type de résultats, comme par exemple :

- le nombre de cycles réalisés par seconde de la simulation,
- la durée moyenne d'acheminement des messages sur le réseau,
- le nombre total d'appels des services de l'intergiciel,
- la différence qu'il y a pu avoir entre le temps logique et le temps réel de la simulation.

Il devient alors possible, par interprétation des résultats, d'estimer le degré de performances du modèle de la simulation, c'est-à-dire, dans la mesure où le modèle est suffisamment réaliste, le niveau de performance de la simulation réelle.

**Le model checking probabiliste pour l'évaluation de performances**

Une idée à l'origine de cette thèse est d'utiliser les récents outils de model checking probabiliste [1, 5, 4, 2, 40, 30, 6] afin d'évaluer les performances d'une simulation distribuée et ainsi, comme avec l'outil PERFOSIM, d'aider le concepteur de simulations distribuées dans sa tâche. Avant d'expliquer les avantages offerts par le model checking probabiliste nous rappelons brièvement ce qu'est le model checking probabiliste. Des définitions plus approfondies et les détails techniques concernant ces méthodes seront exposés dans la section 4.1.4.

Le model checking caractérise un ensemble de méthodes décidables permettant de vérifier qu'un modèle donné satisfait une formule logique donnée. Pour réussir cela, ces méthodes travaillent sur des logiques et des modèles donc l'expressivité est restreinte. Le model checking est un moyen à la fois riche et souple pour analyser et obtenir des informations sur le comportement général d'un modèle. La simulation, elle, n'offre que des cas particuliers de comportements qu'il faut ensuite analyser pour en déduire une généralité approximative. Le model checking dispense de cette analyse et offre des résultats exhaustifs plus sûrs et plus variés. Il comporte aussi des inconvénients puisqu'il faut faire l'effort de formaliser les contraintes et le modèle et cela, parfois, dans des langages assez peu expressifs alors que les langages de modélisation pour la simulation n'ont généralement pas (ou peu) de limite dans leur expressivité.

Le model checking probabiliste a la particularité de travailler sur des modèles probabilistes et des logiques permettant d'exprimer des comportements probabilistes. Étant donné un modèle, une simulation distribuée par exemple, une logique pour le model checking non probabiliste pourra permettre de formuler et de vérifier une proposition du type : *"Pour toute exécution de la simulation aucun message ne sera perdu"* ou encore son antonyme *"Il existe une exécution telle qu'un message soit perdu"*. Le model checking non probabiliste exprime uniquement des propositions d'ordre qualitatif. Dans certains cas une telle dichotomie -il existe/il n'existe pas- n'est pas vraiment pertinente. Il est par exemple assez difficile d'établir qu'un message ne sera jamais perdu sur un réseau réel. D'autre part cela peut constituer une sur-exigence. Il n'est pas nécessaire d'avoir un réseau totalement fiable. Une modélisation plus adéquate consiste donc à pouvoir quantifier ce degré de perte (notamment par l'utilisation des probabilités). C'est ce que permet de faire le model checking probabiliste. Ainsi il sera aussi possible de vérifier des propositions d'ordre quantitatif, comme par exemple : *"La probabilité de perdre un message est inférieure à* 1%*"*.

## 2.2 Modélisation et model checking

Pour les raisons d'indécidabilité expliquées plus haut, il n'est pas possible de vérifier n'importe quelle formule logique sur n'importe quel processus probabiliste. En général,

si la classe des processus probabilistes considérée est très expressive, alors les logiques décidables sur cette classe sont assez pauvres (voir [2, 11] par exemple). Inversement si une certaine logique probabiliste bénéficie d'une riche expressivité alors les algorithmes de model checking ne pourront opérer que sur des processus de bas niveau.

Les processus probabilistes de plus bas niveau sont les *chaînes de Markov* avec un nombre fini d'états (voir la section 3.2.1 pour la définition de chaîne et de processus de Markov). Une chaîne de Markov à temps discret (CMTD) est définie par l'ensemble de ses états possibles et par les probabilités de passage entre ses états. Une chaîne de Markov n'a pas de mémoire, ce qui traduit le fait que la probabilité de passer d'un état à un autre ne dépend que de l'état courant du processus et pas de ses états antérieurs. Une chaîne de Markov à temps continu (CMTC) capture non seulement les probabilités de transitions entre états mais aussi le temps de séjour dans un état avant de tirer une transition. Cette facilité de représentation du temps de séjour permet de représenter plus facilement l'écoulement du temps au sein du modèle. Pour cette raison on a choisi de travailler sur les chaînes de Markov à temps continu plutôt que discret. Toutefois nous verrons que cette distinction continu/discret n'est pas si évidente puisqu'il existe une correspondance entre les deux (voir section 3.2.4). En outre l'avantage pratique d'utiliser des chaînes de Markov à temps continu est qu'il existe un model checker probabiliste Open-source relativement performant, PRISM [39, 53], qui accepte en entrée des CMTCs. Il existe aussi une algèbre, nommée PEPA [32], permettant de décrire les CMTCs de manière compositionnelle, reconnue par PRISM et disposant d'un ensemble d'outils Open-source permettant de faire des manipulations sur les modèles de cette algèbre.

### 2.2.1 Une contrainte de performance : la deadline probabiliste

La contrainte de performance la plus étudiée au cours de cette thèse répond à une propriété de type deadline probabiliste. Une telle propriété énonce qu'une échéance donnée ne sera satisfaite que dans une certaine proportion de cas, c'est-à-dire d'un point de vue probabiliste avec une certaine probabilité. Voici un exemple de propriété de type deadline probabiliste :

*La probabilité de lire cette phrase en moins*
*de quatre seconde est d'au moins 0.7*

Dans le registre des simulations distribuées temps réel une deadline probabiliste intéressante à vérifier sera par exemple :

*La probabilité que tel composant de simulation*
*communique une mise à jour des ses attributs aux*
*autres composants en moins de 300ms est d'au moins 0.9*

Il existe naturellement d'autres contraintes intéressantes à vérifier, comme par exemple la probabilité de perte de message. Mais pour des raisons techniques expliquées au *chapitre* 5 nous nous sommes principalement focalisé sur des contraintes de type deadline probabiliste.

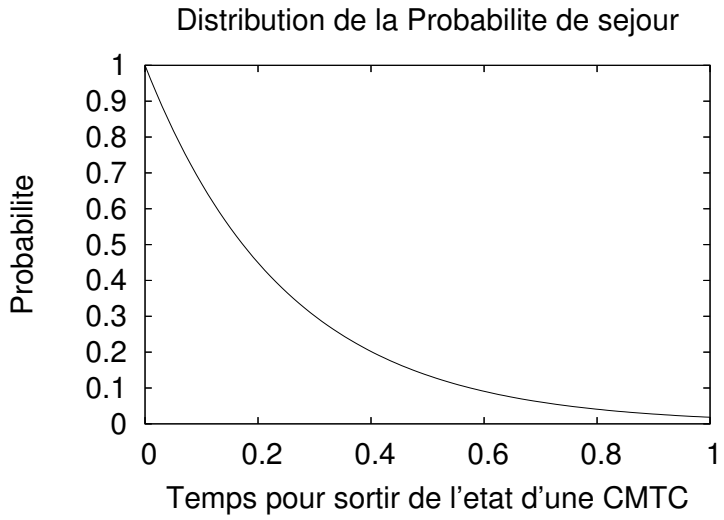



Figure 2.1: *Probabilité de séjour en fonction du temps*

### 2.2.2 Réalisme du modèle suivant l'écoulement du temps

La propriété sans mémoire des chaînes de Markov à temps continu implique que le temps de séjour dans un état est distribué suivant une loi exponentielle décroissante. En effet, la loi exponentielle est la seule qui donne elle-même par translation des abscisses (le temps étant représenté sur les abscisses) modulo une remise à l'échelle des ordonnées. Si le temps s'écoule (ce qui correspond à une translation des abscisses) et qu'aucune transition n'est tirée, le processus, étant dans le même état que l'instant précédent, doit se comporter de la même façon qu'avant. Le fait de savoir que le processus est toujours dans le même état correspond à la remise à l'échelle des ordonnées. La *figure* 2.1 représente la probabilité au cours du temps d'être dans un état donné, noté $s$, d'une CMTC sachant qu'au temps 0 la CMTC était dans $s$.

Dans un système réel, il est courant d'observer des distributions, associées au temps de séjour dans un certain état, qui ne sont pas exponentielles. C'est par exemple le cas de la distribution du temps de séjour d'un message dans un réseau. La courbe de droite de la *figure* 2.2 montre la distribution de probabilité du temps aller retour d'un message de type ping. Au temps 0 un message part de l'ordinateur client, traverse le réseau LAN, ricoche contre l'ordinateur serveur et revient par le LAN vers l'ordinateur client. Les données pour construire cette courbe ont été obtenues avec 10000 pings à travers un réseau LAN réel.

Aussi, afin d'avoir des modèles plus réalistes il peut être intéressant de représenter des distributions non-exponentielles associées au temps de séjour dans tel ou tel état. Ceci est en fait possible en représentant un seul état par plusieurs états et en considérant la probabilité de sortir de cet ensemble d'états. La quantité d'états et la manière de les connecter entre eux conditionnent alors la forme de la distribution. Les distributions

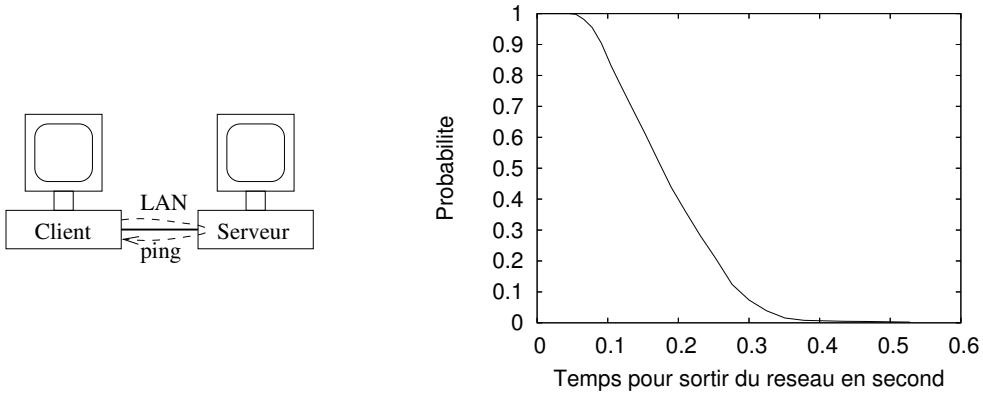



Figure 2.2: *Distribution du temps de séjour d'un message de type ping dans un réseau LAN*

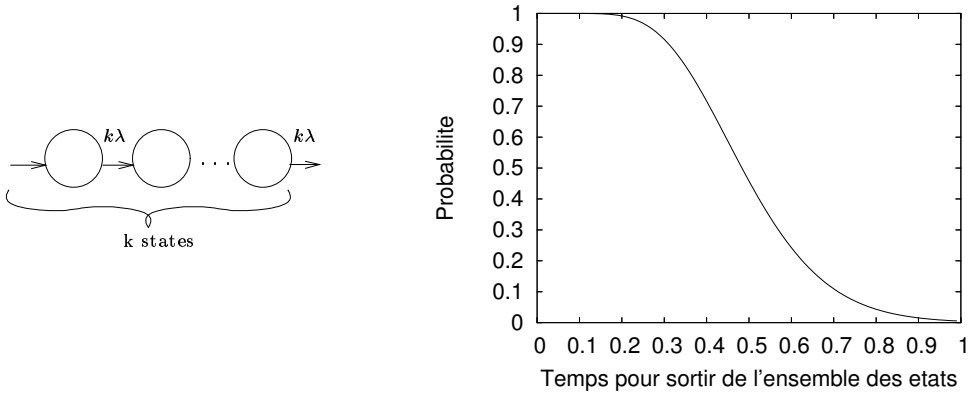



Figure 2.3: *Distribution d'une loi Erlang-10 de taux 1/2*

obtenues de cette manière s'appellent distributions de type phase [49, 7, 57]. Dans la *figure* 2.3 la chaîne de Markov à temps continu à gauche représente une distribution Erlang-$k$ de taux $\lambda$. Une distribution Erlang-$k$ de taux $\lambda$ est une distribution de type phase construite par une séquence de $k$ états et de $k$ transitions de taux $k\lambda$. Le taux $\lambda$ de la distribution signifie qu'en moyenne le temps pour traverser les $k$ états et sortie du $k^{ième}$ est de $1/\lambda$. La courbe à droite de la figure représente la probabilité en fonction du temps de sortir du dernier état. L'entier $k$ conditionne la pente de la distribution. Plus $k$ est grand plus la pente est raide. La section 3.3 présente les définitions exactes se rapportant aux distributions de type phase.

### 2.2.3 Réalisme du modèle suivant les paramètres

L'objectif du concepteur de simulation est de trouver un paramétrage de son modèle qui améliore les performances de la simulation réelle. Ceci nécessite alors que son modèle soit réaliste pour toutes les valeurs possibles des paramètres : par exemple les différentes

tailles de messages, les vitesses de la machine qui héberge l'intergiciel, les caractéristiques du réseau employé, etc. On ne s'intéresse bien entendu qu'à des valeurs de paramètres comprises dans des intervalles réalistes. Par exemple il n'est pas nécessaire d'étudier les comportements de la simulation quand la taille des messages est supérieure à la limite autorisée par l'intergiciel choisi, 1Mo pour le cas de CERTI[1]. Nous verrons dans la section 6.1 comment résoudre, de façon simple mais partielle, cette difficulté.

### 2.2.4 Réalisme et structure : l'approximation sur des modèles PEPA

Afin d'obtenir une modélisation réaliste d'une simulation distribuée il est possible d'utiliser des algorithmes, dits de fitting, permettant d'approcher des distributions probabilistes réelles par des distributions de type phase, c'est-à-dire descriptibles par des CMTCs. De cette manière il est possible d'utiliser les méthodes de model checking probabiliste sur des modèles à la fois complexes et réalistes. Un modèle dépend d'un certain nombre de paramètres (tailles de messages, vitesses des machines, etc). Aussi, afin de conserver son réalisme suivant ses différent paramètres, on peut appliquer ces méthodes de fitting sur différentes choix de paramètres pour ensuite générer une CMTC paramétrique modélisant le système dans ses différentes configurations possibles (voir la *section* 6.1).

Bien que cette approche puisse donner de bon résultats, elle souffre du problème suivant : les modèles obtenus par ces algorithmes de fitting sont incompréhensibles par un utilisateur. Même si la connaissance est bien présente à l'intérieur du modèle il est difficile, voire impossible, d'analyser cette connaissance et d'en extraire, par exemple, un système en terme de ses composants et de leurs interactions. Il est donc possible d'obtenir des modèles réalistes de systèmes aussi complexes que les simulations distribuées temps réel mais il est difficile de manipuler de tels modèles, de les transformer aiséement et de les réutiliser pour d'autres situations.

Supposons par exemple qu'à l'aide de ces différentes méthodes de fitting le concepteur ait réussi à obtenir un modèle très réaliste d'un système distribué composé d'un certain réseau et de certains routeurs. Admettons que le concepteur doive essayer un autre type de réseau tout en gardant les mêmes routeurs, il aura envie de conserver la connaissance, et donc le réalisme, de son modèle concernant ses routeurs et de changer simplement son ancien modèle de réseau par un nouveau sans avoir à refaire tout le travail de modélisation du système complet. Pour cela il a besoin de représenter son modèle de manière compositionnelle. Un ensemble de valeurs enfouies et inextricables à l'intérieur d'une chaîne de Markov ou d'un polynôme ne lui sera pas d'un grand secours. C'est pourquoi une des principales contributions de cette thèse concerne l'extension d'algorithmes de fitting sur les algèbres de processus stochastiques, et plus particulièrement sur l'algèbre PEPA [33]

[1] CERTI [13] est un intergiciel développé au CERT, le centre ONERA de Toulouse.

que nous avons utilisée. Le *chapitre* 6 introduit en détail ce problème et propose une solution.

# Chapter 3

# Pré-requis

## 3.1 Rappels sur la théorie des probabilités

Une probabilité est une valeur comprise entre 0 et 1 qui indique la chance qu'un événement se produise par rapport à d'autres évènements. Elle peut se voir aussi comme la valeur limite du nombre d'occurrences d'un événement par rapport au nombre d'expériences répétées dans des conditions identiques au sein d'un univers donné. Une probabilité de 1/2 pour pile et 1/2 pour face lors de l'expérience répétée du lancer d'une pièce signifie qu'après un grand nombre de lancers, les proportions d'occurrences entre pile et face s'équilibrent.

On rappelle ici le vocabulaire mathématique employé dans la suite de ce mémoire. Pour davantage d'information le lecteur pourra se référer aux ouvrages [44, 46, 28] :

- Une *expérience* ou encore *expérience élémentaire*, souvent notée $\omega$ est un résultat possible d'une expérience.
- L'*univers* ou encore *espace fondamental*, souvent noté $\Omega$, est l'ensemble de toutes les expériences possibles.
- Un *événement* est une partie de $\Omega$.
- Une *tribu* ou encore $\sigma$-*algèbre*, souvent notée $\mathcal{F}$, est un ensemble de parties de $\Omega$, contenant $\Omega$, stable par passage au complémentaire et par réunion finie ou infinie dénombrable. $\mathcal{F}$ définit l'ensemble de tous les évènements possibles.

- Une mesure de *probabilité* est une fonction $\sigma$-*additive*[1] $\mathbb{P} : \mathcal{F} \mapsto [0;1]$ telle que :

$$\mathbb{P}(\Omega) = 1$$

$$\forall A, B \in \mathcal{F}\ \mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) \quad \textit{(additivité)}$$

$$\forall A_1, \ldots, A_i, \ldots \in \mathcal{F} \text{ tel que } A_i \cap A_j = \emptyset,\ i \neq j,$$

$$\mathbb{P}(\bigcup_{i>0} A_i) = \sum_{i>0} \mathbb{P}(A_i) \quad (\sigma\textit{-additivité})$$

- Le couple $(\Omega, \mathcal{F})$ est alors appelé espace mesurable et le triplet $(\Omega, \mathcal{F}, P)$ espace de probabilité ou espace probabilisé.

On définit les notions de probabilité conditionnelle et probabilité d'évènements indépendants

- La probabilité conditionnelle permet de définir la probabilités d'évènements sachant que d'autres évènements se sont produits. Soit $A, B \in \mathcal{F}$ avec $\mathbb{P}(B) > 0$ deux évènements, on définit la probabilité conditionnelle de $A$ sachant $B$ comme :

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

- Si la probabilité d'avoir $A$ sachant $B$ (respectivement $B$ sachant $A$) et égale à la probabilité d'avoir $A$ (respectivement $B$) alors on dira que $A$ et $B$ sont deux évènements indépendants :

$$A \text{ et } B \text{ sont indépendants} \quad \Leftrightarrow \quad \mathbb{P}(A|B) = \mathbb{P}(A) \text{ et } \mathbb{P}(B|A) = \mathbb{P}(B)$$

Il facile de montrer d'après la définition de la probabilité conditionnelle que :

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A) \times \mathbb{P}(A)}{\mathbb{P}(B)}$$

Cette équation s'appelle la *formule de Bayes*. Aussi, d'après la définition de l'indépendance et de la probabilité conditionnelle on peut déduire que :

$$A \text{ et } B \text{ sont indépendants} \quad \Leftrightarrow \quad \mathbb{P}(A \cap B) = \mathbb{P}(A) \times \mathbb{P}(B)$$

[1] Le lecteur peut se demander pourquoi l'ensemble des évènements possibles d'un espace de probabilité est nécessairement une tribu ou pourquoi la fonction de probabilité est forcement $\sigma$-additive. Bien sûr il s'agit d'un choix de modélisation du concept intuitif de probabilité. Mais voici tout de même deux arguments pour justifier un tel choix :

1. Tout d'abord, l'union finie et infinie, et le complémentaire offrent un langage de base pour exprimer de nouveaux évènements à partir d'évènements élémentaires.
2. Ensuite, les opérateurs de $\sigma$-union et de $\sigma$-additivité sont indispensables pour prouver l'un des théorèmes les plus important de la théorie des probabilités : la loi des grands nombres (qui exprime la relation entre probabilité et fréquence des événements). Ils se justifient donc même quand $\Omega$ est finie.

Si la probabilité d'un événement donnée est nulle on dira que cet évènement est *négligeable*, et si sa probabilité est égale à 1 on dira que cet évènement est *presque sûr* ou arrive *presque sûrement*. Il arrive d'utiliser ces deux notions sur des évènements n'appartenant pas à la tribu considérée. Dans ce cas on dit que tout sous-ensemble d'un évènement négligeable est *négligeable* et tout sur-ensemble d'un évènement presque sûr est *presque sûr*.

**Exemple 3.1** Voici un exemple à partir d'un lancé d'une pièce épaisse.

- Les expériences élémentaires sont :
  - la pièce tombe sur un de ses côtés et laisse apparaître le côté *face*,
  - la pièce tombe sur l'autre côté et laisse apparaître le côté *pile*,
  - la pièce tombe sur sa *tranche*.

  L'univers des expériences élémentaire est $\Omega_\circ = \{face, pile, tranche\}$.
- On choisit l'ensemble des évènements possibles comme l'ensemble de toutes les parties de $\Omega_\circ$ :

$$\begin{aligned}\mathcal{F}_\circ \;=\; \{\;& \emptyset,\\ & \{face\}, \{pile\}, \{tranche\}\\ & \{face, pile\}, \{face, tranche\}, \{pile, tranche\}\\ & \Omega_\circ\\ \}&\end{aligned}$$

- On choisit d'abord les probabilités des expériences élémentaires :

$$\mathbb{P}_\circ(\{face\}) = \mathbb{P}_\circ(\{pile\}) = 49/100$$

$$\mathbb{P}_\circ(\{tranche\}) = 2/100$$

  Ensuite, la propriété de la $\sigma$-*additivité* permet de définir les probabilités pour tous les autres éléments de la tribu :

$$\mathbb{P}_\circ(\{pile, face\}) = 98/100$$

$$\mathbb{P}_\circ(\{pile, tranche\}) = \mathbb{P}_\circ(\{face, tranche\}) = 51/100$$

$$\mathbb{P}_\circ(\Omega_\circ) = 1$$

Le lecteur pourra vérifier que $(\Omega_\circ, \mathcal{F}_\circ, \mathbb{P}_\circ)$ est bien un espace de probabilité car $\mathcal{F}_\circ$ et $\mathbb{P}_\circ$ vérifient les définitions de tribu et de mesure de probabilité. De façon triviale l'événement $\emptyset$ est négligeable et l'événement $\Omega_\circ$ est presque sûr. La probabilité conditionnelle d'avoir $\{face\}$ sachant $\{pile\}$ est nulle car les évènements $\{face\}$ et $\{pile\}$ sont disjoints. La probabilité conditionnelle d'avoir $\{pile\}$ sachant $\{pile, tranche\}$ est :

$$\mathbb{P}(\{pile\}|\{pile, tranche\}) =$$

$$\frac{\mathbb{P}(\{pile\} \cap \{pile, tranche\})}{\mathbb{P}(\{pile, tranche\})} = \frac{\mathbb{P}(\{pile\})}{\mathbb{P}(\{pile, tranche\})} = \frac{49/100}{51/100} = 49/51$$

### 3.1.1 Élément Aléatoire

Un élément aléatoire est une application mesurable (voir [44, 46, 28] pour la définition d'application mesurable) entre deux espaces mesurables $(\Omega, \mathcal{F}_\Omega)$ et $(E, \mathcal{F}_E)$, souvent notée $X$. Si l'ensemble d'arrivée $E$ de $X$ est $\mathbb{R}$ ou une partie de $\mathbb{R}$ on pourra préciser qu'il s'agit d'une variable aléatoire, au lieu d'élément aléatoire. Si cet ensemble d'arrivée est $\mathbb{N}$, ou un ensemble enumérable ou fini, on pourra parler d'une variable aléatoire discrète. Si cet ensemble est un espace vectoriel on pourra parler de vecteur aléatoire, etc.

## 3.2 Les processus stochastiques

Un *processus probabiliste* est

1. un processus, c'est-à-dire une entité dont les caractéristiques évoluent au cours du temps,
2. probabiliste, dont les caractéristiques sont probabilisées.

Dans certains ouvrages le terme *processus stochastique* est utilisé pour désigner un processus probabiliste à temps continu. Dans d'autres ouvrages ces deux termes sont équivalents. Dans ce mémoire, un processus stochastique désignera un processus probabiliste à temps continu.

L'espace fondamental $\Omega$ d'un processus probabiliste (ou stochastique) correspond à l'ensemble de toutes les exécutions infinies ou finies du processus et la tribu $\mathcal{F}_\Omega$ à l'ensemble de tous les ensembles d'exécutions définis comme étant mesurables. $\Omega$ et $\mathcal{F}_\Omega$ n'ont pas besoin d'être explicités car les probabilités du processus sont définies et manipulées au travers de lois d'éléments aléatoires. Un processus probabiliste est formalisé par une famille d'éléments aléatoires $X = (X_t)_{t \in T}$. Les valeurs possibles de chaque élément $X_t$ sont les états possibles du système à l'instant $t \in T$ où $T$ définit l'ensemble des instants. Cet ensemble peut être discret ($\mathbb{N}$ par exemple) ou continu ($\mathbb{R}_+$ par exemple, dans le cas d'un processus stochastique). La loi de probabilité sur $X$ définit la probabilité du processus à chaque instant de se trouver dans tel ou tel état, on appelle cette loi la *trajectoire* du processus.

### 3.2.1 Les processus de Markov

Un *processus de Markov* est un processus probabiliste dont le comportement futur ne dépend que de l'instant courant et pas de ce qui s'est passé avant. Pour cette raison, on dit qu'un processus de Markov est sans mémoire. On peut parler indifféremment de processus de Markov ou *processus markovien* ou encore processus possédant la *propriété de Markov*. En voici la définition formelle.

**Définition 3.1 (Propriété de Markov)** *Un processus probabiliste $(X_t)_{t \in T}$ possède la* propriété de Markov *si et seulement si pour tout $t < t'$, $\mathbb{P}(X_{t'} = s' \mid X_r = f(r), 0 \leq r \leq$*

$t) = \mathbb{P}(X_{t'} = s' | X_t = f(t))$ *quelle que soit* $f$*, une fonction du temps vers les états du système.*

Dans ce mémoire on s'intéressera aux processus markoviens qui sont *homogènes dans le temps*, c'est-à-dire tels que les probabilités de prendre les transitions en fonction de l'état courant restent fixes au cours temps :

$$\forall t \leq t', \mathbb{P}(X_{t'} = s' | X_t = s) = \mathbb{P}(X_{t-t'} = s' | X_0 = s)$$

Il est toutefois possible de rendre à un processus markovien sa mémoire, et même de le rendre non-homogène dans le temps en inscrivant dans ses états courants tout ou partie de son histoire. Néanmoins le processus avançant dans le temps et la quantité d'information pour décrire un état (et l'histoire qui a conduit à cet état) grandissante, la quantité d'états nécessaire pour décrire le système peut devenir infinie. Il est donc courant d'inscrire dans les états courants d'un processus markovien seulement une partie de son histoire, comme par exemple les $n$ derniers états rencontrés, on parle alors de *processus de Markov de degré* $n$.

### 3.2.2 Les Chaînes de Markov à Temps Discret

Une *chaîne de Markov à temps discret* (CMTD) est un processus de Markov dont l'espace d'états est fini ou infini dénombrable et le temps modélisé par l'ensemble des entiers naturels. Une CMTD peut se représenter par un graphe orienté étiqueté. Les sommets de ce graphe représentent les états et les valeurs sur les transitions représentent la probabilité avec laquelle la transition est prise.

**Définition 3.2 (Chaîne de Markov à temps discret)** *Une chaîne de Markov à temps discret, notée aussi CMTD, est la donnée de :*

- $S$ *un ensemble d'états fini ou infini dénombrable,*
- *une matrice* $\mathbf{P} : S \times S \mapsto [0;1]$ *appelée* matrice de transition*, telle que pour tout état* $s \in S$ *la somme des probabilités de tirer une transition sortante de* $s$ *est égale à 1, ou 0 et dans ce cas l'état* $s$ *est qualifié d'*absorbant *:*

$$\forall s \in S, \sum_{s' \in S} \mathbf{P}(s, s') = \begin{cases} 1 \\ 0 \quad \textit{si } s \textit{ est absorbant} \end{cases}$$

- *un vecteur (ici il s'agit d'un vecteur ligne)* $\pi : S \mapsto [0;1]$ *appelée la* distribution initiale*, tel que :*

$$\sum_{s \in S} \pi(s) = 1$$

*Soit $X = (X_t)_{t\in\mathbb{N}}$ le processus markovien associé à une matrice de transition $\mathbf{P}$ et une distribution initiale $\pi$. La loi de $X$ est définie par les équations suivantes :*

$$\forall s \in S,\ \mathbb{P}(X_0 = s) = \pi(s)$$

$$\forall s, s' \in S,\ \mathbb{P}(X_1 = s' | X_0 = s) = \mathbf{P}(s, s')$$

Il est à noter que ces deux formules définissent entièrement la loi de $X$. Il est possible de démontrer par récurrence [28], en prenant en compte la propriété de Markov et d'homogénéité que la probabilité d'atteindre un état $s'$ depuis $s$ en $n$ étapes se calcule ainsi :

$$\mathbb{P}(X_{t+n} = s' | X_t = s) = \mathbf{P}^n(s, s')$$

où $\mathbf{P}^n$ est la $n^{ième}$ puissance associée au produit matriciel.

Enfin, on appelle un *chemin* ou une *exécution* de $X$ une suite d'états de $S$ indexée sur le temps $\sigma : T \mapsto S$ définissant à chaque instant $t$ l'état $\sigma(t)$ visité par le chemin. Si $X$ ne possède pas d'état absorbant alors $T = \mathbb{N}$ sinon $T = \{0, 1, 2, \ldots, m\}$ et $\sigma(m)$ est un état absorbant.

La *figure* 3.1 représente une chaîne de Markov, à gauche sa représentation sous forme de graphe et à droite sa matrice de transition. La flèche entrante pointant vers l'état $s_1$ du graphe désigne l'état initial, ce qui correspond au vecteur initial, $\pi(s_1) = 1$ et $\pi(s_2) = \pi(s_3) = 0$. Un chemin possible pour cette chaîne pourra être par exemple :

$$\sigma(0) = s_1 \quad \sigma(1) = s_2 \quad \sigma(2) = s_1 \quad \sigma(3) = s_3 \quad \sigma(4) = s_2 \quad \ldots$$

qui pourra aussi se représenter de la manière suivante :

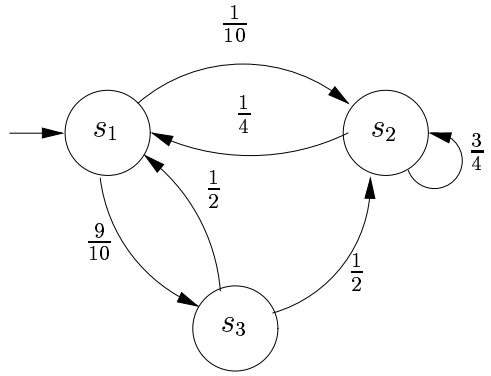
$$s_1 \longrightarrow s_2 \longrightarrow s_1 \longrightarrow s_3 \longrightarrow s_2 \ldots$$

### 3.2.3 Les chaînes de Markov à temps continu

Une chaîne de Markov à temps continu (CMTC) est un processus de Markov dont l'espace d'état est fini ou infini dénombrable et dont le temps est modélisé par $\mathbb{R}_+$. Une CMTC peut se représenter par un graphe orienté étiqueté. Les sommets de ce graphe représentent les états et les valeurs sur les transitions représentent le taux avec lequel une transition est prise. Le lecteur désirant plus d'information à propos des CMTCs pourra se reporter aux ouvrages [51, 46, 28].

**Définition 3.3 (Chaîne de Markov à temps continu)** *Une* chaîne de Markov à temps continu*, notée aussi CMTC, est la donnée de :*

- *$S$ un ensemble fini ou infini dénombrable d'états,*




$$\begin{pmatrix} 0 & \frac{1}{10} & \frac{9}{10} \\ \frac{1}{4} & \frac{3}{4} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Figure 3.1: *Chaîne de Markov à temps discret et sa matrice de transition*

- *Une matrice* $\mathbf{Q} : S \times S \mapsto \mathbb{R}$*, le générateur infinitésimal, telle que :*

$$\forall s \in S, \mathbf{Q}(s,s) = -\sum_{\substack{s' \in S \\ s' \neq s}} \mathbf{Q}(s,s')$$

- *Un vecteur (ici il s'agit d'un vecteur ligne)* $\pi : S \mapsto [0,1]$*, la distribution initiale tel que :*

$$\sum_{s \in S} \pi(s) = 1$$

*Soit* $X = (X_t)_{t \in \mathbb{R}_+}$ *le processus markovien associé au générateur infinitésimal* $\mathbf{Q}$ *et à la distribution initiale* $\pi$*. La loi de* $X$ *est définie par les équations ci-dessous :*

$$\forall s \in S,\ \mathbb{P}(X_0 = s) = \pi(s)$$

$$\forall s, s' \in S\ \forall t, \Delta t \in \mathbb{R}_+,$$

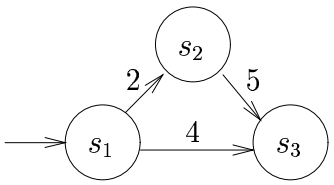
$$\mathbb{P}(X_{t+\Delta t} = s' | X_t = s) = \begin{cases} \mathbf{Q}(s,s') \times \Delta t + o(\Delta t) & si\ s \neq s' \\ 1 - \mathbf{Q}(s,s) \times \Delta t + o(\Delta t) & si\ s = s' \end{cases}$$

*avec* $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$

La définition ci-dessus décrit le comportement du processus par les variations infinitésimales de la probabilité, en fonction du temps, de rester ou de changer d'état. Si $p_s(t)$ désigne la probabilité $\mathbb{P}(X_t = s)$ et $\mathbf{p}(t)$ le vecteur ligne avec $p_s(t)$ pour $s \in S$ une composante de ce vecteur, la trajectoire de $X$, c'est-à-dire $\mathbf{p}(t)$, correspond à la solution du système différentiel ordinaire $\dot{\mathbf{p}}(t) = \mathbf{p}(t) \times \mathbf{Q}$ :

$$\dot{\mathbf{p}}(t) = \pi e^{t \times \mathbf{Q}}$$

Le lecteur peut consulter l'*annexe* A.1 pour plus de détails concernant la dérivation de ce système différentiel. Ainsi, la probabilité de rester dans un état $s$ décroît de manière




$$\begin{pmatrix} -6 & 2 & 4 \\ 0 & -5 & 5 \\ 0 & 0 & 0 \end{pmatrix}$$

Figure 3.2: *Exemple d'une chaîne de Markov à temps continu*

exponentielle en fonction du temps (voir aussi l'*annexe* A.2). D'un point de vue microscopique $\mathbf{Q}(s, s')$ représente le coefficient directeur de la loi de probabilité d'être dans l'état $s'$ sachant que le système est dans l'état $s$. Il correspond aussi d'un point de vue macroscopique à la fréquence moyenne de tirer de la transition $(s, s')$. Ainsi le temps moyen pour le processus de sortir de $s$ est $\frac{1}{-\mathbf{Q}(s,s)}$. La probabilité pour passer de l'état $s$ à $s'$ sachant qu'une transition est tirée est :

$$\mathbf{P}(s, s') = \begin{cases} \frac{\mathbf{Q}(s,s')}{-\mathbf{Q}(s,s)} & \text{si } \mathbf{Q}(s, s) \neq 0 \\ 0 & \text{sinon} \end{cases}$$

L'*annexe* A.3 contient la justification mathématique du calcul de $\mathbf{P}$.

**Exemple 3.2** Soit $X$ une CMTC défini par :
- $S = \{s_1, s_2, s_3\}$,
- $\pi(s_1) = 1$, $\pi(s_2) = \pi(s_3) = 0$,
- $\mathbf{Q}(s_1, s_2) = 2$, $\mathbf{Q}(s_1, s_3) = 4$, $\mathbf{Q}(s_2, s_3) = 5$, $\mathbf{Q}(s_1, s_1) = \mathbf{Q}(s_2, s_1) = \mathbf{Q}(s_2, s_2) = 0 = \mathbf{Q}(s_3, s_1) = \mathbf{Q}(s_3, s_2) = 0$.

$X$ suit les lois de probabilités[2] ci-dessous :

$$\begin{aligned} &\mathbb{P}(X_t = s_1) = e^{-6t} \\ &\mathbb{P}(X_t = s_3) = 1 + e^{-6t} - 2e^{-5t} \\ &\mathbb{P}(X_{t+t'} = s_3 | X_t = s_3) = 1 \\ &\mathbb{P}(X_{t+t'} = s_1 | X_t = s_3) = \mathbb{P}(X_{t+t'} = s_1 | X_t = s_2) = 0 \end{aligned}$$

La *figure* 3.2 représente à gauche le graphe de cette CMTC et à droite sont générateur infinitésimal. La probabilité initiale est $\pi(s_1) = 1$ et $\pi(s_2) = \pi(s_2) = 0$ et est indiquée par la flèche pointant vers l'état $s_1$ (comme dans la figure 3.1).

On appelle *probabilité transitoire* de $X$ à l'instant $t$ les probabilités que $X$ se trouve dans chacun des états, ce qui correspond aux valeurs du vecteur $\mathbf{p}(t)$. Les CMTCs vérifient

[2]Ces lois s'obtiennent à partir de $\mathbf{p}(t) = \pi e^{t \times \mathbf{Q}}$. Mais le calcul est assez compliqué. Pour obtenir $\mathbb{P}(X_t = s_3)$ il faut utiliser un produit de convolution sur les densités de probabilités de passage entre $s_1 \to s_2$ et $s_2 \to s_3$.

une propriété importante concernant leur comportement asymptotique. Quand $t$ tend vers l'infini leur probabilité transitoire se stabilise :

$$\exists \mathbf{c} \in S \mapsto [0;1],\ \lim_{t\to\infty} \mathbf{p}(t) = \mathbf{c} \text{ et } \sum_{s\in S} \mathbf{c}(s) = 1$$

**Exemple 3.3** Soit $X$ une CMTC d'espace d'état $S = \{s_1, s_2, s_3\}$, avec l'état initial $s_1$ et la matrice de transition :

$$\mathbf{Q} = \begin{pmatrix} -4 & 1 & 3 \\ 5 & -7 & 2 \\ 3 & 3 & -6 \end{pmatrix}$$

Voici les valeurs numériques de $\mathbf{p}(t)$ à différents instants :

| $t$ | 0 | 0.1 | 0.3 | 0.5 | 0.7 | 1 |
|---|---|---|---|---|---|---|
| $\mathbf{p}(t)$ | $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ | $\begin{pmatrix} 0.71 \\ 0.08 \\ 0.19 \end{pmatrix}$ | $\begin{pmatrix} 0.50 \\ 0.18 \\ 0.30 \end{pmatrix}$ | $\begin{pmatrix} 0.49 \\ 0.19 \\ 0.30 \end{pmatrix}$ | $\begin{pmatrix} 0.48 \\ 0.20 \\ 0.31 \end{pmatrix}$ | $\begin{pmatrix} 0.48 \\ 0.20 \\ 0.31 \end{pmatrix}$ |

On peut en effet remarquer que les probabilités transitoires se stabilisent à partir du temps 0.7.

On définit maintenant la notion de *chemin* ou *exécution* d'une CMTC ainsi que des notations utilisées dans la suite du document.

**Définition 3.4 (Chemin, exécution)** *On appelle* chemin *ou* exécution *de $X$ une fonction $\sigma : \mathbb{R}_+ \mapsto S$ définissant à chaque instant $t$ l'état $\sigma(t)$ visité par le processus.*

Remarque : dans ce mémoire les CMTCs considérées auront la plupart du temps un état initial, noté en général $s_\star$, au lieu d'une distribution initiale. Ceci revient simplement à considérer une CMTC dont la distribution initiale $\pi$ est telle que $\pi(s_\star) = 1$ et $\pi(s) = 0$ pour tout $s \neq s_\star$.

Il est montré dans [22] que l'ensemble des chemins (ou exécutions) possédant une infinité de transitions entre deux instants est de probabilité nulle. On peut ainsi sans perte de généralité représenter une exécution comme une suite finie ou infinie d'états et d'instants, notée de la manière suivante :

$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots s_i \xrightarrow{t_i} \ldots$$

Une telle exécution commence donc par l'état $s_0$, y reste pendant un temps $t_0$ puis saute dans l'état $s_1$, y reste pendant un temps $t_1$, etc. On introduit aussi les deux notations suivantes :

– Pour tout $i \in \mathbb{N}$, $\sigma[i]$ désigne le $i^{ème}$ état de $\sigma$ :

$$\sigma[i] = s_i$$

– Pour tout $t \in \mathbb{R}_+$, $\sigma@t$ désigne l'état de $\sigma$ au temps $t$ :

$$\forall t \in \mathbb{R}_+,\ \exists i,\ t < \sum_{j=0}^{i} t_j,\ \sigma@t = \sigma[i]$$

Si le chemin est fini, noté $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots s_{k-1} \xrightarrow{t_{k-1}} s_k$, cela implique que $s_k$ est un état absorbant, autrement dit $\mathbf{Q}(s_k, s_k) = 0$. Dans ce cas $\sigma[i]$ est seulement défini pour tout $i \leq k$ et $\sigma@t = s_k$ pour tout $t \geq \sum_{j=0}^{k-1} t_j$.

Du fait de la propriété sans-mémoire du processus, le seul type de distribution définissant la probabilité, en fonction du temps, de prendre une transition en fonction de l'état courant est exponentielle car l'exponentielle est la seule fonction dont toute translation sur ses abscisses donne elle-même modulo une remise à l'échelle sur ses ordonnées (voir la *page* 18 *section* 2.2.2 pour plus d'explications). Quoiqu'il en soit il est quand même possible d'approcher n'importe quel type de distribution en composant un nombre fini d'états. De telles distributions sont appelées distributions de type phase [50]. Dans la section 3.3 nous aborderons plus en détail les distributions de type phase.

### 3.2.4 Comparaison entre CMTD et CMTC

Étant données une CMTC et une certaine unité de temps, on peut construire une CMTD qui approche le comportement de la CMTC. Soient $\mathbf{Q}$ le générateur infinitésimal d'une CMTC et $\pi$ sa distribution initiale. La matrice de transitions $\mathbf{P}$ de la CMTD associée s'obtient :

– en multipliant $\mathbf{Q}$ par l'unité de temps choisie $\epsilon$,
– en complétant la diagonale de telle sorte que la somme des probabilités de chaque ligne soit égale à 1.

Ceci correspond à l'opération matricielle suivante :

$$\mathbf{P} = \mathbf{I} + \epsilon\mathbf{Q}$$

où $\mathbf{I}$ est la matrice identité. Les distributions initiales des deux chaînes restent identiques. La *figure* 3.3 montre une CMTC et sa CMTD associée. Dans cette CMTD associée, chaque transition marque le passage d'un temps égal à l'unité choisie. Le comportement de la CMTD associée tend vers le comportement de la CMTC quand l'unité de temps tend vers zéro. C'est ce que formalise l'équation suivante (voir l'*annexe* A.4 pour les détails mathématiques) :

$$\forall t \in \mathbb{R}_+ \lim_{\epsilon \to 0} \pi(\mathbf{I} + \epsilon\mathbf{Q})^{[t/\epsilon]} = \pi e^{t\mathbf{Q}}$$

Malgré cette propriété, il n'est pas évident en pratique de produire une CMTD fidèle à sa CMTC car la matrice de transition de la CMTD associée est calculée avec une précision plus ou moins bonne. Le choix de modélisation d'un système par une CMTC ou par une CMTD n'est donc pas arbitraire. Cette équation définit en fait la méthode numérique

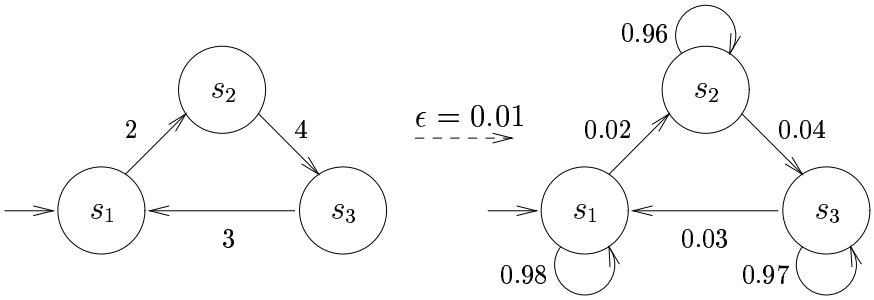



Figure 3.3: *A gauche une chaîne de Markov à temps continu, à droite sa chaîne à temps discret associée avec pour unité de temps* $\epsilon = 0.01$

la plus élémentaire pour approcher la trajectoire d'une CMTC. Il existe de nombreuses méthodes numériques plus efficaces en temps et en précision pour approcher la trajectoire d'une CMTC. C'est pourquoi travailler avec des CMTDs plutôt que des CMTCs pour des raisons uniquement calculatoires ne constitue pas une bonne démarche. Dans la cadre du problème de l'évaluation de performances d'un système distribué l'aspect temporel du modèle est prédominant car les performances portent sur des caractéristiques temporelles. Il apparaît donc plus intéressant de travailler avec des CMTCs plutôt que des CMTDs.

## 3.3 Les distributions de type phase

Une distribution de type phase [50] est obtenue par une CMTC composée de plusieurs états transitoires, appelés phases, et d'un état absorbant. Elle correspond à la probabilité d'être dans l'état absorbant en fonction du temps. Les distributions de type phase sont intéressantes car elle permettent de représenter des distributions complexes tout en étant des CMTCs à états fini.

**Définition 3.5 (Distribution de type phase)** *Une distribution de type phase est une CMTC qui possède un état absorbant, elle est définie par :*

- $S$ *un ensemble fini d'états dont l'état absorbant, noté nil.* $S_{ph}$ *désigne l'ensemble des phases, c'est-à-dire* $S$ *sans nil,* $S = S_{ph} \cup \{nil\}$.
- $\mathbf{t} : S_{ph} \mapsto \mathbb{R}_+$ *est le vecteur de sortie.*
- $\mathbf{T} : S_{ph} \times S_{ph} \mapsto \mathbb{R}$ *est le générateur infinitésimal de phases, tel que :*

$$\forall s \in S_{ph} \quad \mathbf{T}(s,s) = -\mathbf{t}(s) - \sum_{\substack{s' \in S \\ s' \neq s}} \mathbf{T}(s,s')$$

- $\pi : S_{ph} \mapsto [0;1]$ *est la distribution initiale, telle que :*

$$\sum_{s \in S_{ph}} \pi(s) = 1$$

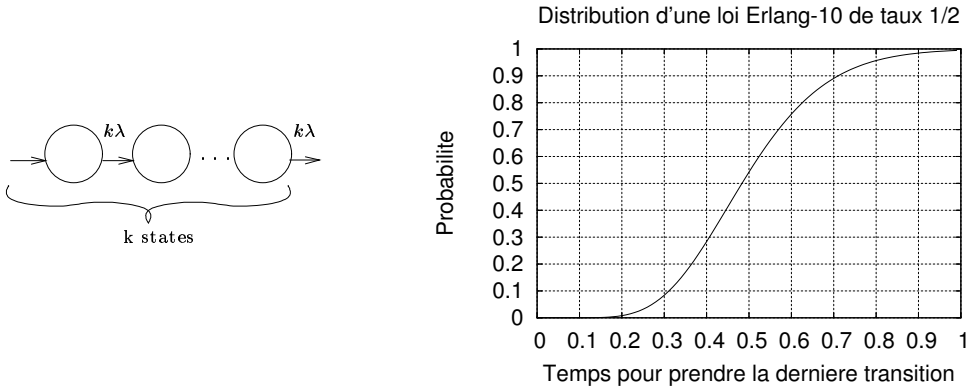



Figure 3.4: *Distribution cumulative de Erlang-k*

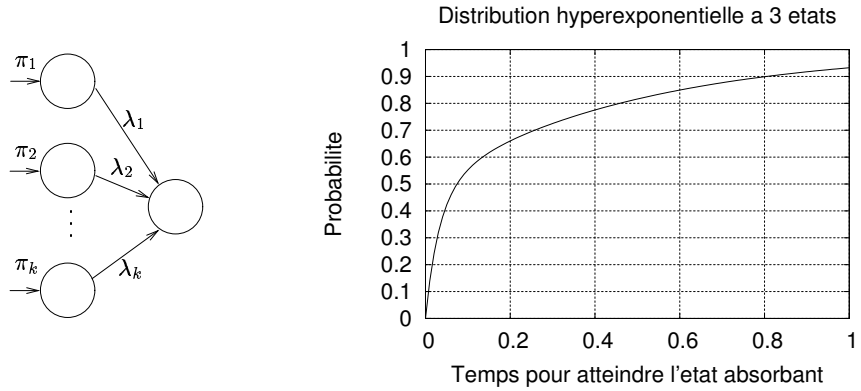



Figure 3.5: *Distribution cumulative hyperexponentielle*

La CMTC définissant une distribution de type phase de générateur infinitésimal de phases $\mathbf{T}$, de vecteur de sortie $\mathbf{t}$ et de distribution initiale $\pi$ a donc pour générateur infinitésimal $\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \mathbf{t} \\ 0 \ldots 0 & 0 \end{pmatrix}$ et pour vecteur initial $(\pi\ 0)$.

Les distributions de Erlang-k définies par une série de k états, les distributions hyperexponentielles définies par une composition parallèle d'états ou les distributions de Cox, une généralisation de Erlang-k et de hyperexponentielles, sont toutes des cas particuliers de distributions de type phase. Les *figures* 3.4, 3.5 et 3.6 représentent respectivement une distribution de Erlang-k, hyperexponentielle et de Cox. En général une distribution de type phase laisse la possibilité d'avoir n'importe quelles transitions entre les états transitoires. Les distributions de type phase de ce type sont plus expressives que les distributions de Cox. En revanche toute distribution de type phase acyclique est équivalente à une distribution de Cox. Nous verrons dans la section 4.3.4 qu'il existe des algorithmes permettant d'induire le générateur infinitésimal et le vecteur de sortie d'un distribution de type phase de sorte que celle-ci approche de suffisamment près une distribution quelconque.

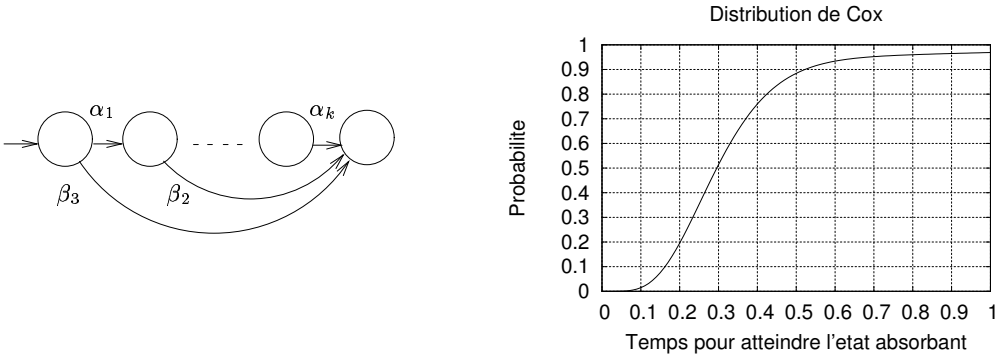



Figure 3.6: *Distribution cumulative de Cox*

# Chapter 4

# État de l'Art

Dans ce chapitre on commence par présenter différentes méthodes pour le model checking probabiliste. On insiste davantage sur le modèle checking pour les chaînes de Markov à temps continu avec la logique CSL car il a été utilisé au *chapitre* 5 pour formaliser et évaluer deux contraintes de performances d'une simulation distribuée. On présente ensuite différents types de modèles définissant des processus probabilistes pour le model checking probabiliste. On présente en détail l'algèbre PEPA qui permet de décrire une chaîne de Markov à temps continu de façon compositionnelle. Enfin on présente l'algorithme EM pour l'approximation de distributions de type phase permettant de trouver les coefficients d'une chaîne de Markov à temps continu avec état absorbant qui maximisent (localement) la vraisemblance d'un ensemble de temps d'absorption.

## 4.1 Outils et méthodes pour le model checking probabiliste

### 4.1.1 Généralités sur la vérification formelle

La vérification formelle [29] est un moyen d'assurer formellement, c'est-à-dire en usant uniquement de la "forme", qu'un modèle ou plus généralement une axiomatique vérifie une propriété. On définit un système formel comme la donnée de :

1. Un langage définissant syntaxiquement l'ensemble des formules logiques du système.
2. Un ensemble de formules logiques appelées axiomes.
3. Un jeu de règles de réécriture pour produire de nouveaux théorèmes à partir d'un ensemble d'axiomes ou de théorèmes.

L'enchaînement des étapes de réécritures entre les axiomes et la propriété à vérifier s'appelle une preuve formelle. L'équivalence entre la preuve formelle et la preuve mathématique informelle consiste à traduire le système formel dans un autre système (en fait peut-être aussi formel que le premier mais éventuellement plus proche de l'intuition), en

général la théorie des ensembles [37]. Cette traduction définit la sémantique du système formel. L'approche preuve formelle est plus intéressante que le test[1] dans la mesure où un jeu de tests peut permettre de conforter ou d'infirmer qu'un modèle vérifie une propriété mais rarement d'affirmer[2].

Bien qu'il soit en théorie et en pratique assez facile de mécaniser le procédé de recherche d'une preuve formelle, il est assez difficile et rare d'obtenir des résultats dans une borne de temps acceptable. Si le système formel choisi est suffisamment riche pour se représenter lui-même, la complexité algorithmique du procédé de recherche n'est bornée par aucune fonction. Il s'agit alors d'un problème indécidable [56, 47].

Dans le cadre du model checking, on parlera souvent de modèle plutôt que d'ensemble d'axiomes ou axiomatique. En réalité il n'existe pas vraiment de distinction entre les deux (voir pour cela les travaux de Herbrand [31]) mais le terme modèle a tendance à être employé pour désigner une axiomatique représentant intuitivement un seul objet[3].

### 4.1.2 Le model checking

Le *model checking* est une technique permettant d'obtenir des preuves formelles suivant un paradigme simple. Voici un exemple de paradigme : *Si la propriété est vraie pour tous les états du modèle alors elle est vraie pour toute exécution de celui-ci.* La méthode consiste alors à explorer tous les états du modèle jusqu'à l'infirmation de la propriété ou, dans le cas où tous les états ont été explorés, son affirmation. L'avantage d'un tel procédé est que si le modèle a une quantité finie d'états la procédure de preuve est sûre d'aboutir[4]. L'inconvénient est que, dans bien des cas, il peut exister des preuves beaucoup plus courtes ne vérifiant pas ce paradigme et pour cette raison, inconstructibles par le model checker. Il est par exemple tout à fait possible de laisser travailler une semaine un model checker pour aboutir au message d'erreur *"dépassement de capacité de mémoire"* sur une propriété démontrable en 5 minutes par un expert humain.

Étant en développement depuis plus de 20 ans, le model checking fait appel à de nombreux langages de description de modèles, comme les automates de Büchi [60] ou les automates temporisés [3], de nombreuses logiques comme LTL [55], CTL [15, 23] ou le $\mu$-calcul [48] et de nombreux algorithmes. Il existe également des méthodes pour compresser

[1] Néanmoins, dans un bon nombre de situations la preuve formelle n'est pas encore applicable et on ne peut donc pas se passer des procédures de tests. De plus, il est toujours indispensable de faire des tests sur le système réel pour vérifier l'adéquation entre le modèle et la réalité.

[2] Sauf sur des propriétés décrivant exactement le cas particulier couvert par le test.

[3] Si du strict point de vue de la preuve formelle la question de savoir si une axiomatique désigne un ou plusieurs objets n'a pas beaucoup de sens, en pratique cela dépend de l'utilisation qu'on fait de cette axiomatique en dehors de la preuve formelle. Par exemple si l'axiomatique définit un automate fini, alors si l'automate est utilisé comme un programme exécutable sur un OS on aura envie de voir un seul objet mais si l'automate est utilisé pour définir l'ensemble des automates avec lesquels il est en relation par rapport à une certaine relation de bi-simulation, alors on aura envie d'en voir plusieurs.

[4] Elle est théoriquement sûre d'aboutir mais pratiquement cela dépend des ressources en espace et en temps, et de la rapidité du matériel.

les modèles à vérifier. La compression destructive, appelée aussi abstraction ou simulation [16], peut accélérer énormément la procédure de model checking. En effet elle consiste à masquer autant que possible les parties du modèle n'apportant pas d'information sur la validité ou l'invalidité d'une propriété donnée. Une telle opération peut toutefois être coûteuse et difficile à réaliser et encore plus difficile à automatiser. La compression non-destructive, réalisée par exemple en codant le modèle avec des BDDs [12] permet de représenter de façon plus concise le modèle qu'avec une description énumérative. Ce qui a en général pour conséquence d'économiser de la ressource mémoire au détriment de la ressource CPU.

### 4.1.3 Le model checking temporisé

Il existe également des model checkers dits temporisés, par exemple Kronos [65] ou Uppaal [9]. Les model checkers temporisés utilisent des automates temporisés [3] comme langage de modélisation et des logiques temporelles temporisées [58] comme langage de spécification des formules à vérifier. Les automates temporisés sont des automates étendus avec des horloges, c'est-à-dire des variables à valeurs dans $\mathbb{R}_+$ qui modélisent le temps passé dans le système. Ceci génère potentiellement des modèles avec une infinité d'états. Toutefois, si l'automate et la logique temporisé sont suffisamment limités dans leurs expressivités [10], il est possible d'abstraire cet automate temporisé par un automate fini équivalent [3] et de traduire la formule en logique temporelle temporisée par une formule en logique temporelle équivalente [8]. Ce qui ramène la vérification du modèle temporisé à une procédure de model checking traditionnelle.

### 4.1.4 Le model checking probabiliste

Dans cette thèse le model checking probabiliste a été utilisé pour vérifier des formules CSL (présenté en *section* 4.1.4) sur un modèle de simulation distribuée HLA décrit par une CMTC. CSL est une extension à temps continu de PCTL. Avant de présenter CSL on présente d'abord PCTL laquelle permet de comprendre plus facilement CSL.

#### La logique PCTL

PCTL pour *Probabilistic Computation Tree Logic* a été introduite en 1994 par Hans Hansson et Bengt Jonsson dans [30]. C'est une extension probabiliste de CTL [15, 23]. PCTL permet de formuler des propriétés probabilistes sur des CMTDs, les états de la CMTD contenant des propositions élémentaires. On donne dans un premier temps la syntaxe des formules de PCTL puis on explique leur sémantique :

**Définition 4.1 (Syntaxe des formules de PCTL)** *Soit $A$ un ensemble fini de propositions atomiques, $A = \{q, r, \ldots\}$.* La syntaxe des formules de PCTL *est définie, par induction, de la manière suivante :*

$$\Phi, \Psi \ ::= \ \mathbf{vrai} \mid A \mid \neg\Phi \mid \Phi \wedge \Psi \mid \mathcal{X}_{\bowtie p}\Phi \mid \Phi \mathcal{U}^{\leq t}_{\bowtie p}\Psi$$

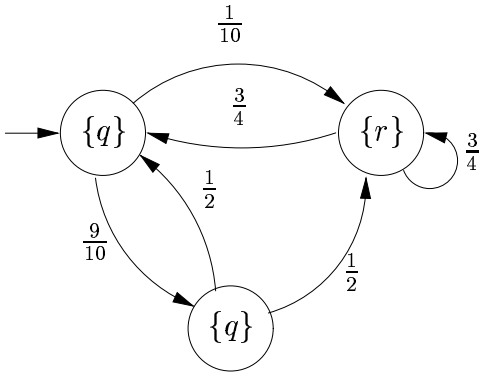



Figure 4.1: *Exemple de modèle pour PCTL*

*avec* $\bowtie \in \{<, \leq, >, \geq\}$, $p \in \mathbb{R}, 0 \leq p \leq 1$ *et* $t \in \mathbb{N} \cup \{\infty\}$

Les propositions atomiques $q \in A$, les formules **vrai** ou celles construites avec les opérateurs $\neg$ ou $\wedge$ s'interprètent de façon traditionnelle et non probabiliste. En revanche les deux opérateurs $\mathcal{X}_{\bowtie p}\Phi$ et $\Phi\mathcal{U}^{\leq t}_{\bowtie p}\Psi$, appelés respectivement opérateur *Next* et opérateur *Until*, permettent de prendre en considération la mesure de probabilité d'un ensemble de chemins spécifié par les formules $\Phi$ et $\Psi$. Dans la formule $\Phi\mathcal{U}^{\leq t}_{\bowtie p}\Psi$, si $t \neq \infty$ alors on parlera de Until *borné* et de Until *non-borné* sinon. Avant de donner la sémantique exacte de chacun des opérateurs de cette logique il faut définir précisément sur quels modèles cette logique s'interprète.

**Définition 4.2 (Modèle d'interprétation de PCTL)** *Un* modèle d'interprétation d'une formule PCTL *est un quintuplet* $(S, s, \mathbf{P}, A, L)$ *où :*

- $S$ *est un ensemble fini d'états,*
- $s \in S$ *est l'état initial (il peut aussi être considéré comme l'état courant de l'interprétation)*
- $\mathbf{P} : S \times S \mapsto [0; 1]$ *est une matrice de transition (voir la* définition *3.2),*
- $A$ *est un ensemble de propositions atomiques,*
- $L : S \mapsto \mathcal{P}(A)$ *une fonction qui assigne à chaque état* $s$ *l'ensemble des propositions atomiques vraies dans* $s$*.*

La *figure* 4.1 représente la même chaîne de Markov que celle de la *figure* 3.1 à la *page* 29. L'ensemble des propositions atomiques $A = \{q, r\}$ et la fonction $L = \{s_1 \mapsto \{q\}, s_2 \mapsto \{r\}, s_3 \mapsto \{q\}\}$ ont été ajoutés. On définit comment une formule de PCTL s'interprète sur un modèle.

**Définition 4.3 (Sémantique de PCTL)** *Soit* $(S, s, \mathbf{P}, A, L)$ *un modèle,* $\Phi$ *et* $\Psi$ *deux formules de PCTL. On définit la* sémantique de PCTL *à l'aide de la relation* $\models$*, par induction :*

- $(S, s, \mathbf{P}, A, L) \models$ **vrai** *est toujours vrai,*
- $(S, s, \mathbf{P}, A, L) \models a$ *si et seulement si* $a \in L(s)$*,*

- $(S, s, \mathbf{P}, A, L) \models \neg\Phi$ *si et seulement si on n'a pas* $(S, s, \mathbf{P}, A, L) \models \Phi$,
- $(S, s, \mathbf{P}, A, L) \models \Phi \wedge \Psi$ *si et seulement si :*

$$(S, s, \mathbf{P}, A, L) \models \Phi \text{ et } (S, s, \mathbf{P}, A, L) \models \Psi$$

- $(S, s, \mathbf{P}, A, L) \models \mathcal{X}_{\bowtie p}\Phi$ *si et seulement si* $\mathbb{P}(\sigma_{\mathcal{X}\Phi}|X_0 = s) \bowtie p$, *avec :*

$$\sigma_{\mathcal{X}\Phi} = \{\sigma|\sigma(0) = s \text{ et } (S, \sigma(1), \mathbf{P}, A, L) \models \Phi\}$$

- $(S, s, \mathbf{P}, A, L) \models \Phi\mathcal{U}^{\leq t}_{\bowtie p}\Psi$ *si et seulement si* $\mathbb{P}(\sigma_{\Phi\mathcal{U}^{\leq t}\Psi} \mid X_0 = s) \bowtie p$, *avec :*

$$\sigma_{\Phi\mathcal{U}^{\leq t}\Psi} = \{\sigma|\exists i \leq t, (S, \sigma(i), \mathbf{P}, A, L) \models \Psi \text{ et } \forall j < i, (S, \sigma(j), \mathbf{P}, A, L) \models \Phi\}$$

*avec* $\bowtie \in \{<, \leq, >, \geq\}$.

Intuitivement $\mathcal{X}_{\bowtie p}\Phi$ est vrai si la probabilité $p'$ que la prochaine transition arrive dans un état où $\Phi$ est vrai vérifie $p' \bowtie p$. La formule $\Phi\mathcal{U}^{\leq t}_{\bowtie p}\Psi$ est vraie si la probabilité $p'$ d'avoir une exécution telle que $\Phi$ est vrai à chaque instant jusqu'à ce que $\Psi$ soit vrai et cela dans un temps inférieur ou égale à $t$ vérifie $p' \bowtie p$.

On présente l'algorithmique proposé par Hans Hansson et Bengt Jonsson dans [30] pour effectuer cette vérification. Étant donnés un modèle d'interprétation $(S, s, P, A, L)$ et une formule $\Phi$, on définit un algorithme permettant d'obtenir l'ensemble des états où la formule $\Phi$ est vraie, noté $Sat(\Phi) \subseteq S$. Aussi une fois $Sat(\Phi)$ calculé il suffit de vérifier que $s \in Sat(\Phi)$.

**Définition 4.4 (Satisfaction de PCTL)** *Le calcul de* $Sat(\Phi)$ *s'opère par récurrence sur la formule* $\Phi$ *:*

- *si* $\Phi = a$ *alors* $Sat(\Phi) = \{s \in S|a \in L(s)\}$
- *si* $\Phi = \neg\Psi$ *alors* $Sat(\Phi) = S \backslash Sat(\Psi)$
- *si* $\Phi = \Psi_1 \wedge \Psi_2$ *alors* $Sat(\Phi) = Sat(\Psi_1) \cap Sat(\Psi_2)$
- *si* $\Phi = \mathcal{X}_{\bowtie p}\Psi$ *alors :*

$$Sat(\Phi) = \{s \in S| \sum_{s' \in Sat(\Psi)} \mathbf{P}(s, s') \bowtie p\}$$

- *si* $\Phi = \Psi_1\mathcal{U}^{\leq t}_{\bowtie p}\Psi_2$, *avec* $\bowtie \in \{<, \leq, \geq, >\}$ *alors* $Sat(\Phi) = Sat(\Psi_2) \cup Sat'(\Phi)$ *où* $Sat'(\Phi) \subseteq S'$ *avec* $S' = Sat(\Psi_1) \backslash Sat(\Psi_2)$ *. On définit aussi* $S'' = S \backslash (Sat(\Psi_1) \cup Sat(\Psi_2))$ *(la* figure *4.2 permet de visualiser* $S'$ *et* $S''$*),*
  - *si* $t = \infty$ *on résout le système linéaire suivant (voir l'*annexe *B.1 pour la justification mathématique de ce système linéaire) :*

$$\forall s \in S', p_s = \sum_{s' \in S'} \mathbf{P}(s, s') \times p_{s'} + \sum_{s' \in Sat(\Psi_2)} \mathbf{P}(s, s') \tag{4.1}$$

  *alors* $Sat'(\Phi) = \{s \in S'|p_s \bowtie p\}$

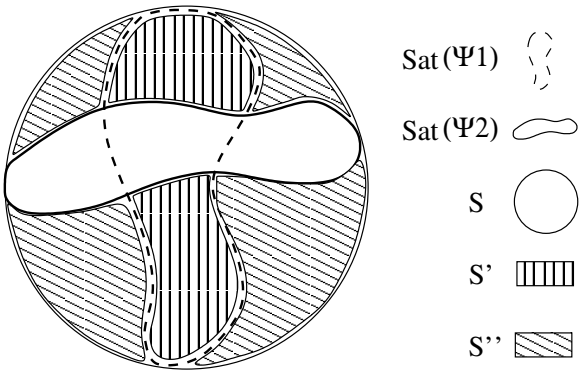



Figure 4.2: $S' = Sat(\Psi_1) \backslash Sat(\Psi_2)$ *et* $S'' = S \backslash (Sat(\Psi_1) \cup Sat(\Psi_2))$

– *si* $t \neq \infty$, *soit* $\mathbf{M}$ *la matrice* $S \times S$ *:*

$$\mathbf{M}(s,s') = \begin{cases} 1 & si\ s \in Sat(\Psi_2)\ et\ s = s' \\ \mathbf{P}(s,s') & si\ s \in S' \\ 0 & sinon \end{cases}$$

*Soit* $\pi'_{Sat(\Psi_2)}$ *le vecteur colonne de taille* $|S|$ *tel que* $\pi'_{Sat(\Psi_2)}(s) = 1$ *si* $s \in Sat(\Psi_2)$ *et 0 sinon, alors :*

$$Sat'(\Phi) = \{s \in S' | (\mathbf{M}^t \times \pi'_{Sat(\Psi_2)})(s) \bowtie p\}$$

La principale difficulté de l'algorithme concerne l'opérateur Until. On peut tout d'abord remarquer que $Sat(\Psi_2) \subseteq Sat(\Phi) \subseteq Sat(\Psi_1) \cup Sat(\Psi_2)$. En effet tout état dans lequel la formule $\Phi_2$ est vraie vérifie immédiatement la formule $\Psi_1 \mathcal{U}^{\leq t}_{\bowtie p} \Psi_2$ et tout état dans lequel ni la formule $\Phi_1$ ni la formule $\Phi_2$ ne sont vraies ne peut vérifier la formule $\Psi_1 \mathcal{U}^{\leq t}_{\bowtie p} \Psi_2$. Il ne reste donc qu'à effectuer le calcul de $Sat(\Phi)$ restreint à $S'$ qu'on note $Sat'(\Phi)$. Si $t = \infty$ il faut résoudre le système linéaire 4.1 (voir l'*annexe* B.1). Si $t \neq \infty$ on considère la matrice $\mathbf{M}$ qui définit comme aspirant[5] tout état vérifiant $\Psi_2$ et qui annule les transitions sortantes des états qui ne vérifient pas $\Psi_1$. Ainsi $\mathbf{M}^t \times \pi'_{Sat(\Psi_2)}$ correspond à la probabilité d'avoir une exécution qui, à l'instant $t$, est dans un état aspirant de $Sat(\Psi_2)$ et qui n'a visité que des états de $Sat(\Psi_1)$ avant d'atteindre cet état aspirant.

**Exemple 4.1** On peut vérifier si le modèle de la *figure* 4.1 satisfait la formule $q \Rightarrow q\mathcal{U}^{\leq 3}_{>0.7} r$. Le langage défini ne contient pas l'opérateur logique $\Rightarrow$ on considère donc la formule $\neg(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r))$. Le modèle de la *figure* 4.1 a pour état initial $s_1$ on cherche donc à vérifier si $s_1 \in Sat(\neg(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r))$. On va appliquer les règles de la *définition* 4.4 pour définir successivement $Sat(q)$, $Sat(r)$, $Sat(q\mathcal{U}^{\leq 3}_{>0.7} r)$, $Sat(\neg(q\mathcal{U}^{\leq 3}_{>0.7} r))$, $Sat(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r))$ et enfin $Sat(\neg(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r)))$ :

[5]Un état aspirant n'a qu'une transition réflexive de probabilité 1. Une fois le processus dans un état aspirant, il y reste. Nous avons choisi ici le terme aspirant simplement pour ne pas le confondre avec la définition d'état absorbant d'une CMTD donnée dans la *définition* 3.2 *page* 27.

1. $$Sat(q) = \{s_1; s_3\}$$

2. $$Sat(r) = \{s_2\}$$

3. $$Sat(q\mathcal{U}^{\leq 3}_{>0.7} r) = Sat(r) \cup \{s \in Sat(q) \backslash Sat(r) | (M^3 \times \pi'_{Sat(r)}(s) > 0.7\}$$

$$\text{avec } M = \begin{pmatrix} 0 & \frac{1}{10} & \frac{9}{10} \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \quad \text{et } \pi'_{Sat(r)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\text{On obtient par calcul } M^3 \times \pi'_{Sat(r)} = \begin{pmatrix} \frac{119}{200} \\ 1 \\ \frac{31}{40} \end{pmatrix}$$

$$\text{on en déduit que } Sat(q\mathcal{U}^{\leq 3}_{>0.7} r) = \{s_2; s_3\}$$

4. $$Sat(\neg(q\mathcal{U}^{\leq 3}_{>0.7} r)) = \{s_1\}$$

5. $$Sat(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r)) = \{s_1\}$$

6. $$Sat(\neg(q \wedge \neg(q\mathcal{U}^{\leq 3}_{>0.7} r))) = \{s_2; s_3\}$$

En conclusion notre modèle ne satisfait pas la formule $q \Rightarrow q\mathcal{U}^{\leq 3}_{>0.7} r$

**La logique CSL**

La logique CSL pour *Continuous Stochastic Logic* a été introduite par Adnan Aziz, Kumud Sanwal, Vigyan Singhal et Robert Brayton en 1996 dans l'article [1]. C'est une logique temporelle semblable à PCTL permettant de formuler des propriétés sur des processus stochastiques à temps continu. L'article cité ne propose pas d'algorithme explicitement mais prouve que la vérification de CSL sur des CMTCs est décidable. Trois années plus tard, Christel Baier, Joost-Pieter Katoen et Holger Hermanns proposent dans l'article [5] un algorithme effectuant une approximation de cette vérification. Il s'agit d'une approximation au sens où la mesure de probabilité définie par une formule est calculée avec une certaine précision. Si par exemple la précision choisie est de $\epsilon = \pm 0.1$, si la formule à vérifier est du type $\Phi\mathcal{U}^{\leq t}_{<0.45}\Psi$ et si l'approximation de la mesure de probabilité est $\mathbb{P}(\sigma_{\Phi\mathcal{U}^{\leq t}\Psi}) = 0.5$ alors l'algorithme ne sera pas en mesure de décider de sa satisfiabilité puisque $0.5 \pm 0.1$ n'est pas comparable avec 0.45. Dans ce type de situation il est alors possible de relancer l'algorithme avec une meilleur précision pour arriver, éventuellement,

à obtenir un résultat. Si la probabilité réelle à approcher est égale à 0.45 il est possible de se trouver dans une situation où la comparaison entre 0.45 et la probabilité calculée ne soit jamais possible. Donc, strictement parlant, cet algorithme, même incluant la procédure de raffinement, n'est pas décidable. Ceci n'est pas trop gênant dans la mesure où la chance de se retrouver dans un tel cas est certainement assez faible.

La logique CSL est syntaxiquement identique à la logique PCTL avec en plus l'opérateur $\mathcal{S}_{\bowtie p}(\Phi)$. La sémantique de cet opérateur est de comparer, vis à vis de la valeur $p$, la probabilité d'être dans un état qui vérifie $\Phi$ à un instant $t$ qui tend vers l'infini[6]. Voici la syntaxe complète de CSL :

$$\Phi, \Psi \ ::= \ \mathbf{vrai} \mid A \mid \neg\Phi \mid \Phi \wedge \Psi \mid \mathcal{X}_{\bowtie p}\Phi \mid \Phi \mathcal{U}^{\leq t}_{\bowtie p}\Psi \mid \mathcal{S}_{\bowtie p}(\Phi)$$

avec $\bowtie \in \{<, \leq, >, \geq\}$, $p \in \mathbb{R}, 0 \leq p \leq 1$ et $t \in \mathbb{R}_+ \cup \{\infty\}$.

Avant de donner la sémantique des opérateurs de CSL on définit sur quels modèles cette logique s'interprète.

**Définition 4.5 (Modèle d'interprétation de CSL)** *Un* modèle d'interprétation d'une formule CSL *est un quadruplet* $(S, s, \mathbf{Q}, A, L)$ *où :*

- $S$ *est un ensemble fini d'états,*
- $s \in S$ *est l'état initial (il peut aussi être considéré comme l'état courant de l'interprétation)*
- $\mathbf{Q} : S \times S \mapsto [0; 1]$ *est un générateur infinitésimal (voir la* définition *3.3),*
- $A$ *est un ensemble de propositions atomiques,*
- $L : S \mapsto \mathcal{P}(A)$ *une fonction qui assigne à chaque état l'ensemble de ses propositions atomiques vraies.*

Ici on a défini un modèle d'interprétation de CSL comme étant une CMTC avec un état initial augmentée d'une fonction d'étiquetage $L$ qui définit pour chaque état ses propositions atomiques vraies. On définit ensuite comment une formule de CSL s'interprète sur un modèle.

**Définition 4.6 (Sémantique de CSL)** *Soit* $(S, s, \mathbf{Q}, A, L)$ *un modèle de CSL,* $\Phi$ *et* $\Psi$ *deux formules de CSL. On définit la* sémantique de CSL *à l'aide de la relation* $\models$*, par induction :*

- $(S, s, \mathbf{Q}, A, L) \models \mathbf{vrai}$ *est toujours vrai.*
- $(S, s, \mathbf{Q}, A, L) \models a$ *si et seulement si* $a \in L(s)$
- $(S, s, \mathbf{Q}, A, L) \models \neg\Phi$ *si et seulement si on n'a pas* $(S, s, \mathbf{Q}, A, L) \models \Phi$*,*
- $(S, s, \mathbf{Q}, A, L) \models \Phi \wedge \Psi$ *si et seulement si :*

$$(S, s, \mathbf{Q}, A, L) \models \Phi \text{ et } (S, s, \mathbf{Q}, A, L) \models \Psi$$

[6]Le $\mathcal{S}$ de $\mathcal{S}_{\bowtie p}(\Phi)$ désigne le terme anglophone *steady* qui peut se traduire en français par stable (voir la *section* 3.2.3 pour plus de renseignements sur le comportement asymptotique d'une CMTC).

- $(S, s, \mathbf{P}, A, L) \models \mathcal{X}_{\bowtie p}\Phi$ *si et seulement si* $\mathbb{P}(Ch \mid X_0 = s) \bowtie p$, *avec :*

$$Ch = \{\sigma | \sigma(0) = s \text{ et } (S, \sigma[1], \mathbf{Q}, L) \models \Phi\}$$

- $(S, s, \mathbf{Q}, A, L) \models \Phi \mathcal{U}^{\leq t}_{\bowtie p} \Psi$ *si et seulement si* $\mathbb{P}(\sigma_{\Phi \mathcal{U}^{\leq t}\Psi} \mid X_0 = s) \bowtie p$, *avec :*

$$\sigma_{\Phi \mathcal{U}^{\leq t}\Psi} =$$

$$\{\sigma | \exists t' \leq t \ (S, \sigma@t', \mathbf{Q}, A, L) \models \Psi \text{ et } \forall t'' < t', \ (S, \sigma@t'', \mathbf{Q}, A, L) \models \Phi\}$$

*et* $t \in \mathbb{R} \cup \{\infty\}$.

- $(S, s, \mathbf{Q}, A, L) \models \mathcal{S}_{\bowtie p}(\Phi)$ *si et seulement si la probabilité* $p_\Phi$ *d'être dans un état satisfaisant* $\Phi$ *quand le temps tend vers l'infini*[7]*, vérifie* $p_\Phi \bowtie p$, *c'est-à-dire formellement :*

$$\lim_{t \to \infty} \mathbb{P}\{X_t \in Sat(\Phi) | X_0 = s\} \bowtie p$$

*avec* $\bowtie \in \{<, \leq, >, \geq\}$.

A l'exception de l'opérateur Until borné, les opérateurs de CSL ne posent pas plus de difficulté algorithmique que pour la logique PCTL. L'opérateur $\mathcal{S}_{\bowtie p}(\Phi)$ consiste à résoudre un système linéaire de taille au plus $|S|$ (voir [5] pour plus de détails), et l'opérateur Until non-borné se résout de la même manière que dans PCTL en prenant pour matrice de transition $\mathbf{P}(s, s') = \mathbf{Q}(s, s')/\mathbf{Q}(s, s)$ si $s \neq s'$ et $\mathbf{P}(s, s) = 0$. Les autres opérateurs, à l'exception du Until borné, se résolvent de la même façon que pour PCTL.

La particularité du Until borné est que sa résolution est approximative. Il peut donc entraîner des erreurs de vérification ou simplement la réponse "je ne sais pas". Toutefois en cas de non satisfaction de la réponse il est possible de relancer l'algorithme avec une meilleur précision et d'obtenir, éventuellement, une réponse définitive. La principale difficulté est de calculer la mesure de probabilité $\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t}\Psi_2} | X_0 = s)$. L'article [5] propose d'exprimer cette probabilité sous forme de l'intégrale ci-dessous (voir l'*annexe* B.2 pour la justification mathématique de cette intégrale) et de la résoudre numériquement à l'aide d'un schéma d'intégration adapté (comme celui de Romberg [59] ou de Simpson comme indiqué dans l'article) :

$$\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t}\Psi_2} | X_0 = s)$$

$$= \begin{cases} 1 & \text{si } s \in Sat(\Psi_2) \\ \displaystyle\int_0^t e^{\mathbf{Q}(s,s)t'} \times \mathbf{Q}(s, s') \times \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t - t'}\Psi_2} | X_0 = s') dt' & \text{si } s \in S' \\ 0 & \text{si } s \in S'' \end{cases}$$

où $S'$ et $S''$ sont définis à la section précédente (voir la *figure* 4.2).

[7] Cette limite est bien toujours définie, pour plus d'information le lecteur pourra se référer à l'ouvrage [51].

### 4.1.5 Model checking probabiliste statistique

Le model checking probabiliste statistique [64] a été proposé en 2002 par Håkan L.S Younes et Reid G. Simmons de l'université de Pittsburg. Leur article repose sur l'utilisation des résultats d'Abraham Wald sur l'analyse séquentielle de 1947 [61] qu'ils ont adaptés au problème du model checking probabiliste. L'idée consiste à estimer la probabilité d'une formule de chemin par des statistiques obtenues par des simulations du modèle. L'avantage d'un tel procédé est que la quantité d'états n'intervient pas dans la complexité algorithmique du processus de vérification[8]. Il devient en fait même possible de faire de la vérification sur des systèmes markoviens à quantité infinie d'états comme les processus semi-markoviens généralisés (voir la *section* 4.2.3). La seule condition est de pouvoir produire une simulation. L'inconvénient est que le résultat fourni par le model checker a une certaine probabilité d'être faux. On appelle cette probabilité le niveau de risque. Le dual du niveau de risque s'appelle le niveau de confiance : 1 - *niveau de risque*. Plus il y a de données, meilleure l'estimation est. Il est possible de connaître (ou du moins de borner inférieurement) le nombre de simulations à produire pour abaisser arbitrairement le niveau de risque.

Il existe des manières statiques[9] de connaître cette borne, par exemple en utilisant l'inégalité de Bienaymé-Tchébychev ou le théorème centrale limite. Mais, en prenant en compte le fait que, dans les problèmes de model checking, ce qui intéresse n'est pas de calculer la probabilité exacte mais seulement de positionner une probabilité par rapport à un seuil, il est possible, par des méthodes plus dynamiques, d'abaisser le nombre de simulations minimum à réaliser [61, 64]. Plus l'estimation statistique obtenue par les simulations déjà produites s'éloigne du seuil, moins il y aura besoin de nouvelles simulations pour conforter cette estimation. En revanche si l'estimation est très proche du seuil, il faudra encore d'avantage de simulations pour atteindre le niveau de confiance souhaité. Par cette méthode les auteurs de [64] arrivent à vérifier des formules du type $\phi \mathcal{U}^{<t}_{\bowtie p} \psi$ sur des modèles à quantité infinie d'états comme les processus semi-markoviens généralisés. Ni les formules de type Steady-state ni de type Until non-bornées ne peuvent se résoudre par ce type d'approche, étant donnée l'impossibilité de produire des simulations infinies. Il existe depuis 2004 un model checker statistique réalisé par Håkan L. S. Younes [63] sous licence GPL à l'adresse suivante : http ://www.cs.cmu.edu/ lorens/ymer.html.

### 4.1.6 Le model checker probabiliste PRISM

PRISM est un model checker probabiliste développé à l'université de Birmingham par D. Parker, G. Norman and M. Kwiatkowska [53, 39]. Il prend en entrée un modèle markovien et une formule CSL ou PCTL (voir la *section* 4.1.4) et vérifie si le modèle satisfait la formule. Le modèle peut être une CMTC (voir la *section* 3.2.3), une CMTD

[8] Éventuellement la quantité d'états peut intervenir dans la complexité algorithmique si elle influe sur la complexité pour produire une simulation.

[9] statiques dans le sens où la borne est fixée au départ et ne dépend pas des simulations obtenues

(voir la *section* 3.2.2) ou même un processus markovien de décision (voir la *section* 4.2.1). PRISM offre un langage permettant de décrire le modèle en terme de variables d'état et de transitions entre les états. Les modèles PEPA sont aussi acceptés par PRISM. Les logiques CSL et PCTL sont agrémentées deux opérateurs supplémentaires. Ils permettent de connaître la valeur de la mesure de probabilité d'un ensemble d'exécutions respectant un certaine propriété au lieu de comparer la probabilité de respecter la propriété par rapport à un seuil. Les deux opérateurs sont notés :

$$\Psi \, \mathcal{U}_{?}^{<t} \Phi$$

$$\mathcal{S}_{?}(\Phi)$$

où $\Psi$ et $\Phi$ sont des formules CSL ou PCTL n'utilisant pas ces opérateurs. La formule $\Psi \mathcal{U}_{?}^{<t} \Phi$ retourne la probabilité d'avoir une exécution du processus, notée $\sigma$, partant de l'état initial, et telle que la formule $\Psi$ soit vraie sur tous les états visité par $\sigma$ jusqu'à ce qu'elle rencontre un état qui vérifie $\Phi$, cela dans un temps inférieur à $t \in \mathbb{R}_{+} \cup \{\infty\}$. La formule $\mathcal{S}_{?}(\Phi)$ retourne la probabilité de se trouver dans un état qui vérifie la propriété $\Phi$ quand le temps écoulé au sien du processus tend vers l'infini.

PRISM peut s'utiliser en ligne de commande où avec une interface graphique possédant un éditeur de modèle et un éditeur de formule CSL ou PCTL, et permettant d'afficher sous forme de graphes et de courbes les résultats retournés par la procédure de model checking. Il implante aussi depuis peu (2004) des algorithmes permettant d'approcher la mesure de probabilité d'un opérateur Until borné par simulation comme expliqué dans la *section* 4.1.5.

## 4.2 Les extensions de processus Markoviens

On présente dans cette section les processus de décision Markoviens (incluant les automates temporisées probabilistes) et les processus semi-markoviens et semi-markoviens généralisés. Ces processus n'ont pas été utilisés pour construire les modèles de simulations distribuées présentés aux *chapitres* 5 et 6 mais méritent tout de même une certaine attention étant donner leur richesse d'expressivité.

### 4.2.1 Les processus de décision Markoviens

Les processus de décision markoviens (PDMs) ont été définis au départ pour faire de la planification et de la prise de décision. Ils permettent de représenter des modèles constitués à la fois de transitions non-probabilistes et de transitions probabilistes. Dans le contexte de la planification, les états du système sont en général pondérés par une valeur représentant la récompense associée à l'entrée dans cet état, l'objectif étant alors de trouver la suite d'actions permettant d'accumuler le plus de récompenses. Dans le contexte de la modélisation probabiliste de système complexes on peut voir un PDM

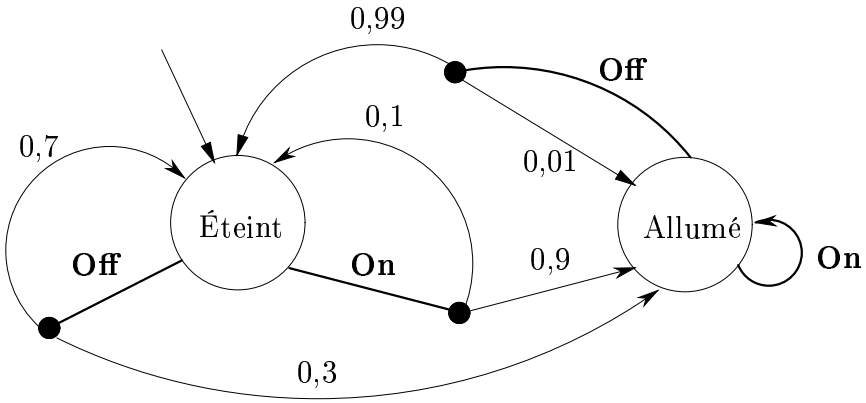



Figure 4.3: *Processus de décision markovien modélisant un interrupteur défaillant connecté à une lampe.*

comme modélisant un système probabiliste en interaction avec un environnement non probabiliste. Ici la notion de récompense n'existe pas forcément. L'environnement agit sur le système et celui-ci répond de façon probabiliste jusqu'à la prochaine action non-probabiliste de l'environnement.

La *figure* 4.3 donne en exemple un processus de décision markovien modélisant un interrupteur défaillant connecté à une lampe. Les probabilités de l'automate modélisent les défaillances de l'interrupteur et les étiquettes *On* et *Off* modélisent les actions non-probabilistes de l'environnement, celles d'un utilisateur humain par exemple. Un tel couple ne forme pas un processus stochastique. En effet les actions non-probabilistes empêchent de définir une mesure de probabilité sur les exécutions du système puisque celui-ci dépend de l'environnement de façon non probabiliste. La notion d'adversaire ou de stratégie a été définie pour permettre d'interpréter un processus de décision markovien comme un processus stochastique. Un adversaire est une fonction définissant le comportement de l'environnement de façon déterministe par rapport aux réactions du système, par exemple *On ;Off ;On ;Off...* ou n'importe quel séquence de *On* et de *Off* sont des adversaires possibles dans notre exemple. Ce type d'adversaire ne réagit pas aux réactions du système. Pour définir un adversaire dont le comportement dépend de la réaction du système on définit une fonction qui prend en entrée un préfixe de l'exécution du système, autrement dit une suite finie d'états caractérisant l'exécution depuis son initialisation jusqu'à l'instant courant (par exemple Éteint ;Allumé ;Éteint ;Allumé) et qui renvoie la prochaine action à effectuer.

Il existe des algorithmes de model checking probabiliste permettant de vérifier si un tel processus de décision markovien vérifie une propriété donnée [6] : l'aspect non probabiliste du modèle est résolu en choisissant un adversaire. Le model checker PRISM implante un algorithme opérant sur les PDMs. Il est alors possible d'évaluer une formule dans le pire ou le meilleur cas. Le pire (resp. le meilleur) cas correspond à un adversaire dont chaque action favorise l'échec (resp. la réussite) de la propriété.

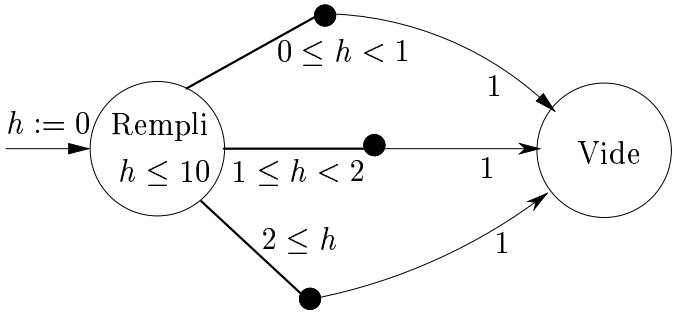



Figure 4.4: *Exemple d'un automate temporisé probabiliste modélisant un message dans un réseau*

### 4.2.2 Les automates temporisées probabilistes

Les automates temporisés probabilistes [40, 41] sont des extensions probabilistes des automates temporisés (voir la section 4.1.3). Comme les automates temporisés ils contiennent une ou plusieurs horloges dans la description de leurs états ce qui génère potentiellement un espace d'états infini. Il est alors possible d'abstraire un automate temporisé probabiliste par un PDM et la formule temporisée probabiliste, TPCTL, par une formule temporelle probabiliste PBTL [6] et ainsi d'appliquer une technique de model checking probabiliste (avec PRISM par exemple).

Il est important de noter la grande différence qui existe entre les automates temporisés probabilistes et les processus semi-markovien généralisés (PSMG) (présentés à la *section* 4.2.3). Les PSMG sont des processus stochastiques tandis que les automates temporisés probabilistes sont des processus de décision. C'est à dire que les PSMG ont un comportement entièrement probabiliste tandis que les automates temporisés probabilistes sont partiellement probabilistes. D'où vient alors la composante non probabiliste ? Des différentes gardes existantes sur les transitions de l'automate.

La *figure* 4.4 donne l'exemple d'un automate temporisé probabiliste modélisant un message dans un réseau. La première transition initialise l'horloge $h$ à zéro. L'invariant $h \leq 10$ présent dans l'état *Rempli* atteste qu'un message ne reste jamais plus de 10 secondes dans le réseau. L'aspect non probabiliste du processus réside dans le choix de l'une des trois transitions étiquetées par $0 \leq h < 1$, $1 \leq h < 2$ ou $2 \leq h$. Dans la réalité la probabilité pour un message de sortir d'un réseau dépend hautement du temps qui s'est écoulé depuis l'entrée du message dans le réseau. Pour pouvoir représenter cette évolution des probabilités au cours du temps on peut modifier l'automate en changeant les probabilités post-garde (voir la figure 4.5). Mais encore une fois le choix de la transition gardée reste non probabiliste. Il semble donc difficile d'obtenir des résultats intéressants avec cette approche car le résultat dépendra hautement de l'adversaire choisi et puisque cet adversaire n'est pas décrit de manière probabiliste il est difficile de connaître son importance. En revanche cette approche peut certainement trouver un intérêt si le système à modéliser inclut un aspect non-probabiliste, comme un ordonnanceur (dans ce cas l'ad-

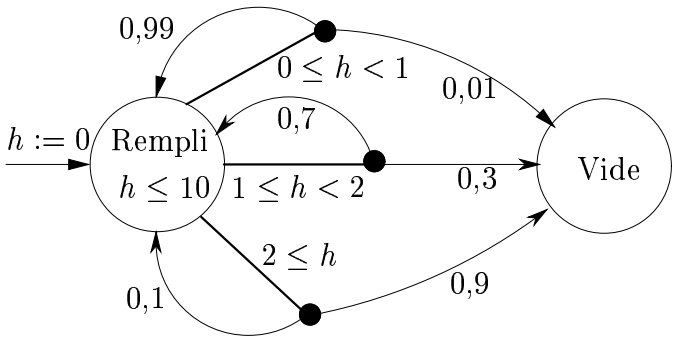



Figure 4.5: *Cette fois les chances de sortir du réseau changent en fonction de la garde choisie.*

versaire pourrait être l'algorithme de l'ordonnanceur) ou un être humain appliquant une stratégie donnée.

### 4.2.3 Les processus semi-markoviens et semi-markoviens généralisés

Il existe dans la littérature d'autres classes de processus stochastiques à temps continu plus expressifs que les processus markoviens. Deux principales extensions de processus markoviens sont à retenir : les processus *semi-markoviens* (PSM) [14, 38, 42] et *semi-markoviens généralisés* (PSMG) [35]. Toutes deux ont la particularité de pouvoir assigner des distributions non-exponentielles à leurs transitions. Pour cela la description d'une ou plusieurs horloges est ajoutée dans le processus, permettant ainsi de définir une évolution des probabilités de tirer les transitions en fonction du temps qui s'écoule dans les horloges. Les processus semi-markoviens non-généralisés possèdent une seule horloge supplémentaire dans la description de leurs états. Les processus semi-markoviens généralisés quant à eux peuvent en posséder plusieurs. Il existe un algorithme[10] de model checking sur la classe complète des processus semi-markoviens [42] et un algorithme sur une sous-classe de processus semi-markoviens généralisés [2] mais beaucoup trop restrictif puisque les formules vérifiables sont du type *"cet événement est presque sûr"* ou *"cet événement est négligeable"*. L'information sur le type de distributions choisies n'importe alors plus.

Les processus semi-markoviens, pourtant plus expressifs que les chaînes de Markov puisqu'ils possèdent une horloge alors que les chaînes de Markov n'en possèdent aucune, souffrent d'un problème supplémentaire : cette classe de processus n'est pas stable par produit synchronisé. Le produit synchronisé de deux processus semi-markoviens n'est en

[10]Il s'agit en fait d'une approximation mais du même ordre de qualité que l'algorithme proposé par C. Baier et al pour les chaînes de Markov [5]. Par contre, contrairement au chaînes de Markov, il n'existe probablement pas d'algorithme décidable non approximatif en raison de la généralité du choix offert sur la fonction de distribution.

général pas un processus semi-markovien mais un processus semi-markovien généralisé[11].

Ceci pose de gros problèmes de modélisation. Il devient très difficile de décrire un système de manière compositionnelle. Par exemple la simple mise en parallèle de deux processus semi-markoviens ne donne pas un processus semi-markovien mais semi-markovien généralisé, ce dernier nécessitant au moins deux horloges pour mémoriser les temps passés dans les états respectifs des deux processus mis en parallèle. Le model checking probabiliste sur les processus semi-markoviens généralisés est très certainement indécidable si aucune restriction n'est faite au niveau du choix des fonctions décrivant les distributions. Il est toutefois possible d'obtenir des sous-classe de PSMG décidables, la première venant en tête étant celle où les fonctions de distributions sont de type phase, auquel cas un tel PSMG correspond à une CMTC avec un nombre fini d'états (voir la *section* 3.3). Une autre solution consiste à faire du model checking statistique [64], c'est à dire à approcher la probabilité d'un événement donnée à partir de statistiques, celles-ci pouvant être obtenues par simulation de Monté-Carlo [54]. Mais dans ce cas la réponse fournie par le model checker possède un certain niveau risque, autrement dit, cette réponse a une certaine probabilité d'être fausse, voir la section 4.1.5 pour plus d'explications.

## 4.3 Outils et méthodes pour la modélisation de processus stochastique

La modélisation d'un système réel peut facilement comporter des millions d'états et de transitions. Ceci peut rendre inenvisageable de le définir en énumérant ses états et ses transitions. Pour pallier à cette difficulté il existe des algèbres et des langages de haut niveau permettant de définir un processus en terme de ses composants et de leurs interactions. PEPA [32, 33] est une algèbre permettant de définir n'importe quelle chaîne de Markov à temps continu finie. Elle a été utilisée aux *chapitres* 5 et 6 pour définir des modèles de simulations distribuées HLA. On la présente en détail dans cette section.

A la difficulté que constitue la définition d'un processus stochastique s'ajoute la difficulté de choisir les bonnes valeurs du modèle afin que celui-ci décrive au mieux le système réel. On présente à la fin de cette section l'algorithme EM pour l'approximation de distribution de phase qui apporte une première solution à cette difficulté (dans le contexte de la modélisation à l'aide de CMTCs). On verra au *chapitre* 6 comment étendre cette algorithme afin de l'utiliser sur des modèles décrit en PEPA.

### 4.3.1 Les algèbres de processus stochastiques

Les algèbres de processus stochastiques sont à la multiplication et l'addition ce que sont les processus stochastiques aux entiers. Elles permettent de décrire un processus

[11] Tout processus semi-markovien généralisé peut cependant être vu comme un processus semi-markovien ou même comme un processus de Markov. Mais ceci demande d'introduire une quantité infinie indénombrable d'états ce qui, dans le contexte du model checking, n'a pour le moment pas d'intérêt.

stochastique non plus en terme d'énumération de ses états et de ses transitions mais en terme de système constitué d'un certain nombre de composants et de leurs interactions. Les opérateurs sont le plus souvent le produit synchronisé, qui définit en quelque sorte la manière dont ses arguments interagissent, le séquencement, qui permet de définir un processus en tant que séquence d'actions ou de sous-processus, et enfin le choix qui caractérise l'indéterminisme probabiliste.

### 4.3.2 L'algèbre stochastique PEPA

PEPA pour *Performance Evaluation Process Algebra* est une algèbre de processus stochastiques introduite par Jane Hillston [32, 33] qui permet de décrire toute CMTC finie[12] en termes de composants exécutant des actions et interagissant ensemble. PEPA peut se voir comme une sorte d'extension probabiliste à temps continu des algèbres CCS pour *Calculus of Communication Systems* [45] ou CSP pour *Communication Sequential Processes* [34]. Nous allons définir la syntaxe des termes PEPA et donner une description informelle de sa sémantique.

**Définition 4.7 (Termes PEPA)** *Soit Act un ensemble d'actions.* $\tau$ *est une action particulière appelée* action invisible, *et* $\top$ *dénote un taux de transition inconnu. La syntaxe de PEPA est définie suivant les règles :*

$$P,\ Q,\ \ldots \quad ::= \quad (\alpha, r).P \ \mid\ P \underset{L}{\bowtie} Q \ \mid\ P + Q \ \mid\ P/L \ \mid\ P[l] \ \mid\ A$$

*avec la sémantique informelle suivante :*

- $(\alpha, r).P$ *est l'opérateur de* préfixe *et représente les transitions élémentaires du processus,* $\alpha$ *est un nom action.* $r \in \mathcal{R}_+ \cup \{\top\}$ *est le taux de transition et* $P$ *est la description du processus après avoir tiré la transition. Si le taux de transition appartient à* $\mathcal{R}_+$ *alors la transition est dite* active, *sinon, si son taux appartient à* $\{\top\}$ *alors elle est dite* passive.
- $P \underset{L}{\bowtie} Q$, *l'opérateur de* coopération, *est le produit synchronisé entre* $P$ *et* $Q$ *sur l'ensemble des actions de coopérations* $L \subseteq Act \backslash \{\tau\}$.
- $P + Q$ *est l'opérateur de* choix, *et signifie qu'à ce point le processus peut évoluer suivant* $P$ *ou* $Q$.
- $P/L$ *est l'opérateur d'*invisibilité, *il invalide les interactions appartenant à* $L$ *entre* $P$ *et les autres composants.*
- $A$ *est une* constante *définissable en utilisant la notation* $A \stackrel{\text{def}}{=} P$.

Un processus PEPA est défini par une modèle PEPA qui consiste en l'assemblage de termes PEPA comme défini ci-dessous.

**Définition 4.8 (Modèle PEPA)** *Un* modèle PEPA *ou* programme PEPA *est constitué d'un ensemble de déclarations de termes PEPA et d'un terme PEPA racine correspondant*

[12] et aussi une certaine classe des CMTCs à quantité d'états infinis dénombrables

*au processus à l'initialisation. Une déclaration est notée* $A \stackrel{\text{def}}{=} P$*, où* $A$ *est un identificateur pour le terme PEPA* $P$*.*

Tout modèle PEPA représente donc une certaine CMTC qui s'obtient en deux étapes :

1. La construction du *graphe de dérivations* (voir la *définition* 4.9) induit par la *Sémantique Opérationnelle* de PEPA qui est un ensemble de règles données dans un style à la Plotkin [32]. La *figure* 4.6 contient l'ensemble des règles de la sémantique opérationnelle (où la règle concernant le produit synchronisé [ou règle de *coopération*] a été simplifiée pour ne fonctionner qu'entre des transitions actives et passives).
2. L'obtention de la CMTC à partir du graphe de dérivations en effaçant les noms d'actions des transitions et en fusionnant les transitions équivalentes.

Dans le document original de Jane Hillston les coopérations peuvent avoir lieu entre n'importe quels couples de transitions (passives ou actives). Dans cette thèse, nous nous sommes restreint à des coopérations uniquement entre transitions passives et transitions actives. De plus un ensemble de transitions actives ne peuvent se synchroniser, à un instant donnée, qu'avec une seul transition passive de même action. Ceci simplifie grandement la sémantique de la coopération : le taux de la transition résultante de la coopération est égal au taux de la transition active. On aura tendance à utiliser les transitions actives pour décrire les sorties ou le comportement interne d'un processus (comportement protagoniste et spontané), et les transitions passives pour décrire les entrées d'un processus (en attente d'un signal pour réagir). La coopération entre deux transitions actives est un peu plus compliquée à définir et fait intervenir de l'arithmétique sur les taux des transitions. De plus le choix de la sémantique est un peu plus discutable que dans le cas exclusif actif/passif. C'est donc pour des raisons de simplicité que nous avons choisi cette restriction.

On définit maintenant le graphe de dérivations d'un modèle PEPA, qui sert de structure intermédiaire entre la description du processus en PEPA et sa CMTC associée.

**Définition 4.9 (Graphe de dérivations)** *Un* graphe de dérivations *est un multi-graphe orienté étiqueté* $(S, Act, s_\star, T)$ *:*

- $S$ *est l'ensemble des états du graphe, chaque élément de* $S$ *est un terme PEPA,*
- $Act$ *est un ensemble d'actions,*
- $s_\star$ *est le terme PEPA racine,*
- $T$ *est un multi-ensemble*[13] *de transitions* $\zeta : S \times S \mapsto Act \times (\mathbb{R}_+ \cup \{\top\})$

Pour faciliter les notations on définit aussi quatre fonctions d'accès aux éléments d'une transition. Supposons que $\zeta = (s, s', a, r)$ :

- $\bullet. : T \mapsto S$ dénote l'état source de la transition, $\bullet\zeta = s$.
- $.\bullet : T \mapsto S$ dénote l'état où arrive la transition, $\zeta\bullet = s'$.
- $act : T \mapsto Act$ dénote l'action de la transition, $act(\zeta) = a$.

[13] Un multi-ensemble est un ensemble pouvant contenir plusieurs occurrences du même objet.

*Préfixe*

$$\frac{}{(\alpha, r).P \xrightarrow{(\alpha,r)} P}$$

*Choix*

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P + Q \xrightarrow{(\alpha,r)} P'} \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P + Q \xrightarrow{(\alpha,r)} Q'}$$

*Coopération*

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P' \underset{L}{\bowtie} Q} \ (\alpha \notin L) \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} Q \underset{L}{\bowtie} Q'} \ (\alpha \notin L)$$

$$\frac{P \xrightarrow{(\alpha,r_1)} P' \ Q \xrightarrow{(\alpha,r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,R)} P' \underset{L}{\bowtie} Q'} \ (\alpha \in L) \ \text{ avec } R = r_1 \text{ si } r_1 \in \mathbb{R}_+ \ \wedge \ r_2 = \top, \ r_2 \text{ sinon}$$

*Invisibilité*

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\alpha,r)} P'/L} \ (\alpha \notin L) \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\tau,r)} P'/L} \ (\alpha \in L)$$

*Constante*

$$\frac{P \xrightarrow{(\alpha,r)} P'}{A \xrightarrow{(\alpha,r)} P'} \ (A \stackrel{def}{=} P)$$

Figure 4.6: *Sémantique Opérationnelle de PEPA.* Ce jeu de règle simplifié ne décrit que les coopérations entre transitions actives et passives. De plus, à un instant donnée et pour une certaine action, les transitions actives ne peuvent se synchroniser qu'avec une seule transition passive.

– $rate : T \mapsto \mathbb{R}_+$ dénote le taux de la transition, $rate(\zeta) = r$.

Si $(S, Act, s_\star, T)$ est un graphe de dérivations alors la CMTC associée à ce graphe a pour ensemble d'états $S$, pour état initial $s_\star$ et pour générateur infinitésimal $\mathbf{Q}$ défini tel que :

$$\forall s, s' \in S \quad \mathbf{Q}(s, s') = \begin{cases} \sum_{\zeta \in T, \bullet\zeta = s \wedge \zeta\bullet = s'} rate(\zeta) & \text{si } s \neq s' \text{ et } rate(\zeta) \neq \top \\ -\sum_{s'' \neq s} \mathbf{Q}(s, s'') & \text{si } s = s' \end{cases}$$

**Exemple 4.2** Exemple d'un modèle PEPA, le terme racine est *Model* et l'ensemble de déclarations est donné ci-dessous :

$$\begin{array}{rcl} Model & \stackrel{def}{=} & Begin \underset{\{\alpha\}}{\bowtie} Loop \\ Begin & \stackrel{def}{=} & (\tau, 2).\big((\alpha, v).End + (\beta, v).End + (\gamma, w).End\big) \\ & + & (\alpha, w).End \\ Loop & \stackrel{def}{=} & (\alpha, \top).Loop \\ End & \stackrel{def}{=} & (\tau, 1).End \end{array}$$

La figure 4.7 représente son graphe de dérivations. La CMTC associée à ce graphe de dérivations se trouve à la *figure* 4.13 *page* 58. Voici la correspondance entre les termes du graphe de dérivations de la *figure* 4.7 et les états de la CMTC de la *figure* 4.13 *page* 58 :

$$\begin{array}{rcl} Begin \underset{\{\alpha\}}{\bowtie} Loop & \longleftrightarrow & s_1 \\ \big((\alpha, v).End + (\beta, v).End + (\gamma, w).End\big) \underset{\{\alpha\}}{\bowtie} Loop & \longleftrightarrow & s_2 \\ End \underset{\{\alpha\}}{\bowtie} Loop & \longleftrightarrow & s_3 \end{array}$$

La transition étiquetée par $(\tau, 2)$ correspond à la transition étiquetée 2 et la transition étiquetée $(\alpha, w)$ correspond à la transition étiquetée $w$. Les trois transitions étiquetées $(\alpha, v)$, $(\beta, v)$ et $(\gamma, w)$ correspondent à la transition étiquetée $2v + w$. La transition $(\tau, 1)$ a disparu du fait de sa réflexivité. Chaque transition de ce graphe a été obtenue en appliquant les règles de la sémantique opérationnelle. Les *figures* 4.8, 4.9, 4.10, 4.11 et 4.12 donnent pour chaque transition les arbres de dérivations obtenus à partir de ces règles. Pour chaque arbre la racine (en bas) correspond à la transition déduite. Dans chacune des figures les lettres $M$, $B$, $L$ et $E$ codent respectivement pour les termes *Model*, *Begin*, *Loop* et *End*.

On va détailler étape par étape l'arbre de dérivation de la transition de la *figure* 4.8. La transition à dériver est :

$$Model \stackrel{(\alpha, w)}{\longrightarrow} End \underset{\{\alpha\}}{\bowtie} Loop$$

*Model* est une constante qu'il faut substituer par le terme qu'elle représente. On doit donc dériver la transition :

$$Begin \underset{\{\alpha\}}{\bowtie} Loop \stackrel{(\alpha, w)}{\longrightarrow} End \underset{\{\alpha\}}{\bowtie} Loop$$

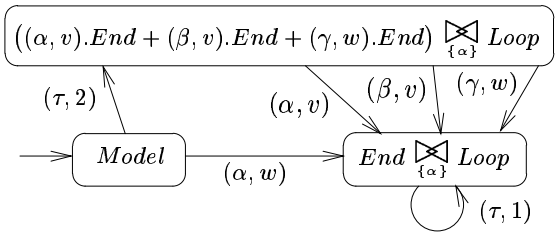



Figure 4.7: *Exemple d'un graphe de dérivations.*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\alpha,w).E \xrightarrow{(\alpha,w)} E}}{(\tau,2).\big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big)+(\alpha,w).E \xrightarrow{(\alpha,w)} E}\,(+)}{B \xrightarrow{(\alpha,w)} E}\,(\overset{def}{=}) \qquad \dfrac{\dfrac{}{(\alpha,\top).L \xrightarrow{(\alpha,\top)} L}}{L \xrightarrow{(\alpha,\top)} L}\,(\overset{def}{=})}{\dfrac{B \underset{\{\alpha\}}{\bowtie} L \xrightarrow{(\alpha,w)} E \underset{\{\alpha\}}{\bowtie} L}{M \xrightarrow{(\alpha,w)} E \underset{\{\alpha\}}{\bowtie} L}\,(\overset{def}{=})}\,(\bowtie)$$

Figure 4.8: *Arbre de dérivation de la transition* $Model \xrightarrow{(\alpha,w)} End \underset{\{\alpha\}}{\bowtie} Loop$.

On peut dériver cette transition en supposant que *Begin* admet une transition active de nom action $a$ et de taux $w$ vers *End*, et que *Loop* admet une transition passive de nom action $a$ vers lui-même. Il faut trouver un moyen de dériver les deux transitions suivantes :

$$Begin \xrightarrow{(\alpha,w)} End \qquad Loop \xrightarrow{(\alpha,\top)} Loop$$

*Begin* et *Loop* sont des constantes et définissent des termes à substituer. Il faut maintenant trouver les dérivations des deux transitions suivantes :

$$(\alpha,\top).Loop \xrightarrow{(\alpha,\top)} Loop$$

$$(\tau,2).\big((\alpha,v).End+(\beta,v).End+(\gamma,w).End\big)+(\alpha,w).End \xrightarrow{(\alpha,w)} End$$

La première transition est un axiome de la règles du *préfixe* et la seconde transition se simplifie en utilisant la règle du *choix* :

$$(\alpha,w).End \xrightarrow{(\alpha,w)} End$$

cette dernière étant aussi un axiome de la règle du *préfixe*.

### 4.3.3 L'algèbre stochastique SPADES

Nous avons vu que PEPA est une algèbre bien adaptée pour le model checking probabiliste car elle décrit des CMTCs à quantité finie d'états. Il existe en outre d'autres

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\tau,2).\big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big) \xrightarrow{(\tau,2)} (\alpha,v).E+(\beta,v).E+(\gamma,w).E}(+)}{(\tau,2).\big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big)+(\alpha,w).E \xrightarrow{(\tau,2)} (\alpha,v).E+(\beta,v).E+(\gamma,w).E}(\stackrel{def}{=})}{B \xrightarrow{(\tau,2)} (\alpha,v).E+(\beta,v).E+(\gamma,w).E}(\bowtie)}{B \underset{\{\alpha\}}{\bowtie} L \xrightarrow{(\tau,2)} \big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big) \underset{\{\alpha\}}{\bowtie} L}(\stackrel{def}{=})$$
$$M \xrightarrow{(\tau,2)} \big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big) \underset{\{\alpha\}}{\bowtie} L$$

Figure 4.9: *Arbre de dérivation de la transition Model* $\xrightarrow{(\tau,2)} \big((\alpha,v).End+(\beta,v).End+(\gamma,w).End\big) \underset{\{alpha\}}{\bowtie} Loop$.

$$\dfrac{\dfrac{\dfrac{}{(\alpha,v).E \xrightarrow{(\alpha,v)} E}}{(\alpha,v).E+(\beta,v).E+(\gamma,w).E \xrightarrow{(\alpha,v)} E}(+) \quad \dfrac{\dfrac{}{(\alpha,\top).L \xrightarrow{(\alpha,\top)} L}}{L \xrightarrow{(\alpha,\top)} L}(\stackrel{def}{=})}{\big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big) \underset{\{\alpha\}}{\bowtie} L \xrightarrow{(\alpha,v)} E \underset{\{\alpha\}}{\bowtie} L}(\bowtie)$$

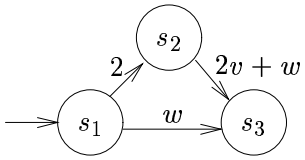
Figure 4.10: *Arbre de dérivation de la transition* $\big((\alpha,v).End+(\beta,v).End+(\gamma,w).End\big) \underset{\{\alpha\}}{\bowtie} Loop \xrightarrow{(\alpha,v)} End \underset{\{\alpha\}}{\bowtie} Loop$.

$$\dfrac{\dfrac{\dfrac{}{(\beta,v).E \xrightarrow{(\beta,v)} E}}{(\alpha,v).E+(\beta,v).E+(\gamma,w).E \xrightarrow{(\beta,v)} E}(+)}{\big((\alpha,v).E+(\beta,v).E+(\gamma,w).E\big) \underset{\{\alpha\}}{\bowtie} L \xrightarrow{(\beta,v)} E \underset{\{\alpha\}}{\bowtie} L}(\bowtie)$$

Figure 4.11: *Arbre de dérivation de la transition* $\big((\alpha,v).End+(\beta,v).End+(\gamma,w).End\big) \underset{\{\alpha\}}{\bowtie} Loop \xrightarrow{(\beta,v)} End \underset{\{alpha\}}{\bowtie} Loop$. *La dérivation de la transition* $\big((\alpha,v).End+(\beta,v).End+(\gamma,w).End\big) \underset{\{\alpha\}}{\bowtie} Loop \xrightarrow{(\gamma,w)} End \underset{\{\alpha\}}{\bowtie} Loop$ *est analogue.*

$$\dfrac{\dfrac{\dfrac{}{(\tau,1).E \xrightarrow{(\tau,1)} E}}{E \xrightarrow{(\tau,1)} .E}(\stackrel{def}{=})}{E \underset{\{\alpha\}}{\bowtie} L \xrightarrow{(\tau,1)} E \underset{\{\alpha\}}{\bowtie} L}(\bowtie)$$

Figure 4.12: *Arbre de dérivation de la transition* $End \underset{\{\alpha\}}{\bowtie} Loop \xrightarrow{(\tau,1)} End \underset{\{\alpha\}}{\bowtie} Loop$.




$$\begin{pmatrix} -2-w & 2 & w \\ 0 & -2v-w & 2v+w \\ 0 & 0 & 0 \end{pmatrix}$$

Figure 4.13: *CMTC associée au graphe de dérivations de la figure 4.7*

d'algèbres de processus stochastiques plus expressives que PEPA. SPADES [17] est une algèbre permettant de définir de façon compositionnelle des processus semi-Markoviens généralisés (PSMG) (voir la section 4.2.3), c'est-à-dire des processus stochastiques où les lois sur les transitions peuvent être non-exponentielles. Bien sûr cela inclut les lois de type phase mais également des lois non exprimables par des distributions de type phase comme par exemple la loi déterministe de temps $T$. Une telle loi signifie que la probabilité d'avoir pris la transition est 0 avant $T$, 1 après $T$. En $T$ exactement elle peut-être de 1 ou de 0 ceci étant plutôt un choix d'ordre conventionnel. Ceci étant, une telle loi peut tout à fait être approchée par une loi de type phase, par exemple par une loi de Erlang-$k$ de taux $1/T$ où $k$ tends vers l'infini (voir la *figure* 2.3 à la *page* 19). Ainsi de façon générale tout modèle décrit en SPADES peut être approché par un modèle décrit en PEPA qui pourra être manipulé par des méthodes de model checking. En revanche, la quantité d'états augmentant avec la précision de l'approximation à atteindre[14], l'explosion de l'espace d'états peut devenir assez rapidement un problème.

Étant donné qu'il n'existe pas encore d'algorithme de model checking permettant de vérifier si un modèle SPADES satisfait une formule CSL, l'algèbre SPADES est surtout utilisée pour faire de la simulation. Il existe tout de même une approche statistique permettant de faire du model checking statistique avec SPADES. En effet puisqu'un modèle SPADES décrit un PSMG il est possible de vérifier des formules de CSL (les opérateurs probabilistes sont bien sûr limités aux opérateurs bornés, le Until borné et le Next borné) en approchant la mesure de probabilité à calculer par un grand nombre de simulations (voir la section 4.1.5). L'inconvénient étant toujours que si la probabilité seuil est proche de la probabilité réelle, le nombre de simulations à effectuer peut devenir très grand. En particulier, si la propriété à vérifier énonce que tel évènement catastrophique a une probabilité rare, inférieure à 10e-9 par exemple, il faudra sans doute effectuer un nombre très important de simulations avant de valider la propriété (ce qui dépend également du niveau de confiance à atteindre, voir la section 4.1.5 à ce sujet).

[14]Cette quantité d'états peut tendre vers l'infini si le modèle SPADES contient des distributions autres que de type phase.

### 4.3.4 L'algorithme EM pour l'approximation de distributions de type phase

#### Rappels sur l'algorithme EM

L'algorithme EM [19, 62], pour *Expectation Maximisation*, est une méthode itérative permettant de trouver les paramètres d'un modèle probabiliste de sorte qu'il maximise (éventuellement localement) la vraisemblance[15] d'un certain ensemble de données. Il est utilisé quand les données, notées $\mathbf{x}$, sont composées d'observations partielles, notées $\mathbf{y}$ et qu'il n'est pas possible de maximiser par calcul direct cette vraisemblance ; contrairement par exemple à l'expérience du lancé de dés, pipés, où la maximisation peut se calculer directement[16]. L'idée consiste à estimer les variables aléatoires qui auraient permis de résoudre le problème par calcul direct (étape $E$) puis de maximiser (étape $M$ cette fois par calcul direct d'après ses estimations) la vraisemblance de $\mathbf{x}$. Il est prouvé[17] que l'étape $M$ améliore (mais ne maximise pas) la vraisemblance de $\mathbf{y}$. Ceci a pour conséquence d'améliorer le calcul de la prochaine étape $E$. Ainsi à chaque répétition des étapes $E$ et $M$ la vraisemblance de $\mathbf{y}$ augmente jusqu'à converger vers un maximum local ou global ou encore vers un point de selle de cheval. L'algorithme EM a été utilisé dans de nombreux problèmes d'apprentissage comme la reconnaissance de formes par mélange de gaussiennes, la reconnaissance de la parole avec chaînes de Markov discrète à états cachés, la classification de textes [43], et bien sûr l'approximation de distributions de type phase.

#### L'approximation de distributions de type phase

L'idée d'utiliser l'algorithme EM pour l'approximation de distributions de type phase provient de Soren Asmussen, Olle Nerman et Marita Olsson [57]. Dans ce contexte une observation complète, notée $x$, est la trajectoire d'un processus $X$ décrit par une distribution de type phase, c'est-à-dire un chemin de taille finie $m$ qui pourra se noter sous la forme :

$$s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{m-1}} s_m$$

où $s_1$ est l'état initial, $s_1$ à $s_{m-1}$ sont les états transitoires et $s_m$ est l'état absorbant. Une observation partielle de $X$, soit la variable aléatoire $Y$, se résume au temps mis par le processus pour atteindre l'état absorbant depuis l'état initial. On définit le vecteur aléatoire $\mathbf{X}$ comme le produit de $n$ processus $X$ identiques et indépendants. Une valeur de $\mathbf{X}$ consiste donc en $n$ exécutions non-observables de $X$ notées $\mathbf{x} = x_1, \ldots, x_n$. On définit le

[15] La vraisemblance correspond en général à la probabilité ou à la densité de probabilité d'observer un ensemble de données connaissant un modèle. Si la distribution sur les modèles est uniforme alors maximiser la vraisemblance des données revient à maximiser celle des modèles.

[16] Il suffit de faire le rapport pour chaque face entre son nombre d'apparitions et le nombre total de lancés. On peut montrer qu'il s'agit bien du maximum de vraisemblance en vérifiant que ce choix annule la dérivée de la vraisemblance de l'ensemble des données.

[17] La preuve est principalement une conséquence de l'inégalité de Jensen [36].

vecteur aléatoire $\mathbf{Y}$ comme le produit de $n$ variables aléatoires $Y$ identiques et indépendantes. Une valeur de $\mathbf{Y}$ consiste donc en $n$ temps d'absorption notés $\mathbf{y} = y_1, \ldots, y_n$. La connaissance de $\mathbf{x}$ pourrait permettre de maximiser par calcul direct sa vraisemblance, malheureusement on connaît seulement $\mathbf{y}$. Comme expliqué plus haut l'idée de l'algorithme EM consiste à combler la connaissance manquante par l'espérance théorique de certaines variables aléatoires et ensuite de maximiser la vraisemblance de $\mathbf{x}$, ce qui améliore (mais ne maximise pas) la vraisemblance de $\mathbf{y}$. Le nouveau modèle obtenu permet de fournir une meilleur estimation théorique des espérances et donc d'accroître à nouveau la vraisemblance de $\mathbf{y}$, et ainsi de suite.

### Définitions des Vraisemblances de x et y

La vraisemblance de $\mathbf{x}$, notée $L(\mathbf{x})$, correspond à la densité de probabilité d'avoir $\mathbf{X} = \mathbf{x}$. De même, la vraisemblance de $\mathbf{y}$, notée $L(\mathbf{y})$, correspond à la densité de probabilité d'avoir $\mathbf{Y} = \mathbf{y}$ :

$$L(\mathbf{x}) = \mathbb{P}(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{n} \mathbb{P}(X = x_i) \tag{4.2}$$

$$L(\mathbf{y}) = \mathbb{P}(\mathbf{Y} = \mathbf{y}) = \prod_{i=1}^{n} \mathbb{P}(Y = y_i) \tag{4.3}$$

La notation $\mathbb{P}$ désigne habituellement une mesure de probabilité. Les mesures de probabilité des évènements $\mathbf{X} = \mathbf{x}$, $\mathbf{Y} = \mathbf{y}$, $X = x$ ou $Y = y$ sont nulles. Étant donné que les densités de probabilité se manipulent presque de la même manière[18] que les probabilités on choisit de noter $\mathbb{P}(\mathbf{X} = \mathbf{x})$ (respectivement $\mathbb{P}(\mathbf{Y} = \mathbf{y})$ ou $\mathbb{P}(Y = y)$) comme la densité de probabilité d'avoir $\mathbf{X} = \mathbf{x}$ (respectivement d'avoir $\mathbf{Y} = \mathbf{y}$ ou $Y = y$). On donne ci-dessous la densité de probabilité d'avoir $Y = y$ (voir l'*annexe* C.1.2) :

$$\mathbb{P}(Y = y) = \frac{\partial \mathbb{P}(Y = y)}{\partial y} = \lim_{h \to 0} \frac{\mathbb{P}(y \leq Y < y + h)}{h} = \pi_{s_\star} e^{y\mathbf{T}} \mathbf{t}$$

Il est rappelé dans l'*annexe* C.1.1 que la densité de probabilité de $X = x$ avec $x = s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{m-1}} s_m$ est :

$$\begin{aligned} \mathbb{P}(X = x) = \; & \mathbf{T}(s_1, s_2) \times e^{\mathbf{T}(s_1, s_1)t_1} \\ & \times \mathbf{T}(s_2, s_3) \times e^{\mathbf{T}(s_2, s_2)t_2} \\ & \vdots \\ & \times \mathbf{t}(s_m) \times e^{\mathbf{T}(s_m, s_m)t_m} \end{aligned}$$

[18] Mais il faut être un peu plus prudent, prendre garde par exemple à respecter l'ordre de dérivé de la densité.

Si $z_s$ dénote pour tout $s \in S_{ph}$ le temps total de séjour de $x$ dans l'état $s$ et $n_{s,s'}$ pour tout $s \in S_{ph}$ et $s' \in S$ le nombre total de fois que $x$ a pris une transition de $s$ à $s'$ alors il est possible de noter la densité de probabilité de $X = x$ de la façon suivante (en utilisant le fait que l'exponentielle transforme le produit en somme et en posant comme convention $a^b = 1$ quand $b = 0$) :

$$\begin{aligned} \mathbb{P}(X = x) &= [\prod_{s \in S_{ph}} e^{\mathbf{T}(s,s) \times z_s}] \\ &\times [\prod_{\substack{s,s' \in S_{ph}, \\ s' \neq s}} \mathbf{T}(s,s')^{n_{s,s'}}] \\ &\times [\prod_{s \in S_{ph}} \mathbf{t}(s)^{n_{s,nil}}] \end{aligned} \tag{4.4}$$

Cette équation nous permet de justifier le choix des espérances à calculer pendant l'étape $E$.

**L'étape $E$**

On considère les variables aléatoires suivantes :

- Soient :
$$Z_s(\omega) = \int_0^\infty \mathbf{1}_{\{X_t(\omega)=s\}} dt$$
$|S_{ph}|$ variables aléatoires désignant chacune le temps de séjour total du processus $X$ dans l'état $s \in S_{ph}$. Puisque que $X$ possède un état absorbant $Z_s$ converge presque sûrement.
- Soient :
$$N_{s,s'}(\omega) = \lim_{\epsilon \to 0} \sum_{i=0}^{\infty} \mathbf{1}_{\{X_{i\epsilon}(\omega)=s \ \wedge \ X_{(i+1)\epsilon}(\omega)=s'\}}$$
$|S_{ph}| \times |S|$ variables aléatoires désignant chacune le nombre total de fois que la transition partant de $s \in S_{ph}$ et allant vers $s' \in S$ est tirée par le processus $X$. Pour la même raison $N_{s,s}$ converge presque sûrement.

- Soient :
$$\mathbf{Z}_s(\omega_1, \ldots, \omega_n) = \sum_{i=1}^{n} Z_s(\omega_i)$$
$|S_{ph}|$ variables aléatoires désignant chacune le temps de séjour total de tous les processus de $\mathbf{X}$ dans l'état $s \in S_{ph}$.
- Soient :
$$\mathbf{N}_{s,s'}(\omega_1, \ldots, \omega_n) = \sum_{i=1}^{n} N_{s,s'}(\omega_i)$$

$|S_{ph}| \times |S|$ variables aléatoires désignant chacune le nombre total de fois que la transition partant de $s \in S_{ph}$ et allant vers $s' \in S$ est tirée par tous les processus de $\mathbf{X}$.

L'étape $E$ consiste à calculer les espérances conditionnelles $\mathbb{E}[\mathbf{Z}_s|\mathbf{Y} = \mathbf{y}]$ et $\mathbb{E}[\mathbf{N}_{s,s'}|\mathbf{Y} = \mathbf{y}]$. Une fois ces espérances conditionnelles calculées il sera facile de maximiser la vraisemblance de $\mathbf{x}$. Dans l'article [57] Soren Asmussen, Olle Nerman et Marita Olsson proposent de résoudre ce calcul en exprimant ces espérances sous forme de systèmes d'équations différentielles ordinaires. Ils définissent pour cela $|S| + 1$ vecteurs de fonctions, $\mathbf{a}$, $\mathbf{b}$ et $(\mathbf{c}^s)_{s \in S_{ph}}$ chacun de dimension $|S_{ph}|$, défini de $\mathbb{R}_+$ dans $\mathbb{R}_+^{|S_{ph}|}$, et montrent que (voir les *annexes* C.3.1 et C.3.3) :

$$\mathbb{E}[\mathbf{Z}_s|\mathbf{Y} = \mathbf{y}] = \sum_{i=1}^{n} \frac{c_s^s(y_i)}{\pi_\star \mathbf{b}(y_i)}$$

$$\mathbb{E}[\mathbf{N}_{s,s'}|\mathbf{Y} = \mathbf{y}] = \sum_{i=1}^{n} \frac{\mathbf{Q}(s,s')}{\pi_\star \mathbf{b}(y_i)} \times \begin{cases} c_{s'}^s(y_i) & \text{si } s' \neq nil \\ a_s(y_i) & \text{si } s' = nil \end{cases}$$

avec

$\mathbf{a}(t) = \pi_\star e^{t\mathbf{T}}$

$\mathbf{b}(t) = e^{t\mathbf{T}}\mathbf{t}$

$\forall s \in S_{ph} \quad \mathbf{c}^s(t) = \int_0^t \pi_\star e^{u\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du$

On peut noter que $\mathbf{a}(t)$ définit le vecteur des probabilités de se trouver dans un état $s \in S_{ph}$ au temps $t$, et $\pi_\star \mathbf{b}(t)$ désigne la densité de probabilité de passer dans l'état absorbant $nil$ au temps $t$.

Ils considèrent ensuite $\mathbf{a}$, $\mathbf{b}$ et $(\mathbf{c}^s)_{s \in S_{ph}}$ comme étant la solution des systèmes d'équations différentielles ordinaires ci-dessous, solvables de manière efficace en utilisant la méthode de Runge-Kutta à l'ordre 4 à pas adaptatif. Pour $\mathbf{a}$ et $\mathbf{b}$ la vérification est aisée, pour $(\mathbf{c}^s)_{s \in S_{ph}}$ voir l'*annexe* C.2 :

$\dot{\mathbf{a}}(t) = \mathbf{a}(t)\mathbf{T}$ $\qquad$ $\mathbf{a}(0) = \pi_\star$

$\dot{\mathbf{b}}(t) = \mathbf{T}\mathbf{b}(t)$ $\qquad$ $\mathbf{b}(0) = \mathbf{t}$

$\forall s \in S_{ph} \quad \dot{\mathbf{c}}^s(t) = \mathbf{T}\mathbf{c}^s(t) + a_s(t)\mathbf{t}$ $\quad$ $\mathbf{c}^s(0) = 0$

**l'étape** $M$

Les espérances $\mathbb{E}[\mathbf{Z}_s|\mathbf{Y} = \mathbf{y}]$ et $\mathbb{E}[\mathbf{N}_{s,s'}|\mathbf{Y} = \mathbf{y}]$ une fois calculées il ne reste plus qu'à mettre à jour les paramètres du modèle de sorte qu'il maximise la vraisemblance de $\mathbf{x}$. Les paramètres du modèle sont tous les coefficients de la matrice $\mathbf{T}$ et du vecteur $\mathbf{t}$. Il est

facile de montrer d'après l'équation 4.4 que la maximisation de $\mathbf{x}$ s'obtient en affectant les coefficients de $\mathbf{T}$ et de $\mathbf{t}$ de la façon suivante (voir l'*annexe* C.4.1) :

$$\forall s, s' \in S_{ph}, \qquad \mathbf{T}(s, s') = \frac{\mathbb{E}[\mathbf{N}_{s,s'} | \mathbf{Y} = \mathbf{y}]}{\mathbb{E}[\mathbf{Z}_s | \mathbf{Y} = \mathbf{y}]}, \quad \mathbf{t}(s) = \frac{\mathbb{E}[\mathbf{N}_{s,nil} | \mathbf{Y} = \mathbf{y}]}{\mathbb{E}[\mathbf{Z}_s | \mathbf{Y} = \mathbf{y}]]}$$

Intuitivement cette maximisation provient du fait que chaque coefficient de $\mathbf{T}$ et de $\mathbf{t}$ correspond à un taux de transition c'est-à-dire au nombre de fois moyen que cette transition est tirée divisé par le temps d'attente moyen du processus dans l'état qui précède la transition. $\mathbf{T}$ et $\mathbf{t}$

### Schémas de l'algorithme EM

Nous donnons maintenant une description pseudo détaillée de l'algorithme $EM$ pour l'approximation de distribution de type phase :

1. `Initialiser les coefficients de` $\mathbf{T}$ `et de` $\mathbf{t}$.
2. `Jusqu'à ce qu'un critère d'arrêt soit vérifié, itérer :`
   - Étape $E$ : `pour tout` $s \in S_{ph}$, $s' \in S$ `calculer :`

$$\mathbf{z}_s := \mathbb{E}[\mathbf{Z}_s | \mathbf{Y} = \mathbf{y}], \quad \mathbf{n}_{s,s'} := \mathbb{E}[\mathbf{N}_{s,s'} | \mathbf{Y} = \mathbf{y}]$$

   - Étape $M$ : `pour tout` $s, s' \in S_{ph}$ `affecter :`

$$\mathbf{T}(s, s') := \frac{\mathbf{n}_{s,s'}}{\mathbf{z}_s}, \quad \mathbf{t}(s) := \frac{\mathbf{n}_{s,nil}}{\mathbf{z}_s}$$

3. `Retourner` $\mathbf{T}$ `et` $\mathbf{t}$.

L'initialisation des coefficients de $\mathbf{T}$ et de $\mathbf{t}$ peut jouer un rôle important dans l'issue du résultat car l'algorithme ne converge pas forcément vers le maximum global. Il est donc conseillé :

- soit de faire une bonne initialisation manuelle pour aider l'algorithme à converger vers un bon maximum local ou vers le maximum global,
- soit de faire plusieurs tentatives avec différentes initialisations aléatoires ou manuelles afin d'augmenter les chances d'obtenir un bon résultat.

Le critère d'arrêt de l'algorithme correspond en général au moment où le logarithme de la vraisemblance de $\mathbf{y}$, noté $LogL(\mathbf{y})$, n'augmente plus de façon significative. Pour cela il faut rajouter les instructions nécessaires pour calculer $LogL(\mathbf{y})$ à la fin de chaque itération mais ceci n'ajoute pas de complexité supplémentaire car cette vraisemblance est déjà quasiment calculée grâce à la fonction $\mathbf{b}$, en effet :

$$LogL(\mathbf{y}) = \sum_{i=1}^{n} log(\mathbb{P}(Y = y_i)) = \sum_{i=1}^{n} log(\pi_\star \mathbf{b}(y_i))$$

### Quelques mots sur la complexité de l'algorithme

La quasi totalité des calculs de cet algorithme se produisent pendant les étapes $E$. La complexité algorithmique du calcul des espérances conditionnelles au cours d'une itération est de l'ordre de $C \times |S|^3$ où $C$ désigne le nombre de pas nécessaires pour résoudre $\mathbf{a}$ et $\mathbf{b}$ et $(\mathbf{c}^s)_{s \in S_{ph}}$ jusqu'à atteindre la plus grande valeur de $\mathbf{y}$. En effet, la complexité pour un pas de calcul de chaque système est de l'ordre de la complexité du produit vecteur-matrice, i.e $|S|^2$, et il y a $|S|+1 \approx |S|$ systèmes, soit une complexité de $|S|^3$. Si $I$ itérations sont nécessaires pour arriver à un résultat intéressant la complexité totale du calcul est de $I \times C \times |S|^3$. Dans la pratique le nombre d'itérations $I$ se situe entre 100 et 500, et dépasse rarement le millier. La constante $C$ en revanche dépasse facilement le millier, le million voire le milliard. Pour cette raison il est actuellement difficile d'appliquer cet algorithme sur des modèles dépassant les quelques centaines d'états.

# Chapter 5

# Évaluer les Performances d'une Simulation HLA par le Model Checking Probabiliste

Dans ce chapitre est exposé un exemple d'utilisation du model checking probabiliste pour l'évaluation de performances d'une simulation HLA temps réel. Le model construit ici manque de réalisme et ne répond que partiellement au problème posé par la thèse. Ce chapitre a donc pour objectif de montrer que l'approche évaluation de performances par le model checking probabiliste est d'une part faisable et d'autre part intéressante (si on dispose d'un modèle plus réaliste). Le modèle considéré est une simulation HLA avec trois fédérés et un intergiciel disposés sur quatre ordinateurs reliés par un réseau, une liaison full-duplex[1]. Les contraintes de performances à vérifier sont :

1. Une deadline de probabiliste des temps d'arrivées des messages de mise à jour.
2. La probabilité de perdre des messages pendant une certaine durée de fonctionnement.

Le langage choisi pour construire la modèle est le langage algébrique PEPA (voir la section 4.3.2) et la logique CSL (voir la section 4.1.4) pour formaliser les contraintes de performances. L'outil qui a été utilisé est le model checker probabiliste PRISM (voir la section 4.1.6).

## 5.1 Première tentative de modélisation d'une simulation HLA

Dans cette section on présente les spécifications informelles et formelles, en PEPA, d'une simulation distribuée HLA à 3 fédérés. Ce modèle bien que, dans cet exemple,

[1]Une partie de cette modélisation comprenant la formalisation de la contrainte de performance de type deadline probabiliste est présente dans l'article [27]

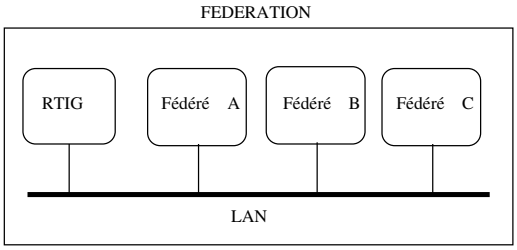



Figure 5.1: *Une Simulation Distribuée HLA.*

limité à 3 trois fédérés, est facilement modifiable pour représenter une nombre quelconque de fédérés (sans compter bien sûr le problème de l'explosion du nombre d'états).

### 5.1.1 Spécification du système modélisé

L'architecture matérielle du système consiste en 4 ordinateurs et un réseau local de type LAN. La fédération est composée de trois fédérés et d'un intergiciel (RTIG) placés chacun sur un ordinateur. La *figure* 5.1 donne un schémas du système à modéliser où chaque boîte correspond à un ordinateur.

Le modèle décrit l'interaction de trois composants. Chaque composant est modélisé par un ensemble de termes PEPA et la modélisation entière est donnée par la synchronisation de ces trois composants. Ceux sont :

1. Trois *fédérés* qui calculent la simulation.
2. Le *RTIG* (l'intergiciel qui réalise les communications entre les fédérés) dont le rôle est de décider vers quels fédérés répartir les différents messages.
3. Le *réseau* qui transporte les messages entre les fédérés et le RTIG.

Chaque composant possède des entrées et des sorties. Par convention chaque nom d'action du modèle PEPA représentant une entrée (resp. une sortie) commencera par $E$ (resp. $S$). Il est important de noter que les noms d'actions sont locaux à chaque composant. Les communications entre les différents composants sont alors représentées en les renommant de façon adéquate avant l'utilisation du produit synchronisé. Ceci représente en fait le branchement des entrées d'un composant sur les sorties d'un autre et réciproquement. Dans le modèle, une transition étiquetée par une action correspondant à une entrée sera toujours une transition passive, c'est-à-dire de taux inconnu, noté $\top$. Réciproquement une transition étiquetée par une action correspondant à une entrée sera toujours une transition active, c'est-à-dire de taux connu.

#### Les fédérés

Chaque fédéré avance à son propre rythme. Un fédéré calcule l'évolution de sa simulation pendant un cycle de calcul. A la fin de chaque cycle il envoie des messages de mise à

jour au RTIG pour que celui-ci les renvoie aux fédérés concernés. Pour indiquer la provenance d'un message de mise à jour le nom d'action associé à l'interaction de ce message contiendra le caractère $a$, $b$ ou $c$ pour désigner son fédéré émetteur A, B ou C.

A tout moment durant son cycle il peut recevoir des mises à jour d'autres fédérés. Il n'attend donc pas de recevoir des mises à jour pour continuer son travail. On veut pouvoir mesurer le temps de traversée du réseau et du RTIG au cours d'une activité normale, c'est-à-dire avec une charge réseau et RTIG normales. Ceci implique, entre autre, que plusieurs messages de mise à jour d'un même émetteur peuvent être présents en même temps dans le système en différent points. Par exemple, pendant que le fédéré $A$ envoie un message de mise à jour sur le réseau, un autre message de mise à jour, précédemment envoyé par $A$, peut se trouver en phase de traitement dans le RTIG. Afin de mesurer correctement les temps de traversée de ces messages il est nécessaire que le modèle fasse la distinction entre eux. Dans la pratique, ce genre de distinction s'opère en étiquetant les messages par des identificateurs uniques, un identificateur possible pouvant être la date d'émission du message. Dans le modèle, nous n'allons pas pouvoir ajouter des identificateurs uniques par message pour des raisons évidentes d'explosion de l'espace d'états. Aussi distingue-t-on entre deux types de messages :

1. les messages servant à mesurer le temps de traversée. On les distingue dans le modèle par l'ajout du symbole $_\uparrow$ à coté du symbole codant la provenance du message,
2. les autres messages servant simplement à encombrer le réseau et le RTIG afin de modéliser leurs charges.

On ajoute une autre hypothèse : *un message servant à mesurer le temps de traversée sera toujours seul dans le réseau durant son existence.* Ainsi a-t-on la garantie de ne pas le confondre avec un autre message. On verra qu'il est possible de vérifier la validité de cette hypothèse simplement par model checking (voir la *section* 5.2).

On peut alors écrire une formule CSL qui mesure le temps de transmission d'un message du fédéré source jusqu'aux fédérés destination pendant une activité normale du système. Dans la modélisation, il apparaît que les messages étiquetés par $_\uparrow$ doivent être envoyés tous les six cycles pour garantir leur unicité durant leur voyage (voir la *section* 5.2 pour la preuve).

Dans la formalisation en PEPA du modèle chaque fédéré est le résultat d'un produit parallèle[2] entre deux processus :

- $EnvFedX$ qui décrit le fait qu'à chaque fin de cycle le fédéré $X$ envoie un message de mise à jour au RTIG.
- $FedXRec$ qui décrit le fait que pendant son temps de cycle le fédéré $X$ reçoit des mises à jour des autres fédérés.

[2]C'est-à-dire un produit synchronisé avec comme ensemble d'actions de synchronisation l'ensemble vide.

$$
\begin{array}{lcl}
FedA & \stackrel{def}{=} & EnvFedA \underset{\emptyset}{\bowtie} FedARec \\
EnvFedA & \stackrel{def}{=} & \underbrace{(Sa, \lambda_A) \ldots (Sa, \lambda_A)}_{5}.EnvFedA_{\uparrow} \\
EnvFedA_{\uparrow} & \stackrel{def}{=} & (Sa_{\uparrow}, \lambda_A).EnvFedA \\
FedARec & \stackrel{def}{=} & (Eb, \top).FedARecb \\
 & + & (Eb_{\uparrow}, \top).FedARecb_{\uparrow} \\
 & + & (Ec, \top).FedARecc \\
 & + & (Ec_{\uparrow}, \top).FedARecc_{\uparrow}
\end{array}
$$

$FedARecb$, $FedARecb_{\uparrow}$, $FedARecc$ , $FedARecc_{\uparrow}$ ont exactement la même définition que $FedARec$.

Table 5.1: Définition du modèle $FedA$

| | | |
|---|---|---|
| $Sa$ | : | envoie une mise à jour de $A$ vers le réseau, |
| $Sa_{\uparrow}$ | : | envoie une mise à jour (valable pour faire une mesure) de $A$ vers le réseau, |
| $Eb$ | : | reçoit une mise à jour de $B$ depuis le réseau, |
| $Eb_{\uparrow}$, $Ec$, $Ec_{\uparrow}$ | : | reçoivent des mises à jours, |
| $\lambda_A$ | : | correspond à la fréquence du cycle de $A$, habituellement cette fréquence se situe autour de 40Hz. |

Table 5.2: Signification des symboles d'entrées/sorties et de taux de $FedA$

Les termes PEPA définissant le fédéré $A$ sont listés dans la *Table* 5.1. Le terme désignant le fédérés $A$ s'appelle $FedA$. $EnvFedA \underset{\emptyset}{\bowtie} FedARec$ modélise le produit parallèle entre $EnvFedA$ et $FedARec$. La séquence de 5 préfixes dans le terme $EnvFedA$ avant de déclencher le terme $EnvFedA_{\uparrow}$ signifie que le fédéré doit envoyer 5 messages non marqués avant d'envoyer un message marqué. Le but de $FedARec$ est de mémoriser la dernière mise à jour reçue par le fédéré $A$. Le modèle ne représente pas le contenu d'un message mais seulement sa taille, sa provenance et éventuellement la marque $_{\uparrow}$.
La *table* 5.2 contient les détails concernant ses entrées/sorties et les taux de transitions.

Les termes PEPA définissant le fédéré $B$ sont similaires aux termes définissant le fédéré $A$ modulo un renommage des lettres $A$ par $B$, $a$ par $b$, $b$ par $a$ et $c$ par $c$. Les renommages à effectuer pour obtenir le fédéré $C$ suivent la même logique.

### Le RTIG

Le rôle du RTIG est de répartir correctement les messages de mise à jour d'un fédéré vers les fédérés concernés. Dans ce modèle le RTIG effectue une diffusion générale des mise à jour mais est aisément modifiable pour modéliser une diffusion sélective. Le modèle de RTIG ne contient aucun buffer, il peut seulement traiter un message à la fois. Les mises à jour venant des fédérés sont amenées au RTIG par le réseau et ce dernier les retransmet

| | | |
|---|---|---|
| $RTIG$ | $\stackrel{def}{=}$ | $(Ea, \top).Repartir\_a$ |
| | $+$ | $(Ea_{\uparrow}, \top).Repartir\_a_{\uparrow}$ |
| | $+$ | $(Eb, \top).Repartir\_b$ |
| | $+$ | $(Eb_{\uparrow}, \top).Repartir\_b_{\uparrow}$ |
| | $+$ | $(Ec, \top).Repartir\_c$ |
| | $+$ | $(Ec_{\uparrow}, \top).Repartir\_c_{\uparrow}$ |
| $Repartir\_a$ | $\stackrel{def}{=}$ | $(SaB, \lambda_{RTIG}).(SaC, \lambda_{RTIG}).RTIG$ |
| | $+$ | $(SaC, \lambda_{RTIG}).(SaB, \lambda_{RTIG}).RTIG$ |

Table 5.3: Définition du terme PEPA $RTIG$

| | | |
|---|---|---|
| $Ea, \ldots, Ec_{\uparrow}$ | : | le RTIG reçoit une mise à jour du réseau, |
| $SaC, \ldots, Sc_{\uparrow}B$ | : | le RTIG renvoie une mise à jour vers le réseau pour les fédérés $A$, $B$ or $C$ |
| $\lambda_{RTIG}$ | : | taux avec lequel un message est traité, la valeur du taux dépend de la vitesse de la machine hébergeant le RTIG. |

Table 5.4: Définition des entrées/sorties est du taux utilisé dans le terme PEPA $RTIG$

aux autres fédérés concernés (à savoir ceux qui ont souscrit à ces mise à jours) à nouveau par le réseau. La *table* 5.3 contient la définition des termes $RTIG$ et $Repartir\_a$. La *table* 5.4 contient la sémantique des noms d'actions et des taux de transitions du terme $RTIG$.

Si le RTIG est disponible et si une mise à jour de $A$, $B$ ou $C$ est sur le réseau alors il prend cette mise à jour du réseau et la renvoie aux fédérés concernés. Le processus défini par le terme $Repartir\_a$ renvoie la mise à jour $a$ vers les fédérés $B$ et $C$. Les définitions de $Repartir\_a_{\uparrow}$, $Repartir\_b$, $Repartir\_b_{\uparrow}$, $Repartir\_c$, $Repartir\_c_{\uparrow}$ suivent le même principe. Il est possible de modifier les termes $Repartir\_x$ où $x \in \{a, b, c, a_{\uparrow}, b_{\uparrow}, c_{\uparrow}\}$ de façon à choisir les fédérés concernés par les mise à jour. Par exemple les deux termes ci-dessous modélisent le fait que le RTIG renvoie les mise à jour de $A$ uniquement vers $C$ :

$$\begin{aligned} Repartir\_a &\stackrel{def}{=} (SaC, \lambda_{RTIG}).RTIG \\ Repartir\_a_{\uparrow} &\stackrel{def}{=} (Sa_{\uparrow}C, \lambda_{RTIG}).RTIG \end{aligned}$$

### Le Réseau

Le réseau est de type LAN (Local Area Network). Il s'agit très simplement d'un lien full duplex connecté aux machines qui hébergent les fédérés et la machine qui héberge le RTIG. Les communications partant des fédérés vers le RTIG sont assurées par une moité du lien full duplex, et les communications partant du RTIG pour les fédérés sont assurées par la second moitié du lien. Dans cette section on fait l'hypothèse d'un réseau parfait

ne perdant aucun message. On présentera une variante du modèle de ce réseau dans la *section* 5.2.3 où des pertes de messages seront possibles.

Les distributions probabilistes qui décrivent la probabilité pour un message de sortir d'un réseau en fonction du temps ne sont pas de type exponentiel. En sondant des réseaux réels, LAN ou Internet on a constaté que ces distributions ressemble davantage à des distributions de Erlang ou mieux encore, des distributions de Cox (voir la section 3.3 pour les détails concernant les distributions de Erlang et de Cox). Bien que les distributions de Cox soient plus réalistes que les distributions de Erlang elle sont aussi plus difficiles à paramétrer, ce qui motive le choix pour cette première modélisation d'une distribution de Erlang pour modéliser la distribution du délai de transmission du réseau.

Le réseau résulte d'une composition parallèle de deux processus :

1. *LANF2R* qui transporte les mises à jour depuis les fédérés vers le RTIG,
2. *LANR2F* qui transporte les mises à jour depuis le RTIG vers les fédérés.

Les termes commençant par *Erlang-k* ont été utilisés pour modéliser une distribution de type Erlang-k. La *table* 5.5 contient la définition du terme PEPA *Reseau* et la *table* 5.6 la sémantique des entrées/sorties des termes et des taux.

#### Le modèle complet de la simulation distribuée

Le modèle complet, dénoté par le terme PEPA *SimulDist*, est obtenu par la composition des fédérés, du RTIG et du réseau. La plus grosse partie de la définition concerne en fait le renommage des noms d'actions des fédérés et du RTIG de manière à les connecter au réseau. La *table* 5.7 contient les définitions des termes PEPA modélisant le système au complet.

L'ensemble des étiquettes $\mathcal{I}$ contient toutes les interactions impliquées dans les communications entre les fédérés et le réseau. $\mathcal{J}$ contient toutes les interactions impliquées dans les communications entre le RTIG et le réseau, voir la *table* 5.8 pour leurs définitions.

## 5.2 Formalisation des contraintes de performances en CSL

Cette section montre comment utiliser CSL (voir la *section* 4.1.4) pour formaliser deux contraintes de performances sur le modèle :

1. Une deadline probabiliste du temps de transmission d'un message de mise à jour entre deux fédérés, ceci est présenté à la *section* 5.2.2.

| | | |
|---|---|---|
| $Reseau$ | $\stackrel{def}{=}$ | $LANF2R \underset{\emptyset}{\bowtie} LANR2F$ |
| $LANF2R$ | $\stackrel{def}{=}$ | $(Ea2R, \top).Erlang\text{-}k\_a$ |
| | $+$ | $(Ea_{\uparrow}2R, \top).Erlang\text{-}k\_a_{\uparrow}$ |
| | $+$ | $(Eb2R, \top).Erlang\text{-}k\_b$ |
| | $+$ | $(Eb_{\uparrow}2R, \top).Erlang\text{-}k\_b_{\uparrow}$ |
| | $+$ | $(Ec2R, \top).Erlang\text{-}k\_c$ |
| | $+$ | $(Ec_{\uparrow}2R, \top).Erlang\text{-}k\_c_{\uparrow}$ |
| $Erlang\text{-}k\_a$ | $\stackrel{def}{=}$ | $\underbrace{(\tau, k\lambda_a) \ldots (\tau, k\lambda_a)}_{k-1}.(Sa2R, k\lambda_a).LANF2R$ |
| $LANR2F$ | $\stackrel{def}{=}$ | $(Ea2B, \top).Erlang\text{-}k\_RaFedB$ |
| | $+$ | $(Ea_{\uparrow}2B, \top).Erlang\text{-}k\_Ra_{\uparrow}FedB$ |
| | $+$ | $(Ea2C, \top).Erlang\text{-}k\_RaFedC$ |
| | $+$ | $(Ea_{\uparrow}2C, \top).Erlang\text{-}k\_Ra_{\uparrow}FedC$ |
| | $+$ | $(Eb2A, \top).Erlang\text{-}k\_RbFedA$ |
| | $+$ | $(Eb_{\uparrow}2A, \top).Erlang\text{-}k\_Rb_{\uparrow}FedA$ |
| | $+$ | $(Eb2C, \top).Erlang\text{-}k\_RbFedC$ |
| | $+$ | $(Eb_{\uparrow}2C, \top).Erlang\text{-}k\_Rb_{\uparrow}FedC$ |
| | $+$ | $(Ec2A, \top).Erlang\text{-}k\_RcFedA$ |
| | $+$ | $(Ec_{\uparrow}2A, \top).Erlang\text{-}k\_Rc_{\uparrow}FedA$ |
| | $+$ | $(Ec2B, \top).Erlang\text{-}k\_RcFedB$ |
| | $+$ | $(Ec_{\uparrow}2B, \top).Erlang\text{-}k\_Rc_{\uparrow}FedB$ |
| $Erlang\text{-}k\_RaFedB$ | $\stackrel{def}{=}$ | $\underbrace{(\tau, k\lambda_a) \ldots (\tau, k\lambda_a)}_{k-1}.(Sa2B, k\lambda_a).LANR2F$ |

Table 5.5: Définition du terme PEPA *Reseau*

| | | |
|---|---|---|
| $Ea2R, \ldots, Ec_{\uparrow}2R$ | : | met dans le réseau une mise à jour depuis un fédéré vers le RTIG. |
| $Sa2R, \ldots, Ec_{\uparrow}2R$ | : | sort du réseau une mise à jour depuis un fédéré vers le RTIG. |
| $Ea2B, \ldots, Ec_{\uparrow}2B$ | : | met dans le réseau une mise à jour depuis le RTIG vers un fédéré. |
| $Sa2B, \ldots, Sc_{\uparrow}2B$ | : | sort du réseau une mise à jour depuis le RTIG vers un fédéré. |
| $\lambda_a$ | : | taux avec lequel le message $a$ traverse un aller de réseau, cette valeur dépend de la taille du message, du débit et de la latence du réseau. |
| $k$ | : | nombre d'étages de la distribution de Erlang et cette grandeur influe sur l'écart type de la distribution. |

Table 5.6: Définition des entrées, sorties et taux du terme PEPA *Reseau*

| | | |
|---|---|---|
| $SimulDist$ | $\stackrel{def}{=}$ | $(FederesConnect \underset{\mathcal{I}}{\bowtie} Reseau \underset{\mathcal{J}}{\bowtie} RTIGConnect)$ |
| $FederesConnect$ | $\stackrel{def}{=}$ | $FedA[Sa \leftarrow Ea2R, Sa_{\uparrow} \leftarrow Ea_{\uparrow}2R, Eb \leftarrow Sb2A,$ $Eb_{\uparrow} \leftarrow Sb_{\uparrow}2A, Ec \leftarrow Sc2A, Ec_{\uparrow} \leftarrow Sc_{\uparrow}2A]$ |
| | $\underset{\emptyset}{\bowtie}$ | $FedB[Sb \leftarrow Eb2R, Sb_{\uparrow} \leftarrow Eb_{\uparrow}2R, Ea \leftarrow Sa2B,$ $Ea_{\uparrow} \leftarrow Sa_{\uparrow}2B, Ec \leftarrow Sc2B, Ec_{\uparrow} \leftarrow Sc_{\uparrow}2B]$ |
| | $\underset{\emptyset}{\bowtie}$ | $FedC[Sc \leftarrow Ec2R, Sc_{\uparrow} \leftarrow Ec_{\uparrow}2R, Ea \leftarrow Sa2C,$ $Ea_{\uparrow} \leftarrow Sa_{\uparrow}2CEb \leftarrow Sb2C, Eb_{\uparrow} \leftarrow Sb_{\uparrow}2C]$ |
| $RTIGConnect$ | $\stackrel{def}{=}$ | $RTIG[Ea \leftarrow Sa2R, Ea_{\uparrow} \leftarrow Sa_{\uparrow}2R, Eb \leftarrow Sb2R,$ $Eb_{\uparrow} \leftarrow Sb_{\uparrow}2R, Ec \leftarrow Sc2R, Ec_{\uparrow} \leftarrow Sc_{\uparrow}2R,$ $SaB \leftarrow Ea2B, Sa_{\uparrow}B \leftarrow Ea_{\uparrow}2B, SaC \leftarrow Ea2C,$ $Sa_{\uparrow}C \leftarrow Ea_{\uparrow}2C, SbA \leftarrow Ea2A, Sb_{\uparrow}A \leftarrow Ea_{\uparrow}2A,$ $SbB \leftarrow Ea2B, Sb_{\uparrow}B \leftarrow Ea_{\uparrow}2B, ScA \leftarrow Ea2A,$ $Sc_{\uparrow}A \leftarrow Ea_{\uparrow}2A, ScB \leftarrow Ea2B, Sc_{\uparrow}B \leftarrow Ea_{\uparrow}2B]$ |

Table 5.7: Définition du modèle complet

$$
\begin{array}{rcl}
\mathcal{I} & = & \{Ea2R, Ea_{\uparrow}2R, Eb2R, Eb_{\uparrow}2R, Ec2R, Ec_{\uparrow}2R, \\
 & & Sa2B, Sa_{\uparrow}2B, Sa2C, Sa_{\uparrow}2C, \\
 & & Sb2A, Sb_{\uparrow}2A, Sb2C, Sb_{\uparrow}2C, \\
 & & Sc2A, Sc_{\uparrow}2A, Sc2B, Sc_{\uparrow}2B\}
\end{array}
$$

$\mathcal{J}$ est similaire à $\mathcal{I}$ mais la lettre $E$ est substituée par la lettre $S$
et la lettre $S$ est substituée par la lettre $E$ dans chacun des noms d'actions.

$$
\begin{array}{rcl}
\mathcal{J} & = & \{Sa2R, Sa_{\uparrow}2R, Sb2R, Sb_{\uparrow}2R, Ec2R, Sc_{\uparrow}2R, \\
 & & Ea2B, Ea_{\uparrow}2B, Ea2C, Ea_{\uparrow}2C, \\
 & & Eb2A, Eb_{\uparrow}2A, Eb2C, Eb_{\uparrow}2C, \\
 & & Ec2A, Ec_{\uparrow}2A, Ec2B, Ec_{\uparrow}2B\}
\end{array}
$$

Table 5.8: Définition des ensembles de coopérations $\mathcal{I}$ et $\mathcal{J}$

2. Une contrainte permettant de vérifier que la perte d'un message pendant une certaine durée d'exécution ne dépasse pas une certaine probabilité, et cela est présenté à la *section* 5.2.3.

On compare ensuite, à la *section* 5.3, des résultats obtenus avec le model checker PRISM sur le modèle et sur la première contrainte de performance, la deadline probabiliste, avec des données tirées d'une simulation HLA réelle. Cette comparaison permet de conclure sur la validité d'une telle démarche à condition d'avoir un modèle réaliste de la simulation distribuée.

### 5.2.1 Interpréter un modèle PEPA sur une formule CSL

Avant de formaliser des contraintes de performances en CSL nous allons définir comment un modèle PEPA s'interprète sur une formule CSL. Comme défini à la *section* 4.1.4 *page* 43 un modèle d'interprétation sur une formule CSL est une CMTC munie d'un ensemble de propositions atomiques $\mathcal{A}$ et d'une fonction $\mathcal{L}$ qui assigne à chaque état de la CMTC un sous ensemble de $\mathcal{A}$ identifiant l'ensemble des propositions atomiques vraies de cet état.

Dans ce cas présent la CMTC en question est la CMTC associée au modèle PEPA. Et l'ensemble des propositions atomiques du modèle est l'ensemble des constantes PEPA défini à l'aide de $\stackrel{def}{=}$. Enfin, la fonction d'assignation $\mathcal{L}$ est définie telle que chaque état (rappelons que les états de la CMTC associée sont également des termes PEPA) contient une constante de $\mathcal{A}$ sans préfixe. Ce qui formellement se traduit par :

$$
\mathcal{L}(P) = \begin{cases}
\{P\} \cup \mathcal{L}(Q) & \text{si } P \stackrel{def}{=} Q \\
\mathcal{L}(Q) \cup \mathcal{L}(R) & \text{si } P = Q + R \\
\mathcal{L}(Q) \cup \mathcal{L}(R) & \text{si } P = Q \underset{\mathcal{L}}{\bowtie} R \\
\mathcal{L}(Q) & \text{si } P = Q/\mathcal{L} \\
\mathcal{L}(Q) & \text{si } P = Q[l]
\end{cases}
$$

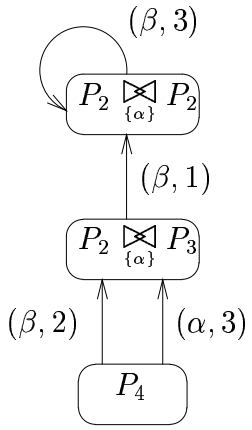



Figure 5.2: *Graphe de dérivation de l'exemple 5.1*

**Exemple 5.1** Soit le modèle PEPA définit par l'ensemble des déclarations suivantes :

$$\begin{aligned} P_1 &\stackrel{def}{=} (\alpha, \top).P_2 + (\beta, 2).P_3 \\ P_2 &\stackrel{def}{=} (\alpha, 3).P_3 \\ P_3 &\stackrel{def}{=} (\beta.1).P_2 \\ P_4 &\stackrel{def}{=} P_1 \underset{\{\alpha\}}{\bowtie} P_2 \end{aligned}$$

et de terme racine $P_4$.
L'ensemble des propositions atomiques du modèle d'interprétation est :

$$\mathcal{A} = \{P_1, P_2, P_3, P_4\}$$

L'ensemble des états de la CMTC associée est (voir la *figure* 5.2 pour le graphe de dérivations de ce modèle) :

$$S = \{P_4, P_2 \underset{\{\alpha\}}{\bowtie} P_3, P_2 \underset{\{\alpha\}}{\bowtie} P_2\}$$

Enfin la fonction d'assignation $\mathcal{L}$ est la suivante :

$$\begin{aligned} \mathcal{L}(P_4) &= \{P_4, P_1, P_2\} \\ \mathcal{L}(P_2 \underset{\{\alpha\}}{\bowtie} P_3) &= \{P_2, P_3\} \\ \mathcal{L}(P_2 \underset{\{\alpha\}}{\bowtie} P_2) &= \{P_2\} \end{aligned}$$

Ce qui signifie littéralement que dans l'état $P_4$ les propositions atomiques vraies sont $P_4$, $P_1$ et $P_2$, ...

### 5.2.2 La deadline probabiliste

Voici l'exemple d'une contrainte intéressante à vérifier pour un concepteur de simulation : *"Le temps de transmission d'un message de mise à jour d'un fédéré vers un autre*

*fédéré est de moins de 1ms avec une probabilité d'au moins 0.9".*

Pour formaliser une telle contrainte on utilise des messages de type $_{\uparrow}$ et l'on écrit formellement en CSL que le temps écoulé entre l'entrée d'un message de type $_{\uparrow}$ dans le réseau en direction du RTIG et sa sortie du réseau vers le fédéré destination est inférieur à 1ms avec une probabilité supérieur à 0.9. Les fédérés considérés dans l'exemple sont $A$ pour le fédéré source et $C$ pour le fédéré destination. Voici ci-dessous la formule CSL formalisant cette deadline probabiliste :

$$\textit{Erlang-k\_a}_{\uparrow} \Rightarrow (true\ \mathcal{U}^{<0.001}_{>0.9} FedCReca_{\uparrow})$$

Le terme $\textit{Erlang-k\_a}_{\uparrow}$ est vrai quand un message de mise à jour marqué par $_{\uparrow}$ et émis par le fédéré $A$ rentre dans le réseau. Le terme $FedCReca_{\uparrow}$ est vrai quand une mise à jour marquée par $_{\uparrow}$ de $A$ est reçue par le fédéré $C$.

Avant de vérifier cette formule on s'assure qu'il n'y a effectivement qu'un seul message de mise à jour de $A$ marqué par $_{\uparrow}$ dans le réseau au même instant. On formalise cela par la formule CSL ci-dessous que nous vérifions avec PRISM :

$$\begin{aligned} \textit{Erlang-k\_a}_{\uparrow} \quad \Rightarrow \quad & \neg \textit{Repartir\_a}_{\uparrow} \\ & \wedge \neg \textit{Erlang-k\_Ra}_{\uparrow} FedB \\ & \wedge \neg \textit{Erlang-k\_Ra}_{\uparrow} FedC \\ & \wedge \neg FedBReca_{\uparrow} \wedge \neg FedCReca_{\uparrow} \end{aligned}$$

Ce qui se traduit, en langage naturel, par : *Supposons qu'un message marqué par $_{\uparrow}$ et émis par A rentre dans le réseau alors aucun message de ce type, i.e. marqué par $_{\uparrow}$ et émis par A, n'est présent : ni dans le RTIG, ni dans le réseau en direction de B, ni dans le réseau en direction de C, ni dans les fédérés B ou C.*

### 5.2.3 La probabilité de perdre un message

Une autre contrainte intéressante concerne la probabilité de perdre un message dans le réseau. La pertinence de ce type de contrainte dépend cependant du type de protocole utilisé. Par exemple le protocole TCP propose des mécanismes d'acquittement qui permettent de détecter si un message n'a pas été reçu par le destinataire et de réitérer l'envoi. Ce type de protocole rend quasiment nulle la probabilité de perdre un message (en tout cas nous n'avons pas réussi à observer de telles pertes sur un réseau LAN, même après des dizaines de milliers de mesures). En revanche le protocole UDP, pour des raisons de rapidité, ne possède pas de mécanismes d'acquittement et il est probable que la proportion de messages perdus ne soit plus si négligeable. Pour des raisons techniques nous n'avons pu expérimenter qu'avec le protocole TCP, le protocole UDP n'étant pas pour le moment implanté dans le RTI que nous avons utilisé : CERTI [13]. On présente donc dans cette

section seulement la formalisation d'une contrainte de performances associée à la perte de messages mais sans expérimentations.

On doit d'abord modifier le modèle de réseau pour lui permettre de représenter la perte de messages. On propose de modifier les termes commençant par *Erlang-k* en rajoutant à la fin un choix entre le passage du message en sortie du réseau ou sa perte, on donne en exemple le terme *Erlang-k_a* :

$$\textit{Erlang-k\_a} \stackrel{def}{=} \underbrace{(\tau, k\lambda_a) \ldots (\tau, k\lambda_a)}_{k-1}.((Sa2R, k\lambda_a).LANF2R + (pe, \lambda_{pe}).LANF2R)$$

où $\lambda_{pe}$ correspond au taux de perte du message et *pe* est un nom d'action symbolisant cette perte.

Afin de pouvoir détecter les messages perdus on définit un processus à deux états, l'un signifiant *pas de message perdu*, noté *PasPerdu*, et l'autre signifiant *au moins un message a été perdu*, noté *Perdu*.

$$\begin{aligned} PasPerdu &\stackrel{def}{=} (pe, \top).Perdu \\ Perdu &\stackrel{def}{=} (pe, \top).Perdu \end{aligned}$$

Lequel est synchronisé avec le réseau par l'action *pe* :

$$Reseau \underset{\{pe\}}{\bowtie} PasPerdu$$

La définition du modèle complet devient donc la suivante :

$$SimulDist \stackrel{def}{=} (FederesConnect \underset{\mathcal{I}}{\bowtie} \text{Reseau} \underset{\{pe\}}{\bowtie} PasPerdu \underset{\mathcal{J}}{\bowtie} RTIGConnect)$$

Il est bien sûr possible de compliquer le processus *PasPerdu* en lui permettant de compter les messages perdus jusqu'à une certaine valeur en suivant la même démarche.

Il est alors possible de formaliser facilement la contrainte de performance suivante : *la probabilité de perdre un message dans les 10 premières secondes de simulation est inférieur à 0.1* :

$$SimulDist \Rightarrow true\ \mathcal{U}^{<10}_{<0.1} Perdu$$

où *SimulDist* est vrai à l'état initial du processus.

## 5.3 Résultats Obtenus

L'intérêt de faire de l'évaluation de performance sur les simulations distribuées est d'aider le concepteur de simulation à trouver la configuration qui maximise ou simplement vérifie la ou les contraintes de performances exigées. Aussi il sera intéressant de pouvoir paramétrer le modèle PEPA de sorte qu'un certain choix de valeurs modélise une certaine configuration de la réalité et satisfasse ainsi les performances de la simulation dans une configuration choisie. Les paramètres du modèle sont :

- la taille des messages de mise à jour,
- la vitesse de la machine qui héberge le RTIG,
- le débit du réseau,
- les souscriptions des fédérés entre eux, permettant de savoir où le RTIG doit redistribuer les messages.

Le nombre de fédérés par machine est aussi un point très important mais il n'est pas pris en compte dans ce modèle. Le nombre total de fédérés a également un impact sur les performances mais dans ce modèle il reste fixé au nombre de trois.

La relation entre les paramètres du modèle et les valeurs du modèle n'est pas directe. En effet le modèle constitue une abstraction de la réalité et de nombreux détails n'ont pas été décrits, comme par exemple l'impact des différentes couches du protocole de communication, la carte réseau utilisée, la qualité des liaisons et leur type, fibre optique ou non, les détails concernant l'algorithmique de l'intergiciel, et beaucoup d'autres détails. Comme dans tout problème de modélisation un certain nombre de connaissances peuvent, ou doivent, se réduire à de simples constantes, et le seul moyen d'obtenir les valeurs de ces constantes est de confronter le modèle à la réalité et de choisir les valeurs qui conduisent le modèle à ressembler le plus à la réalité. Ce problème sera abordé en détail au *chapitre* 6.

Dans ce modèle, les constantes sont toutes les valeurs de taux du modèle PEPA c'est-à-dire :

- $\lambda_a$, $\lambda_b$, $\lambda_c$, les taux des Erlang-k distributions qui gouvernent les distributions probabilistes de sorties d'un lien half duplex du réseau pour les messages émit par $A$, $B$ et $C$.
- $\lambda_{RTIG}$, la vitesse avec laquelle le RTIG envoie le message de mise à jour sur le réseau.
- $\lambda_A$, $\lambda_B$, $\lambda_C$, les fréquences des cycles des fédérés $A$, $B$ et $C$.

On définit les paramètres suivants :

- $L_a$, $L_b$ et $L_c$ représentent respectivement les tailles des messages de mise à jour $a$, $b$ et $c$.
- $V_{RTIG}$ représente la vitesse de la machine qui héberge le RTIG.
- $D$ représente le débit du réseau.

Pour pouvoir mener une vérification des performances en fonction des différents paramètres il faut une fonction, on la note $f$, qui, pour chaque choix de valeur de paramètres, définisse les valeurs du modèle, c'est-à-dire :

$$(\lambda_a, \lambda_b, \lambda_c, \lambda_{RTIG}) = f(L_a, L_b, L_c, V_{RTIG}, D)$$

Les taux $\lambda_A$, $\lambda_B$ et $\lambda_C$ ne sont pas présents dans l'équation ci-dessus car on fait l'hypothèse qu'il ne dépendent pas de ces paramètres. La fonction choisie dans l'expérimentation est une simple fonction affine à plusieurs dimensions :

$$\begin{aligned} f(L_a, L_b, L_c, V_{RTIG}, D) &= (\alpha + \beta \times L_a + \gamma \times L_b + \delta \times L_c + \eta \times V_{RTIG} + \theta \times D, \\ &\quad \vdots \\ &\quad ) \end{aligned}$$

sous entendu :

$$\begin{aligned} \lambda_a &= \alpha + \beta \times L_a + \gamma \times L_b + \delta \times L_c + \eta \times V_{RTIG} + \theta \times D \\ \lambda_b &= \dots \\ &\vdots \end{aligned}$$

où $\alpha$, $\beta$, $\gamma$, ... sont les coefficients de $f$ à déterminer de façon à ce que la correspondance entre les paramètres $L_a$, $L_b$, $L_c$, ... et les taux $\lambda_a$, $\lambda_b$, $\lambda_c$, ... soit la plus réaliste possible. On peut réduire le nombre de coefficients de $f$ en faisant l'hypothèse vraisemblable que la vitesse d'un message pour sortir du réseau est indépendante de la vitesse de la machine qui héberge le RTIG, et en supposant que le comportement d'un message ne dépend que de sa taille :

$$\begin{aligned} f(L_a, L_b, L_c, V_{RTIG}) = (&\alpha + \beta \times L_a + \theta \times D, \\ &\alpha + \beta \times L_b + \theta \times D, \\ &\alpha + \beta \times L_c + \theta \times D, \\ &\gamma + \delta \times V_{RTIG}) \end{aligned}$$

Une fonction affine est intéressante par sa simplicité, même si cette simplification est opérée au détriment d'une part du réalisme du modèle. De plus seul le paramètre $L = L_a = L_b = L_c$ a été considéré pour comparer le modèle à la réalité. En revanche, suivant ce paramètre, le modèle a déjà pu donner de bons résultats (voir la *section* 5.3.1). Les résultats donnés par le modèle en fonction des autres paramètres sont donc seulement théoriques et ne coïncident probablement pas avec la réalité.

Les courbes présentées dans cette section montrent les probabilités de satisfaire la deadline en fonction des paramètres $L$, $V_RTIG$ et $D$. Ces résultats de probabilité ont été obtenus grâce à un opérateur additionnel de CSL fourni dans PRISM permettant de calculer la mesure de probabilité d'un ensemble d'exécutions vérifiant une certaine propriété (voir la *section* 4.1.6). Les *figures* 5.3 et 5.4 montrent la probabilité de respecter la deadline exprimée à la *section* 5.2.2 en fonction du débit du réseau et de la vitesse de la machine hébergeant le RTIG. La *figure* 5.5 montre la probabilité de vérifier la deadline en fonction de la date de la deadline, celle-ci varie de 0ms à 1ms pour trois différentes tailles de messages L=64, L=256 et L=512 octets.

### 5.3.1 Confrontation des résultats avec la réalité

Cette section contient les résultats d'une comparaison entre le modèle et une simulation distribuée réelle. La simulation a été distribuée sur deux machines connectées par un câble full duplex de 100Mbits/s. Comme dans le modèle, le système est composé de trois fédérés et d'un RTIG.

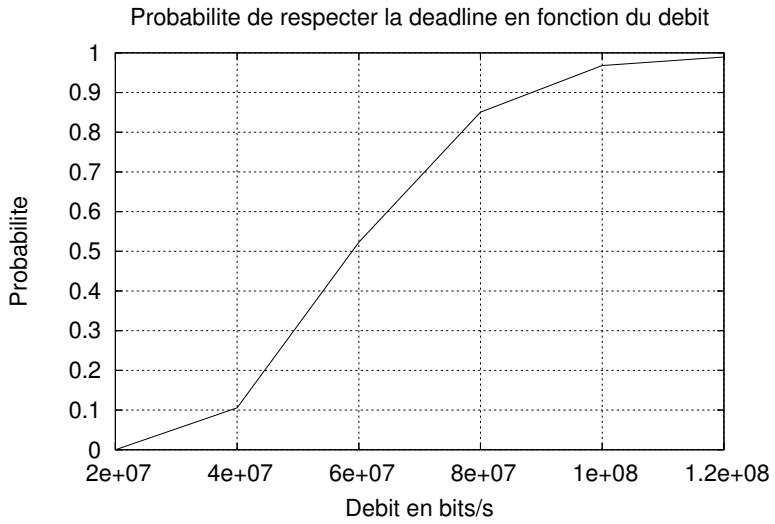



Figure 5.3: Résultats de PRISM en fonction du débit du réseau

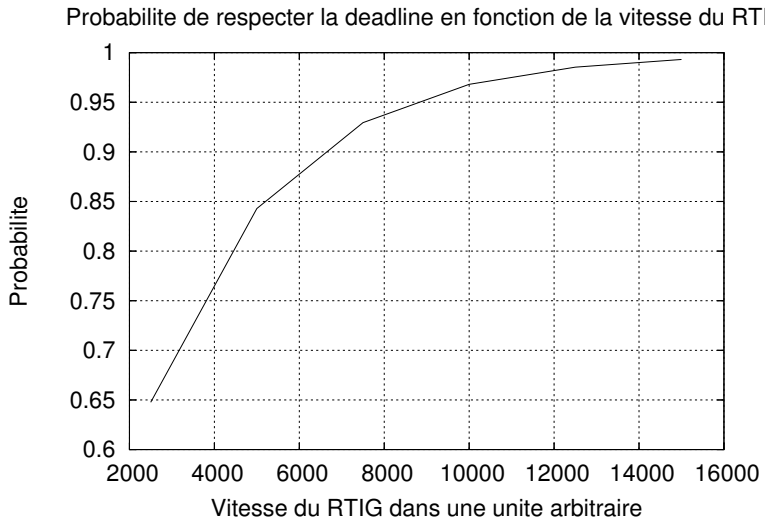



Figure 5.4: Résultats de PRISM en fonction de la vitesse de la machine hébergeant le RTIG

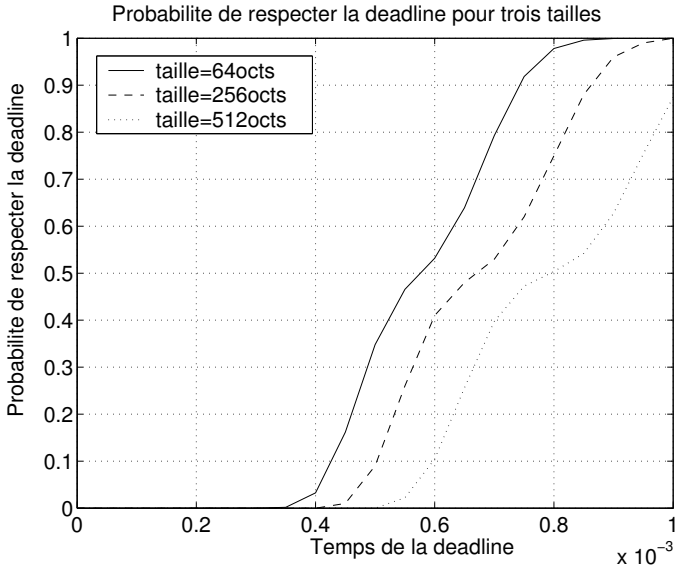



Figure 5.5: *Résultats, fournit par PRISM, de la probabilité de respecter la deadline en fonction de la date de la deadline pour différentes tailles de messages.*

Dans le but de mesurer le temps de transmission d'un message de mise à jour depuis un fédéré vers un autre fédéré avec la même horloge, les trois fédérés ont été hébergés sur la même machine. Le RTIG a été hébergé sur un Intel Pentium 4 cadencé à 1500MHz et les trois fédérés ont été hébergés sur un quadri-processeur[3] Intel cadencé à 800MHz. Dans cette expérience on a tout d'abord fourni des informations pour calibrer (à la main) les coefficients de la fonction $f$ (voir les *figures* 5.6 et 5.8). Puis pour montrer le pouvoir prédictif du modèle (voir *figures* 5.7). On peut noter les différences sur ces trois figures entre les distributions données par le modèle et les distributions réelles. La courbe donnée par le modèle subit une légère inflexion autour de la probabilité 0.5. Ceci s'explique par le modèle du RTIG qui répartit en moyenne une fois sur deux les messages provenant de $A$ vers $B$ et $C$. Aussi dans la réalité la courbe s'affaisse plus rapidement avec le temps tandis que la courbe du modèle reste assez raide jusqu'à la fin, avec pour conséquence que dans la *figure* 5.6, par exemple, après un délai de 0.8ms, le modèle et la réalité diffèrent davantage (d'une amplitude de presque 0.1). Ceci provient du fait que l'on a modélisé le réseau par une distribution de Erlang et que ce type de distribution représente mal les longues queues[4] qui existent sur les distributions associées à la transmission des messages sur un réseau réel.

[3]Nous avons choisi un quadri-processeur dans le but d'obtenir des résultats comparables à une situation où les trois fédérés se trouvaient sur trois machine différentes.

[4]Traduction du terme anglophone long-tail. Ce qui signifie informellement qu'il existe une proportion non négligeable d'individus même pour des valeurs (ici de temps) éloigné d'une certaine limite (ici la moyenne par exemple).

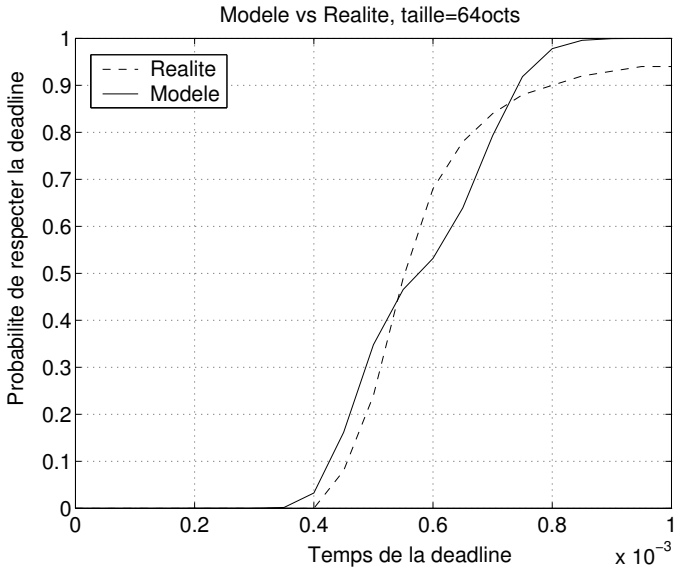



Figure 5.6: Modèle vs Réalité, L=64octs

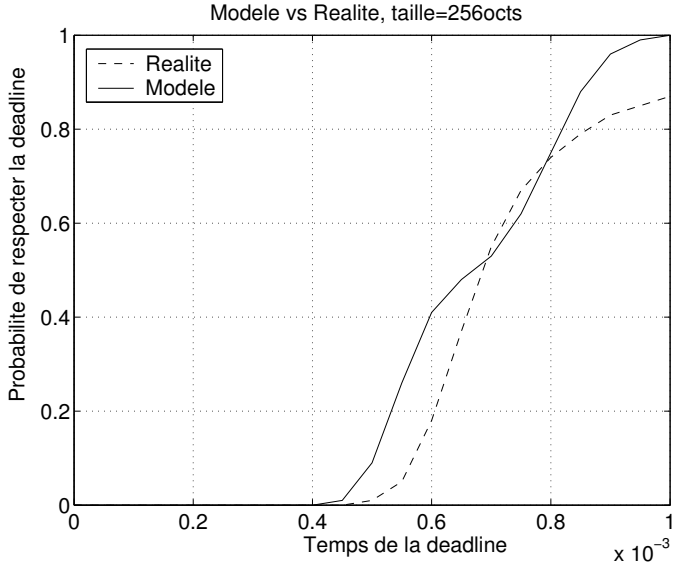



Figure 5.7: Modèle vs Réalité, L=256octs

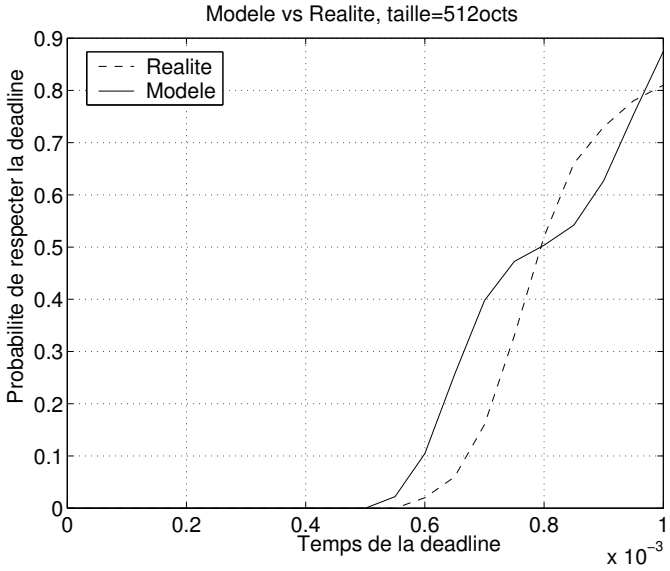



Figure 5.8: Modèle vs Réalité, L=512octs

### 5.3.2 Les difficultés de la méthode

Les difficultés associées à cette méthode sont de deux ordres :
- difficulté à fournir un modèle réaliste,
- difficulté à obtenir un modèle avec un espace d'états d'une taille appréhendable par le model checker.

Dans cette expérimentation, le modèle manque de réalisme, aussi bien dans son comportement en fonction du temps qu'en fonction des choix des paramètres. En effet, toutes les calibrations, c'est-à-dire les choix des coefficients $\alpha$, $\beta$, ... de $f$, ont été faites à la main. Le prochain chapitre montre comment résoudre ce problème de calibration à l'aide d'algorithmes d'approximations. Concernant l'explosion de l'espace d'états, PRISM est capable actuellement de supporter des modèles de plusieurs millions d'états (c'est le cas de ce modèle) mais pas encore plusieurs milliards (du moins n'a-t-on pas réussi à utiliser de tels modèles avec PRISM). Néanmoins il est souvent possible et parfois même facile d'abstraire le modèle par un modèle équivalent au regard de la formule à vérifier et ainsi de rendre le processus de vérification possible.

## 5.4 Conclusion de cette expérimentation

Cette expérimentation a permis de formuler des contraintes de performances et d'obtenir des résultats intéressants mais qui auraient pu l'être davantage si le modèle avait plus de réalisme. Le prochain chapitre montre comment obtenir des modèles plus réalistes, d'abord en utilisant des méthodes d'approximations existantes pour les CMTCs, ensuite en adaptant une telle méthode pour l'algèbre PEPA.

# Chapter 6

# Techniques de Modélisation pour le Model Checking Probabiliste

Le chapitre précédent a montré que l'utilisation du model checking probabiliste pour résoudre les problèmes d'évaluation de performances est pertinente à condition de travailler sur des modèles réalistes. Pour améliorer le réalisme de notre modèle nous allons dans un premier temps, en *section* 6.1, utiliser des algorithmes de fitting existants. Ces algorithmes permettront de générer une CMTC paramétrique (dont les taux dépendent d'une fonction d'un ensemble de paramètres) fidèle à la réalité mais malheureusement sans structure. En effet, le modèle système obtenu n'est pas modulaire. Il n'est pas possible de le transformer ou de l'adapter facilement à d'autres architectures de systèmes : en changeant le composant réseau par un autre, par exemple. La seconde partie de ce chapitre, *section* 6.2, propose d'adapter une technique de fitting existante permettant d'approcher les distributions de type phase de telle sorte qu'elle puisse s'appliquer directement sur un modèle PEPA plutôt que sur une CMTC. De cette manière il devient possible d'améliorer le réalisme du modèle tout en gardant un langage de modélisation modulaire et de haut niveau.

La *section* 6.3 propose une généralisation de l'algorithme donné dans la *section* 6.2. Celle-ci permet de travailler avec des exécutions partiellement observables au lieu de temps d'absorption. Il devient alors possible d'utiliser cet algorithme pour améliorer le réalisme de modèles PEPA sans état absorbant.

La *section* 6.4 présente le logiciel EMPEPA qui implante ces deux algorithmes et quelques fonctions de simulation permettant de tester leur bon fonctionnement.

Enfin, la *section* 6.5 conclue et propose quelques améliorations possibles de ces approches.

## 6.1 Un modèle réaliste mais sans structure

On montre dans cette section comment il est possible de construire un modèle très réaliste d'une simulation distribuée au regard de la contrainte de performances de type

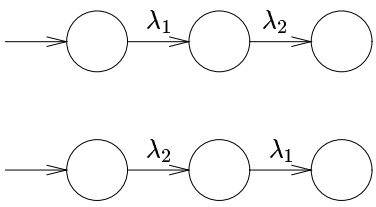



Figure 6.1: *Deux distributions de type phase équivalentes*

deadline probabiliste. Le but est de modéliser la loi probabiliste du temps de transmission d'un message entre deux fédérés pour différentes configurations, c'est-à-dire suivant les paramètres taille du message, nombre de fédérés et vitesse de la machine hébergeant le RTIG. Le paramètre type ou débit du réseau n'a pu être étudié pour des raisons techniques de sécurité (pare-feu n'acceptant pas les connexions entrantes de l'ONERA).

Le modèle que l'on cherche à obtenir est une CMTC, afin de pouvoir utiliser les techniques de model checking probabilistes. Deux méthodes d'approximations (ou dit de fitting) ont été successivement utilisées :

1. pour approcher la loi du temps de transmission d'un message pour une configuration donnée,
2. pour généraliser l'ensemble des lois dans toutes les configuration possible.

Les données nécessaires ont été collectées en mesurant le temps de transmission entre deux fédérés d'une simulation distribuée réelle dans plusieurs configurations (choix de la taille des messages, choix de la machine hébergeant le RTIG, ...). Pour chaque configuration les données ont été approchées par une distribution de type phase en utilisant le logiciel EMpht (voir la *section* 6.1.1). Toutes ces distributions ont été généralisées dans une distribution de type phase paramétrique en utilisant la régression polynomiale de Matlab.

### 6.1.1 Approcher des distributions avec le logiciel EMpht

EMpht est un logiciel écrit par Marita Olsson [52]. Son but est de trouver les taux du générateur infinitésimal d'une CMTC définissant une distribution de type phase de sorte que celle-ci maximise (éventuellement localement) la vraisemblance d'un ensemble de données. L'utilisateur doit décider du nombre d'états de la CMTC et éventuellement peut choisir un masque permettant de choisir parmi des distributions de Cox, de Erlang ou d'un type qu'il peut définir lui-même. L'algorithme maximise localement la vraisemblance. Cela signifie qu'il peut converger vers un maximum local. Il est important de remarquer qu'il peut existe plusieurs CMTCs équivalentes définissant la même distribution de type phase. La *figure* 6.1 donne un exemple de deux distributions de type phase équivalentes. Ceci s'explique car la densité de leur loi de probabilité s'obtient par le produit de convolution des lois exponentielles de taux $\lambda_1$ et $\lambda_2$. Or le produit de convolution est commutatif. Aussi, même si l'algorithme trouve une bonne approximation des données, il trouve seulement une des représentations possible de cette approximation.

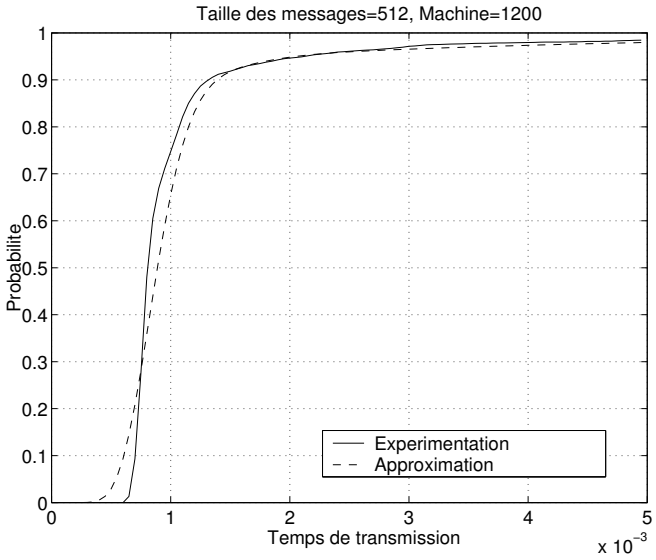



Figure 6.2: *Expérimentation vs approximation de distribution de type phase, l=512, h=1200*

### 6.1.2 Approximation d'une distribution pour chaque configuration

Les données ont été obtenues par des expérimentations paramétrées par deux dimensions :

1. la taille des messages, notée $l$, en octets,
2. la vitesse de la machine hébergeant le RTIG, notée h, en Hertz[1].

Les données des distributions ont été collectées pour différentes configurations. Chaque couple de valeur correspond à un choix de $(l, h)$ :

$$\{ \quad (32, 440); (64, 440); (128, 440); (256, 440); (512, 440); \\ (32, 1200); (64, 1200); (128, 1200); (256, 1200); (512, 1200) \quad \}$$

Pour chaque expérimentation, une CMTC de 16 états a été générée par EMpht. Les figures 6.2 et 6.3 présentent deux résultats d'approximation. Le choix du nombre d'états à été fixé à 16 après un certain nombre d'essais, les approximation obtenues semblant être satisfaisantes pour cette valeur. Celle-ci, quoique petite, engendre déjà 256 taux de transitions, donc autant de fonctions à approcher pour la phase suivante.

[1]Un indice de benchmark pourrait être plus pertinent que la fréquence du microprocesseur mais l'idée reste la même.

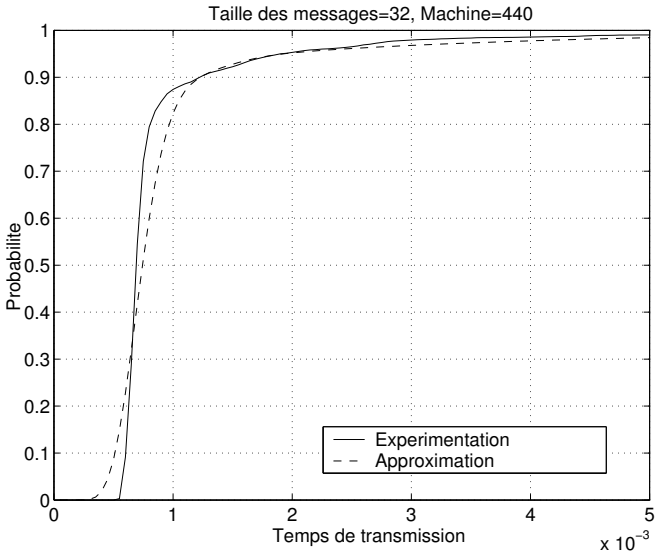



Figure 6.3: *Expérimentation vs approximation de distribution de type phase, l=32, h=440*

### 6.1.3 Généralisation des distributions de type phase en une seule paramétrique

Dans le but d'avoir un modèle prédictif pour toutes les valeurs de paramètres et pas seulement pour les valeurs expérimentées, on utilise l'approximation polynomiale pour induire une CMTC paramétrique généralisant les distributions de type phase obtenue auparavant. Une CMTC paramétrique est une fonction de l'espace des paramètres vers l'espace des CMTCs :

**Définition 6.1 (CMTC paramétrique)** *Soit $S$ un ensemble fini d'états, $\mathbb{P}$ un espace de paramètres, (en générale $\mathbb{R}^k$, $k$ est le nombre de paramètres), une matrice $\mathbf{Q}_{\mathbb{P}} : S \times S \mapsto \mathbb{R}^{\mathbb{P}}$ telle que :*

$$\forall i, j \in S,\ i \neq j \implies \mathbf{Q}_{\mathbb{P}}(i,j) \text{ est une fonction non négative}$$

$$\mathbf{Q}_{\mathbb{P}}(i,i) = -\sum_{k \neq i} \mathbf{Q}_{\mathbb{P}}(i,k)$$

*$\mathbf{Q}_{\mathbb{P}}$ est appelé le générateur infinitésimal paramétrique. $\pi_{\mathbb{P}} : S \mapsto [0,1]^{\mathbb{P}}$ est la distribution initiale paramétrique.*

La principale difficulté de cette phase de généralisation réside dans le choix des fonctions paramétriques $\mathbf{Q}_{\mathbb{P}}(i,j)$, noté $f_{i,j}$. Dans le cas présent, le choix s'est porté sur un polynôme à 2 variables $l$ et $h$. La *figure* 6.4 montre une fonction induite qui définit le taux en fonction des paramètres pour la transition allant de l'état $s_1$ à l'état $s_6$ de la distribution paramétrique de type phase. La *figure* 6.5 montre une distribution en fonction de la taille des messages sur une hypothétique machine tournant à 900MHz hébergeant le RTIG. Elle a été obtenue par la CMTC paramétrique avec $l \in [32, 512]$ et $h = 900$.

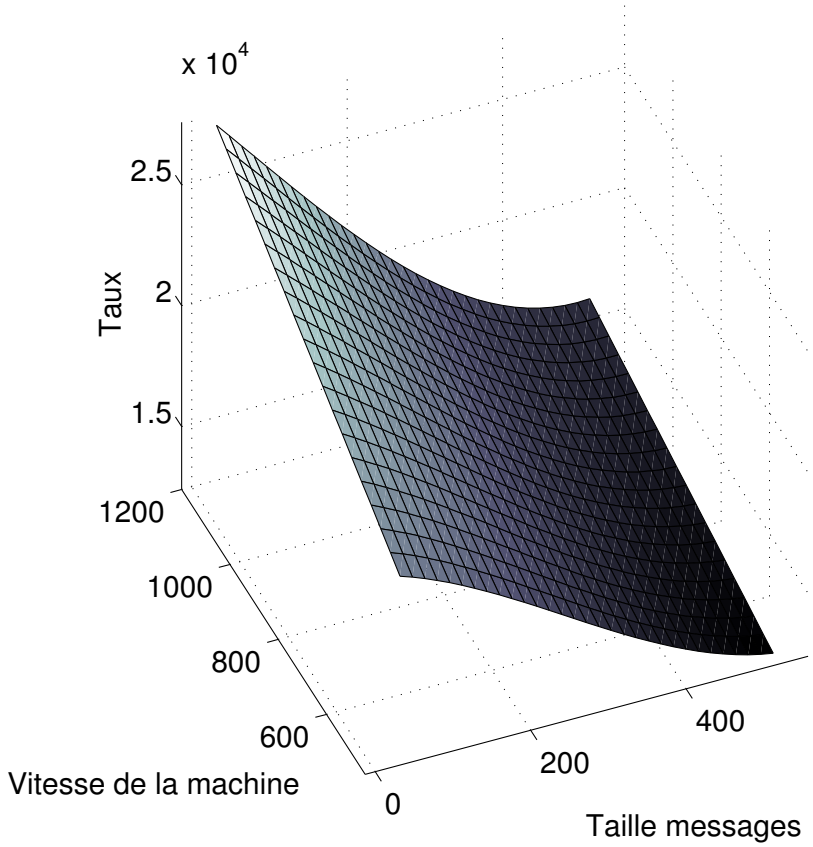



Figure 6.4: $f_{s_1,s_6}(l,h) = 16041 + 7.8 \times l - 0.07 \times l^2 + 10.1 \times l \times h$

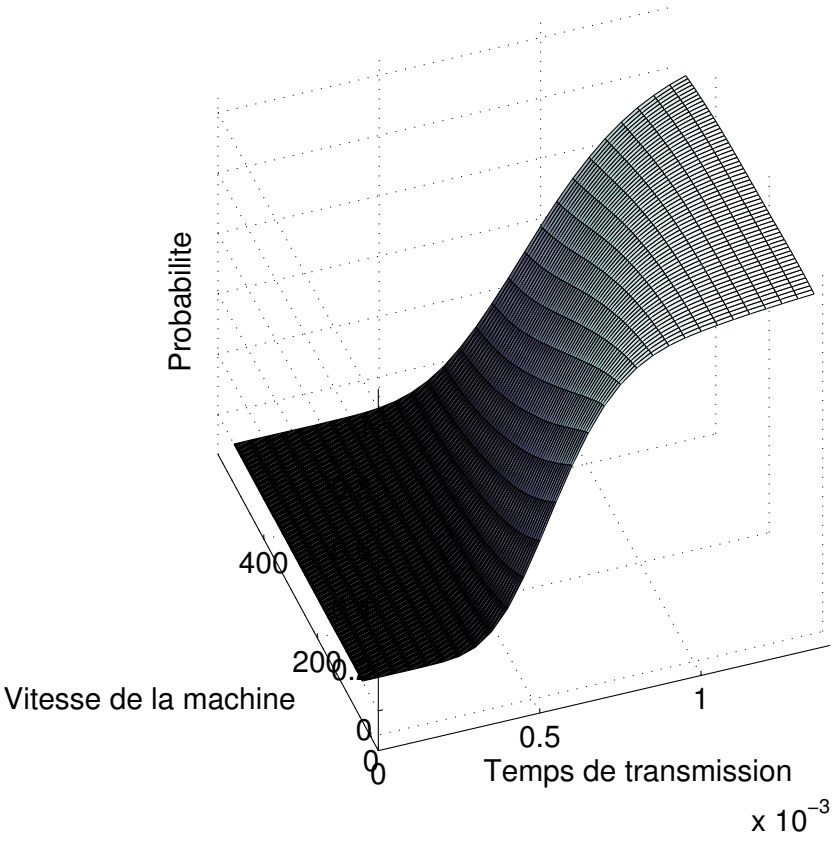



Figure 6.5: *Une distribution dépendant de la taille des messages d'une hypothétique machine à 900MHz hébergeant le RTIG.*

### 6.1.4 Conclusion de cette approche

Ces expérimentations montrent qu'il est possible d'obtenir un modèle assez réaliste avec peu de moyens : quelques distributions de type phase de moins d'une vingtaine d'états obtenues avec EMpht, le tout généralisé par des algorithmes d'approximation polynomiale.

Le problème de cette approche réside dans la représentation du modèle obtenu. Il donne de meilleurs résultats que le modèle PEPA de la *section* 5.1. Mais, en revanche, il est difficilement compréhensible et exploitable. Quels coefficients reflètent le comportement du réseau, des fédérés ou du RTIG ? Impossible de savoir, et donc impossible d'améliorer le modèle ou simplement de le réutiliser pour d'autres configurations d'architecture du système, par exemple en ajoutant de nouveaux composants, ou pour vérifier d'autres contraintes comme par exemple une contrainte de performances sur la perte des message.

## 6.2 Approximation de distributions de type phase dans un modèle PEPA

Afin de bénéficier du réalisme offert par ces méthodes d'approximation tout en permettant à l'utilisateur de modéliser son système avec un langage de haut niveau, nous avons étendu [24, 25, 26] sur l'algèbre PEPA, la méthode de fitting de Soren Asmussen, Olle Nerman et Marita Olsson pour les distributions de type phase (voir la section 4.3.4). Ceci va permettre au concepteur d'imposer une structure à son modèle exprimée en PEPA.

- L'algorithme prend en entrées :
  1. un ensemble de temps d'absorption obtenu à partir d'observations du système réel.
  2. un modèle PEPA contenant un certain nombre d'inconnues et décrivant une certaine distribution de type phase, que nous appellerons pour cette raison modèle PEPA *absorbant*.
- Il retourne en sortie : les valeurs des inconnues qui maximisent (éventuellement localement) la vraisemblance des données.

Afin de caractériser le fait que le modèle PEPA d'entrée décrit une distribution de type phase on ajoute le mot clef *nil* dans la syntaxe des termes PEPA. Ce mot clef définit le processus inactif. Voici la syntaxe des termes PEPA complétée avec le terme *nil* :

$$P,\ Q,\ \ldots \quad ::= \quad (\alpha, r).P \ \Big|\ P \underset{L}{\bowtie} Q \ \Big|\ P + Q \ \Big|\ P/L \ \Big|\ P[l] \ \Big|\ A \ \Big|\ nil$$

La sémantique opérationnelle reste la même puisque le processus *nil* ne produit aucune transition. Ce mot clef permet de définir facilement un état absorbant. Le terme *nil* n'est pas indispensable à la syntaxe PEPA pour exprimer l'inactivité. Un terme comme $(\alpha, 1).P < \alpha, \beta > (\beta, 1).P$ définit aussi un processus inactif. En effet, pour que le processus puisse évoluer les transitions $(\alpha, 1)$ et $(\beta, 1)$ ont besoin de se synchroniser ce qui est

impossible car leurs noms diffèrent. On pourra donc considérer *nil* comme le diminutif d'un tel terme dans la syntaxe PEPA traditionnelle.

On doit aussi définir un ensemble d'inconnues ou variables, noté $V$, et une fonction d'évaluation, notée $ev$, de l'ensemble $V$ vers $\mathbb{R}_+$. Chaque taux de transition du modèle PEPA pourra donc être :

- soit un réel positif,
- soit le taux de transition inconnu $\top$,
- soit une variable de $V$.

Une fois le modèle PEPA défini, l'algorithme a besoin de générer sa représentation markovienne, c'est-à-dire son graphe de dérivations et sa CMTC associée (voir la *section* 4.3.2). Pendant cette étape il est indispensable de s'assurer que le graphe de dérivation obtenu définit bien un processus stochastique, c'est-à-dire qu'il ne contient pas de transitions de taux inconnu (égal à $\top$). Dans le cas contraire, on aura à faire à un processus de décision (voir la *section* 4.2.1) et non à un processus stochastique. Le graphe de dérivations doit également contenir au moins un état absorbant : on parlera dans ce cas d'un graphe de dérivations *absorbant* et, par extension, d'un modèle PEPA *absorbant*.

On pourra supposer sans perte de généralité que le graphe de dérivations contient exactement un état absorbant dans la mesure où, s'il en contient plusieurs, un graphe de dérivations équivalent peut s'obtenir en fusionnant les états absorbants. Ceci revient à choisir un état absorbant, à éliminer les autres états absorbants et à rediriger leurs transitions entrantes vers l'état choisi. L'état absorbant pourra être le terme *nil* ou $nil <> nil$ ou $(a,1).P < a,b > (b,1).P$ ou tout autre terme équivalent à *nil*. Par souci de simplicité on le notera *nil*.

Cet algorithme présente aussi une restriction : les synchronisations du modèle PEPA ne peuvent avoir lieu qu'entre transitions actives (dont les taux de transition appartiennent à $\mathbb{R}_+$ ou $V$) et transitions passives (dont les taux sont $\top$). De plus, à un instant donnée et pour une action donnée, plusieurs transitions actives de peuvent se synchroniser qu'avec une seule transition passive. Dans ce cas la transition resultante d'une telle synchronisation prend comme taux celui de la transition active. Ceci simplifie grandement les types de taux que l'on peut trouver dans le graphe de dérivations associé puisqu'alors les seuls taux possibles seront soit des réels positifs, soit des variables de $V$ et non des combinaisons arithmétiques de réels et de variables ce qui compliquerait davantage la résolution mathématique du problème.

Un graphe de dérivations $G = (S, Act, s_\star, T)$ définit un processus stochastique, noté $\mathfrak{X} = (\mathfrak{X}_t)_{t\in\mathbb{R}_+}$, plus riche que le processus de sa CMTC associée, notée $X = (X_t)_{t\in\mathbb{R}_+}$. En effet les informations concernant les noms des actions sur les transitions ainsi que les transitions réflexives ne sont pas représentées par $X = (X_t)_{t\in\mathbb{R}_+}$. Le processus $\mathfrak{X}$ n'est que partiellement décrit par $X$. Pour compléter sa définition on définit $|T|$ processus aléatoires décrivant pour chaque transition $\zeta \in T$ le nombre de fois que $\zeta$ a été prise au cours du

temps :

$$\text{Pour tout } \zeta \in T, \quad C_t^\zeta = \text{nombre de fois que } \zeta \text{ à été prise entre 0 et } t$$

Par extension $C_t^a$ avec $a \in Act$ désignera le nombre total de fois que les transitions de nom d'action $a$ ont été prises entre 0 et $t$ :

$$\text{Pour tout } a \in Act \cup \{\tau\}, \quad C_t^a = \sum_{\substack{\zeta \in T, \\ a = act(\zeta)}} C_t^\zeta$$

De même $C_t^v$ avec $v \in V$ désignera le nombre total de fois que les transitions de taux $v$ ont été prises entre 0 et $t$ :

$$\text{Pour tout } v \in V, \quad C_t^v = \sum_{\substack{\zeta \in T, \\ v = rate(\zeta)}} C_t^\zeta$$

### Observations complètes, partielles et fonctions de vraisemblance

Une observation complète de $\mathfrak{X}$, notée $\mathfrak{x}$, pourra se définir de la façon suivante :

$$\mathfrak{x} = s_1 \xrightarrow{(t_1,\zeta_1)} \dots s_i \xrightarrow{(t_i,\zeta_i)} \dots s_m$$

où $m$ est la taille de $\mathfrak{x}$, et $t_i$ le temps passé dans l'état $s_i$ avant de prendre la transition $\zeta_i \in T$. L'état $s_1$ correspond à l'état initial, c'est-à-dire $s_\star$, et $s_m$ l'état absorbant, c'est-à-dire $nil$ (ou un état équivalent).

Soit $Y$ la variable aléatoire décrivant le temps d'absorption de $\mathfrak{X}$, c'est-à-dire le temps pour le processus $\mathfrak{X}$ d'atteindre $nil$. Une observation partielle de $\mathfrak{X}$, notée $y$, correspond à ce temps d'absorption. Comme dans la *section* 4.3.4, la vraisemblance de $y$, notée $L(y)$, est à la densité de probabilité d'avoir $Y = y$ :

$$L(y) = \pi_{s_\star} e^{y\mathbf{T}} \mathbf{t}$$

où $\mathbf{T}$ et $\mathbf{t}$ sont respectivement le générateur infinitésimal de phases et le vecteur de sortie de $X$. La vraisemblance de $\mathfrak{x}$, notée $L(\mathfrak{x})$, est la densité de probabilité d'avoir $\mathfrak{X} = \mathfrak{x}$, notée $\mathbb{P}(\mathfrak{X} = \mathfrak{x})$ (voir l'*annexe* C.1.3) :

$$\begin{aligned} \mathbb{P}(\mathfrak{X} = \mathfrak{x}) \quad = \quad & ev(\zeta_1) \times e^{-\mathbf{E}(s_1)t_1} \\ & \times ev(\zeta_2) \times e^{-\mathbf{E}(s_2)t_2} \\ & \vdots \\ & \times ev(\zeta_m) \times e^{-\mathbf{E}(s_m)t_m} \end{aligned}$$

où $\mathbf{E} : S \mapsto \mathbb{R}_+$ est un vecteur contenant, pour chaque $s \in S_{ph}$, le taux de probabilité de prendre une transition (y compris une transition réflexive) :

$$\text{Pour } s \in S_{ph}, \quad \mathbf{E}(s) = \sum_{\substack{\zeta \in T, \\ \bullet\zeta = s}} ev(\zeta)$$

Soit $n$ le nombre d'observations. On considère le vecteur de processus stochastiques $\chi$ comme le produit de $n$ processus $\mathfrak{X}$ identiques et indépendants. $\chi$ désigne une valeur de $\chi$, c'est-à-dire la donnée de $n$ exécutions complètes de $\mathfrak{X}$. Soit $\mathbf{Y}$ le vecteur aléatoire représentant les $n$ temps d'absorption des $n$ processus de $\chi$. $\mathbf{y} = (y_1, \ldots, y_n)$ dénote l'ensemble des temps d'absorption observés. La vraisemblances de $\mathbf{y}$, notée $L(\mathbf{y})$ est mesurée par la densité de probabilité d'avoir $\mathbf{Y} = \mathbf{y}$, c'est-à-dire le produit des densités de probabilités d'avoir chaque observation :

$$L(\mathbf{y}) = \prod_{i=1}^{n} \mathbb{P}(Y = y_i) = \prod_{i=1}^{n} L(y_i)$$

De même la vraisemblance de $\chi$, notée $L(\chi)$, est mesurée par la densité de probabilité de $\chi = \chi$ :

$$L(\chi) = \prod_{i=1}^{n} \mathbb{P}(\mathfrak{X} = \mathfrak{x}_i) = \prod_{i=1}^{n} L(\mathfrak{x}_i)$$

L'algorithme a pour but de trouver la fonction d'affectation $ev : V \mapsto \mathbb{R}_+$ qui maximise la vraisemblance de $\mathbf{y}$. Il est important de noter que l'algorithme ne devine pas le modèle PEPA, c'est à l'utilisateur de choisir ce modèle et de choisir où disposer dans ce modèle les variables du problème. Comme dans la *section* 4.3.4 l'algorithme va consister à alterner successivement les étapes $E$ et $M$.

### Étape $E$ : calcul des espérances

Avant de passer à la description détaillée du calcul des espérances de l'algorithme on introduit quelques notations.

**Définition 6.2** *Soit $G = (S, Act, s_\star, T)$ un graphe de dérivations avec exactement un état absorbant. Soit $V$ l'ensemble des variables du problème d'approximation et $ev : V \mapsto \mathbb{R}_+$ une fonction d'évaluation sur $V$. Chaque transition de $T$ a un taux appartenant à $\mathbb{R}_+$ ou bien à $V$. On définit les notations suivantes :*

- *$\mathbb{T} : V \mapsto \mathcal{P}(T)$ désigne pour chaque variable $v$ de $V$ le multi-ensemble[2] $\mathbb{T}(v)$ contenant toutes les transitions ayant pour taux $v$.*

$$\forall v \in V \quad \mathbb{T}(v) = \{\zeta \in T \mid rate(\zeta) = v\}$$

- *On étend $ev$ à l'ensemble des transitions $\zeta \in T$ tel que :*

$$ev(\zeta) = \begin{cases} rate(\zeta) & si\ rate(\zeta) \in \mathbb{R}_+ \\ ev(rate(\zeta)) & si\ rate(\zeta) \in V \end{cases}$$

Pour obtenir la CMTC associée à $G$ il faudra bien sûr considérer l'évaluation des taux de transitions avant de les additionner (voir la section 4.3.2 pour savoir comment obtenir la CMTC associée à $G$). On considère les variables aléatoires suivantes :

[2]Un multi-ensemble est un ensemble pouvant contenir plusieurs occurrences du même objet.

– Soient :

$$Z_v(\omega) = \sum_{\zeta \in \mathbb{T}(v)} Z_{\bullet\zeta}(\omega)$$

$|V|$ variables aléatoires. Les variables $(Z_s)_{s \in S_{ph}}$ sont définies à la *section* 4.3.4 et définissent le temps total de séjour d'un processus $X$ dans chaque état de $S_{ph}$. Puisque $X$ admet un état absorbant $Z_v$ converge presque sûrement.

– Soient :

$$N_v(\omega) = \lim_{\epsilon \to 0} \sum_{i=0}^{\infty} \mathbf{1}_{\{C^v_{i\epsilon}(\omega) \neq C^v_{(i+1)\epsilon}(\omega)\}}$$

$|V|$ variables aléatoires désignant pour chaque variable $v \in V$ le nombre de fois où le processus $\mathfrak{X}$ emprunte une transition $\zeta$ de taux $v$. Puisque que $\mathfrak{X}$ possède un état absorbant $N_v$ converge presque sûrement.

– Soient :

$$\mathbf{Z}_v((\omega_1, \ldots, \omega_n)) = \sum_{i=1}^{n} Z_v(\omega_1)$$

$|V|$ variables aléatoires où $\omega_i$ est un événement élémentaire de l'univers du processus $\mathfrak{X}$.

– Soient :

$$\mathbf{N}_v((\omega_1, \ldots, \omega_n)) = \sum_{i=1}^{n} N_v(\omega_i)$$

$|V|$ variables aléatoires désignant pour chaque variable $v \in V$ le nombre total de fois où les processus de $\mathbf{X}$ empruntent une transition $\zeta$ de taux $v$.

L'étape $E$ consiste à calculer les espérances conditionnelles $\mathbb{E}[\mathbf{Z}_v | \mathbf{Y} = \mathbf{y}]$ et $\mathbb{E}[\mathbf{N}_v | \mathbf{Y} = \mathbf{y}]$. D'après l'*annexe* C.3.1 et la définition de $Z_v$, l'expression de l'espérance conditionnelle $\mathbb{E}[\mathbf{Z}_v | \mathbf{Y} = \mathbf{y}]$ est immédiate :

$$\mathbb{E}[\mathbf{Z}_v | \mathbf{Y} = \mathbf{y}] = \sum_{i=1}^{n} \frac{\sum_{\zeta \in \mathbb{T}(v)} c^{\bullet\zeta}_{\bullet\zeta}(y_i)}{\pi_{s_\star} \mathbf{b}(y_i)}$$

Aussi, on montre dans l'*annexe* C.3.4 que :

$$\mathbb{E}[\mathbf{N}_v | \mathbf{Y} = \mathbf{y}] = \sum_{i=1}^{n} \frac{1}{\pi_{s_\star} \mathbf{b}(y_i)} \sum_{\zeta \in \mathbb{T}(v)} ev(v) \times \begin{cases} c^{\bullet\zeta}_{\zeta\bullet}(y_i) & \text{si } \zeta\bullet \neq nil \\ a_{\zeta\bullet}(y_i) & \text{si } \zeta\bullet = nil \end{cases}$$

Les fonctions $\mathbf{a}$, $\mathbf{b}$ et $\mathbf{c}$ correspondent aux systèmes d'équations différentielles ordinaires définis dans la *section* 4.3.4. Les méthodes de résolution employées sont les mêmes que celles décrites dans la *section* 4.3.4. On propose en plus une autre méthode pour calculer $\mathbf{Z}_s$ et $\mathbf{N}_v$ par simulation discutée à la *section* 6.4.1.

### Étape $M$ : maximisation de la vraisemblance de $\chi$

Les espérances $\mathbb{E}[\mathbf{Z}_v|\mathbf{Y}=\mathbf{y}]$ et $\mathbb{E}[\mathbf{N}_v|\mathbf{Y}=\mathbf{y}]$ une fois calculées, il ne reste qu'à mettre les variables du modèle à jour de sorte que celui-ci maximise la vraisemblance de $\chi$. On montre dans l'*annexe* C.4.2 que la maximisation de $\chi$ s'obtient en affectant les variables de $V$ avec les valeurs suivantes :

$$\forall v \in V, \quad ev(v) = \frac{\mathbb{E}[\mathbf{N}_v|\mathbf{Y}=\mathbf{y}]}{\mathbb{E}[\mathbf{Z}_v|\mathbf{Y}=\mathbf{y}]}$$

Pour comprendre cette formule plus intuitivement le lecteur pourra imaginer que pour chaque $v \in V$ toutes les transitions de $\mathbb{T}(v)$ ne forment qu'une seule macro-transition et que tous les états dont sortent ces transitions (ou cette macro-transition) ne forment qu'un seul macro-état. Ensuite pour obtenir le taux de cette macro-transition on divise le nombre de fois total où elle a été prise par le temps de séjour total dans le macro-état.

### Schémas de l'algorithme

On résume l'algorithme dans un langage pseudo-formel ci-dessous :

`Entrées :`

1. `un graphe de dérivations absorbant` $G = (S, Act, s_\star, T)$`,`
2. `un ensemble de variables` $V$`,`
3. `un ensemble d'observations` $\mathbf{y}$`.`

`Sortie : une fonction d'évaluation` $ev$

`Algorithme :`

1. `Choisir une initialisation de` $ev$.
2. `Jusqu'à ce qu'un critère d'arrêt soit vérifié, itérer :`
   - `Étape` $E$ : `pour tout` $v \in V$ `calculer :`

$$\mathbf{z}_v := \mathbb{E}[\mathbf{Z}_v|\mathbf{Y}=\mathbf{y}], \quad \mathbf{n}_v := \mathbb{E}[\mathbf{N}_v|\mathbf{Y}=\mathbf{y}]$$

   - `Étape` $M$ : `pour tout` $v \in V$ `affecter :`

$$ev(v) := \frac{\mathbf{n}_v}{\mathbf{z}_v}$$

De la même manière que pour l'algorithme de la *section* 4.3.4 l'initialisation des variables peut jouer un rôle important dans l'issue du résultat car l'algorithme ne converge pas forcément vers le maximum global. Il est donc conseillé :

- soit de faire une bonne initialisation manuelle pour aider l'algorithme à converger vers un bon maximum local ou vers le maximum global,
- soit de faire plusieurs tentatives avec différentes initialisations aléatoires ou manuelles afin d'augmenter les chances d'obtenir un bon résultat.

Le critère d'arrêt de l'algorithme peut par exemple correspondre au moment où le logarithme de la vraisemblance de $\mathbf{y}$ n'augmente plus de manière significative. Comme remarqué dans la *section* 4.3.4 cette vraisemblance se calcule facilement et n'ajoute pas de complexité supplémentaire à l'algorithme.

Enfin la complexité de l'algorithme et équivalente à celui de la *section* 4.3.4 car la majorité des calculs consiste à résoudre les systèmes différentiels ordinaires $\mathbf{a}$, $\mathbf{b}$ et $(\mathbf{c}^s)_{s\in S_{ph}}$ (se reporter à la *section* 4.3.4 pour davantage de détail).

### 6.2.1 Expérimentations

On présente maintenant quelques expérimentations faites avec cet algorithme : deux premiers exemples simples et un troisième exemple plus compliqué car il modélise le temps de transmission des messages d'une simulation distribuée HLA. Les résultats ont été obtenus à l'aide du logiciel *EMPEPA* développé dans le cadre de cette thèse pour implanter cet algorithme. On donne une présentation détaillée de *EMPEPA* à la *section* 6.4.

#### Exemple 1 : retrouver les valeurs initiales d'un modèle

Soit $V = \{\lambda_1, \lambda_2\}$ l'ensemble des variables du problème. Le modèle PEPA ne comporte qu'un seul terme qui est le terme racine suivant :

$$(\tau, \lambda_1).(\tau, \lambda_1).(\tau, \lambda_1).(\tau, \lambda_1).(\tau, \lambda_1).(\tau, \lambda_1).nil$$
$$+$$
$$(\tau, \lambda_2).(\tau, \lambda_2).(\tau, \lambda_2).(\tau, \lambda_2).(\tau, \lambda_2).(\tau, \lambda_2).nil$$

On a initialisé $\lambda_1 = 1$ et $\lambda_2 = 2$ et on a obtenu par simulation du modèle 10000 temps d'absorption décrivant la distribution de type phase décrite par la *figure* 6.6. Ces 10000 temps d'absorption ont été utilisés en entrée de l'algorithme dans le but de retrouver les valeurs de $\lambda_1$ et $\lambda_2$. Les valeurs $\lambda_1$ et $\lambda_2$ ont été initialisées aléatoirement. Après 10 itérations (quelques secondes de calcul) l'algorithme a retrouvé les valeurs $\lambda_1 = 1.01$ $\lambda_2 = 1.98$. L'algorithme a donc été en mesure de retrouver quasiment les valeurs initiales du modèle.

#### Exemple 2 : trouver les valeurs d'une distribution équivalente

Soit $V = \{v, w\}$ l'ensemble des variables du problème. L'ensemble des déclarations des constantes PEPA sont :

$$\begin{aligned} Begin &\stackrel{def}{=} (\tau, 2).\big((\alpha, 4).End + (\beta, v).End + (\gamma, w).End\big) + (\alpha, 1).End \\ Loop &\stackrel{def}{=} (\alpha, \top).Loop \\ End &\stackrel{def}{=} nil \end{aligned}$$

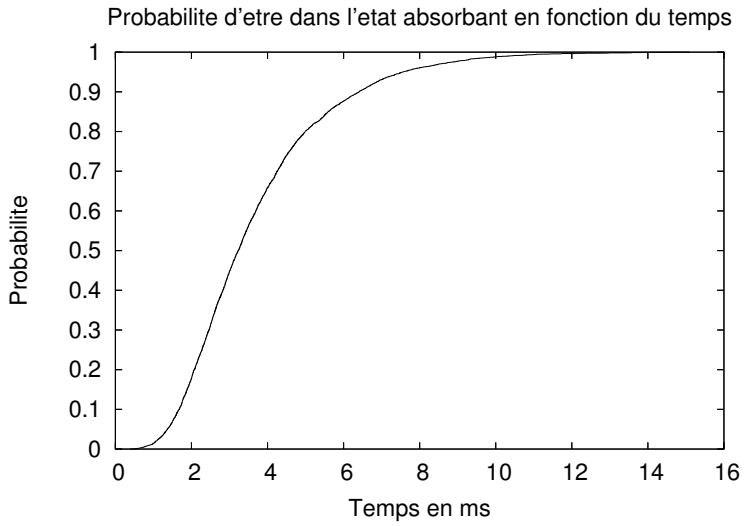



Figure 6.6: *La probabilité d'être dans l'état absorbant en fonction du temps.*

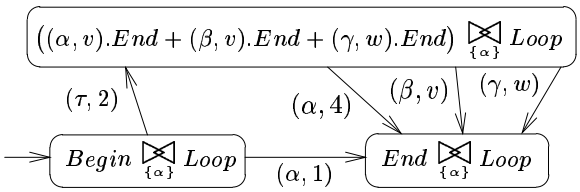



Figure 6.7: *Graphe de dérivations de l'exemple 2*

et le terme racine est : $Begin \underset{\{\alpha\}}{\bowtie} Loop$

Comme dans l'exemple 1 on a initialisé $v = 1$ et $w = 2$ et obtenu par simulation du modèle 10000 temps d'absorption décrivant la distribution de type phase du modèle. Ensuite les valeurs $v$ et $w$ ont été réinitialisées aléatoirement. Après une centaine d'itérations (quelques minutes de calcul) l'algorithme a convergé les valeurs $v = 3.094302$ et $w = 0.006188605$. Bien que ces valeurs ne correspondent pas aux valeurs initiales elles donnent bien une distribution de type phase équivalente. En effet les deux transitions du modèle ayant pour taux $v$ et $w$ partent du même état et vont dans le même état (voir la figure 6.7), or $v + w \approx 3$ dans les deux cas, donc le résultat donné par l'algorithme donne bien une distribution équivalente.

### Exemple 3 : modélisation du temps de traversée d'un message entre deux fédérés d'une simulation HLA

Dans cet exemple on décrit un modèle simplifié mais néanmoins structuré du temps de transmission d'un message de mise à jour entre deux fédérés d'une simulation distribuée

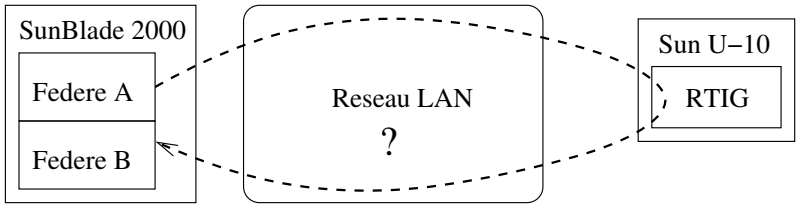



Figure 6.8: *Schémas de l'exemple 3*

HLA. Les données sont, cette fois, des observations obtenues à partir d'une simulation distribuée HLA réelle.

### Description du problème

Soit un système composé de trois entités physiques, deux ordinateurs et un réseau LAN. Le premier ordinateur est un SunBlade 2000 bi-processeur hébergeant deux fédérés $A$ et $B$. Le second ordinateur est un sun U-10 hébergeant un intergiciel, le RTIG de CERTI [13]. 20000 temps d'absorption ont été obtenus en envoyant 20000 messages de 256 octets depuis le fédéré $A$ jusqu'au fédéré $B$ Chaque message a parcouru l'itinéraire suivant (voir la figure 6.8) :

1. traverser le réseau du fédéré $A$ jusqu'au RTIG,
2. être traité par RTIG,
3. traverser le réseau du RTIG jusqu'au fédéré $B$.

L'objectif posé par cet exemple est d'obtenir une bonne approximation du temps de réponse du réseau, sachant que :

1. on possède déjà un modèle du temps de réponse du RTIG,
2. on suppose que traverser le réseau depuis le fédéré $A$ jusqu'au RTIG ou depuis le RTIG jusqu'au fédéré $B$ donne un comportement équivalent.

### Description du modèle

On donne d'abord une description simplifiée du modèle PEPA modélisant le système (c'est à dire le comportement d'un message dans l'ensemble des entités) puis on présente les résultats obtenus avec un modèle PEPA plus complexe et réaliste. Le terme racine du modèle est :

$$AbsSimulDist$$

Soit $V$ l'ensemble des variables du problème d'approximation :

$$V = \{v_1, \ldots, v_5\}$$

Le terme *AbsSimulDist* est égal au produit synchronisé des termes :

$$ReseauVersRTIG,\ RTIG \text{ et } RTIGVersReseau$$

Les deux termes *ReseauVersRTIG* et *RTIGVersReseau* dénotent la même distribution de type phase, une distribution de Cox (de trois états de phase dans l'exemple réduit). Les termes :

$$CoxAller1,\ CoxAller3,\ CoxAller3$$

et

$$CoxRetour1,\ CoxRetour3,\ CoxRetour3$$

sont utilisés pour modéliser chaque état de phase de cette distribution.

Les mêmes variables $v_1$ à $v_5$ sont utilisées à la fois dans

$$ReseauVersRTIG \text{ et } RTIGVersReseau$$

dans le but de représenter l'identité de comportement du réseau à l'aller et au retour.

Le terme *RTIG* est une distribution de Erlang-4 de taux $4\lambda$ où $\lambda$ une valeur connue.

Il y a deux noms d'actions :

$$e \text{ et } s$$

définissant respectivement les interactions mises en jeu lorsqu'un message sort du réseau pour aller sur le RTIG et sort du RTIG pour aller sur le réseau.

Quand le message est dans les câbles du réseau ou dans le silicium de l'ordinateur hébergeant l'intergiciel l'action invisible $\tau$ est utilisée pour modéliser les changement internes (encore d'une manière assez grossière mais réaliste d'un point de vu macroscopique). L'ensemble des déclarations des termes du modèle est donné ci-dessous :

$$\begin{array}{lcl}
AbsSimulDist & \stackrel{def}{=} & ReseauVersRTIG \underset{\{e\}}{\bowtie} RTIG \underset{\{s\}}{\bowtie} RTIGVersReseau \\
\\
ReseauVersRTIG & \stackrel{def}{=} & CoxAller1 \\
CoxAller1 & \stackrel{def}{=} & (\tau, v_1).CoxAller2 + (e, v_2).nil \\
CoxAller2 & \stackrel{def}{=} & (\tau, v_3).CoxAller3 + (e, v_4).nil \\
CoxAller3 & \stackrel{def}{=} & (e, v_5).nil \\
\\
RTIG & \stackrel{def}{=} & (e, \top).(\tau, \lambda).(\tau, \lambda).(\tau, \lambda).(s, \lambda).nil \\
\\
RTIGVersReseau & \stackrel{def}{=} & (s, \top).CoxRetour1 \\
CoxRetour1 & \stackrel{def}{=} & (\tau, v_1).CoxRetour2 + (\tau, v_2).nil \\
CoxRetour2 & \stackrel{def}{=} & (\tau, v_3).CoxRetour3 + (\tau, v_4).nil \\
CoxRetour3 & \stackrel{def}{=} & (\tau, v_5).nil
\end{array}$$

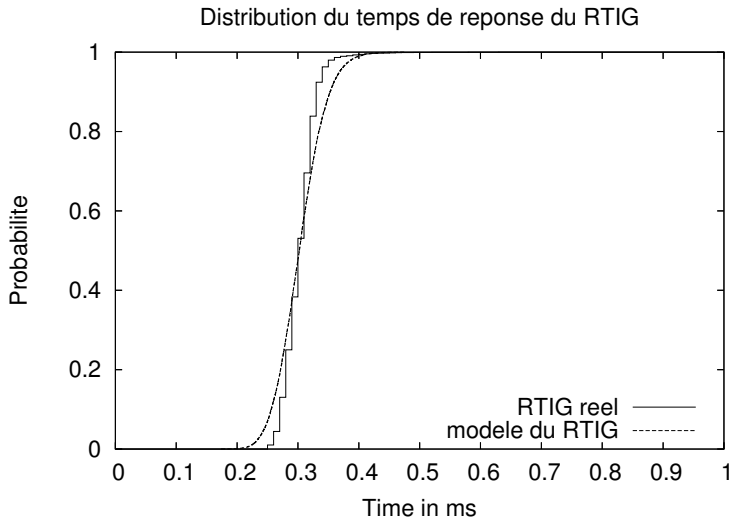



Figure 6.9: *Distribution cumulative du temps de réponse du RTIG*

### Résultats

Le même modèle, mais avec 32 états pour le réseau et 64 états pour le RTIG, a été utilisé comme modèle d'entrée du logiciel EMPEPA. L'ensemble des données consiste en 20000 temps d'absorption obtenus en mesurant le temps de transmission entre deux fédérés pendant l'exécution de la simulation distribuée.

### Obtenir un modèle du RTIG

L'algorithme a d'abord été utilisé pour obtenir une approximation du temps de réponse du RTIG, une distribution de Erlang-64. Après 4 itérations[3] (quelques heures) l'algorithme a convergé vers la bonne valeur. La *figure* 6.9 montre la comparaison entre la distribution du temps de réponse réel du RTIG et celle définie par le terme $RTIG$ après évaluation du taux $\lambda$ par EMPEPA.

### Obtenir un modèle du réseau

Ensuite le modèle complet du système à été utilisé en entrée de l'algorithme. L'ensemble des variables du problème correspondent aux 63 variables définissant les distribution de Cox de *ReseauVersRTIG* et de *RTIGVersReseau*. La *figure* 6.10 montre la distribution obtenue par le modèle après 500 itérations (environ une semaine de calcul). Le modèle du réseau a été extrait très facilement, en remplaçant les variables par les

[3]Le faible nombre d'itération est dû au fait que le RTIG est modélisé par une distribution de Erlang avec une unique variable, $\lambda$. En théorie une itération aurait suffi. Ceci vient probablement du fait que, étant donnée la précision limité des calculs, des erreurs se répercutent d'itération en itération de sorte que la convergence théorique diffère de la convergence pratique.

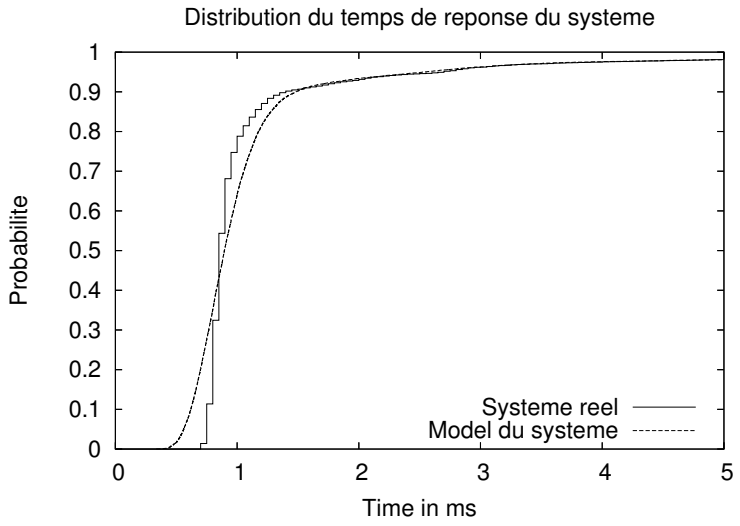



Figure 6.10: *Distribution cumulative du temps de réponse du système, c'est-à-dire du temps pour un message de traverser le réseau, d'être traité par le RTIG et de traverser le réseau dans l'autre sens*

valeurs retournées par l'algorithme dans le terme *RTIGVersReseau* ou (ce qui revient au même) *RTIGVersReseau*. La *figure* 6.11 montre la distribution du temps de réponse du réseau pour un aller. La *figure* 6.12 montre les différentes distributions obtenues.

## 6.3 Trouver les valeurs les plus vraisemblables d'un modèle PEPA à partir d'un ensemble d'exécutions partiellement observables

Dans cette section on présente une généralisation de l'algorithme donné dans la section précédente. L'objectif ici est de maximiser la vraisemblance d'un ensemble d'exécutions *partiellement observables* au lieu d'un ensemble de temps d'absorption. L'algorithme permet de travailler sur des modèles PEPA avec ou sans état absorbant. Par contre il reste toujours limité à des modèles dont les synchronisations ne se produisent qu'entres transitions passives et actives.

Soit $G = (S, Act, s_\star, T)$ un graphe de dérivations (pas forcément absorbant) dont les taux des transitions appartiennent ou bien à $\mathbb{R}_+$ ou bien à $V$. $\mathfrak{X} = (\mathfrak{X}_t)_{t\in\mathbb{R}_+}$ est le processus stochastique décrivant entièrement $G$ (comme défini à la *section* 6.2). Une observation partielle de $\mathfrak{X}$ et caractérisée par le vecteur aléatoire $\mathfrak{Y}$. Une expérience de $\mathfrak{Y}$, notée $\mathfrak{y}$, représente une exécution *partiellement observable* de $\mathfrak{X}$. Elle se note de la

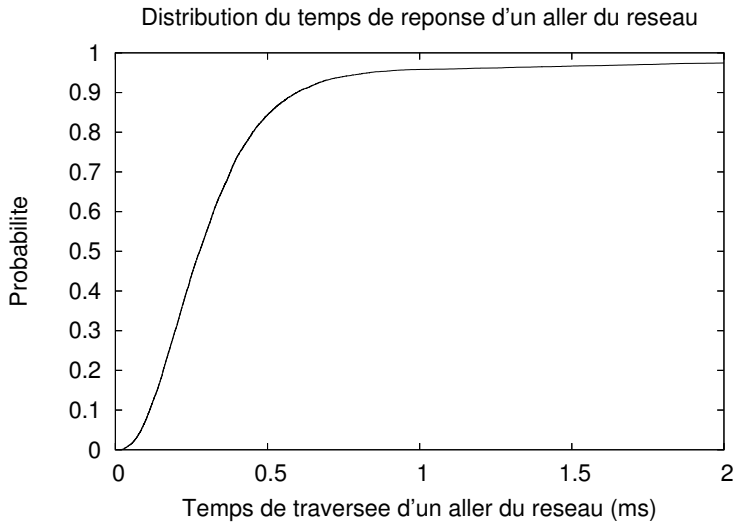



Figure 6.11: *Distribution cumulative du temps de réponse d'un traversant un aller du réseau.*

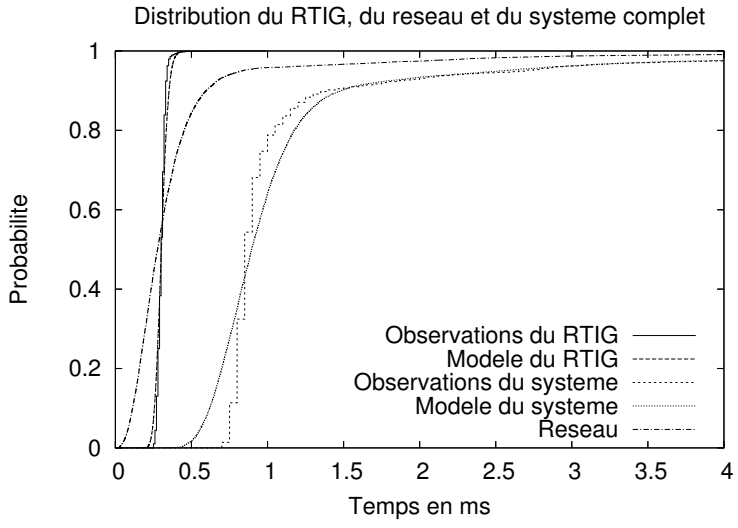



Figure 6.12: *Graphe récapitulatif*

manière suivante :

$$\mathfrak{y} = init \xrightarrow{t_1} a_1 \xrightarrow{t_2} a_2 \ldots a_{m-1} \xrightarrow{t_m} stop$$

avec $a_1, \ldots, a_{m-1} \in Act$ et $t_1, \ldots, t_m \in \mathbb{R}_+$. Toute exécution partiellement observable commence par l'état initial du graphe de dérivations : c'est ce que le mot clef *init* signifie. Le mot clef *stop* indique que l'observateur décide d'arrêter son observation. Chaque triplet $a \xrightarrow{t} b$ avec $a \in Act \cup \{init\}$ et $b \in Act \cup \{stop\}$ signifie qu'il s'est écoulé un temps $t$ entre l'observation $a$ et l'observation $b$. L'exécution est partielle au sens où seules les actions visibles[4], *init* et *stop* incluses, sont observables. En revanche, toute action visible est forcément observée, ce qui se traduit par l'hypothèse suivante : *Aucune transition avec une action observable ne peut être prise entre deux observations.*

L'utilisateur peut facilement choisir quelles sont les actions observables ou inobservables en utilisant l'opérateur d'invisibilité de l'algèbre PEPA (voir la *section* 4.3.2) qui permet de cacher un ensemble de noms d'actions. Ceci pourra se révéler pratique si le modèle contient dans sa description des actions visibles qui, pour des raisons techniques ou autres, ne sont pas observables sur le système réel.

L'étape de maximisation de l'algorithme reste rigoureusement la même que pour l'algorithme précédent. Seul le calcul des espérances change. Puisque le graphe de dérivations ne contient pas forcément d'état absorbant les espérances des variables aléatoires $Z_v$ et $N_v$ ne convergent pas forcément. Afin de remédier à ce problème on va considérer "virtuellement" que le graphe de dérivations possède toujours au moins un état absorbant appelé $s_{stop}$ et que chaque état du graphe de dérivations, à l'exception de $s_{stop}$, possède une transition sortante de nom d'action *stop* et de taux $r_{stop}$ vers l'état $s_{stop}$. Ces transitions vont permettre de modéliser l'action de l'observateur stoppant son observation. L'hypothèse est donc faite que le comportement de l'observateur est distribué suivant une loi exponentielle de taux $r_{stop}$, indépendante du reste du système observé (puisque ce taux est le même quelque soit l'état du système). La *figure* 6.13 représente l'état $s_{stop}$ et les transitions $(stop, r_{stop})$ sur le graphe de dérivations du modèle PEPA défini par les trois déclarations suivantes :

$$P_1 \stackrel{def}{=} (a, x).P_2 \qquad P_2 \stackrel{def}{=} (b, y).P_3 \qquad P_3 \stackrel{def}{=} (c, z).P_1$$

avec comme terme racine $P_1$. Il peut sembler gênant d'inclure l'observateur dans le modèle. Cependant, il s'agit surtout d'une astuce mathématique dans la mesure où, au final, les calculs des espérances sont indépendants de la valeur $r_{stop}$. Étant donné le caractère assez "virtuel" de cet état et de ces transitions on ne les explicitera pas dans la définition de $G$. Autrement dit $s_{stop}$ n'est pas explicité dans $S$ et les transitions $(stop, r_{stop})$ ne sont pas explicitées dans $T$.

Cette hypothèse posée, les variables aléatoires $Z_v$ et $N_v$ convergent presque sûrement et possèdent les même définitions que précédemment (c'est-à-dire les mêmes définitions

[4] c'est-à-dire différentes de $\tau$

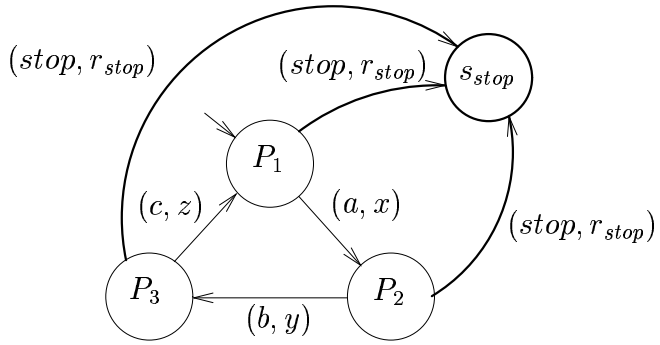



Figure 6.13: *Graphe de dérivations d'un modèle PEPA augmenté de l'état $s_{stop}$ et des transitions $(stop, r_{stop})$.*

que dans le cas d'un graphe de dérivations absorbant) :

$$Z_v(\omega) = \sum_{\zeta \in \mathbb{T}(v)} \int_0^\infty \mathbf{1}_{\{X_t(\omega) = \bullet\zeta\}} dt$$

$$N_v(\omega) = \lim_{\epsilon \to 0} \sum_{i=0}^{\infty} \mathbf{1}_{\{C^v_{i\epsilon}(\omega) \neq C^v_{(i+1)\epsilon}(\omega)\}}$$

### 6.3.1 Calcul des espérances conditionnelles de $Z_v$ et $N_v$

Si les définitions de $Z_v$ et $N_v$ restent identiques, leurs espérances conditionnelles, elles, se compliquent davantage. Ceci est dû au fait que les données d'observations sont des exécutions partiellement observables au lieu de simple temps d'absorption. Voici l'espérance conditionnelle de $Z_v$ sachant $\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop$ :

$$\mathbb{E}[Z_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{\zeta \in \mathbb{T}(v)} \int_0^\infty \mathbb{P}(X_t = \bullet\zeta | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$$

et l'espérance conditionnelle de $N_v$ sachant $\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop$ :

$$\mathbb{E}[N_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\lim_{\epsilon \to 0} \sum_{i=0}^{\infty} \mathbb{P}(C^v_{i\epsilon} \neq C^v_{(i+1)\epsilon} | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop) dt$$

Dans l'*annexe* C.3.2, on montre que l'espérance conditionnelle de $Z_v$ sachant $\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop$ est égale à l'expression suivante :

$$\mathbb{E}[Z_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{e}(v,t)\mathbf{Q}_{a_i}\mathbf{d}(t_{i+1})\mathbf{Q}_{a_{i+1}}\ldots\mathbf{d}(t_{m-1})\mathbf{1}}{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{d}(t_{m-1})\mathbf{1}}$$

où $\mathbf{1}$ est un vecteur de taille $|\mathrm{S}|$ rempli de 1. $\mathbf{d}$ et $\mathbf{e}$ sont des matrices de fonctions exprimables en tant que solution de systèmes d'équations différentielles ordinaires. Et, $\mathbf{Q}_a$ est la matrice contenant les taux de toutes les transitions d'action $a \in Act$ entre deux états (transitions réflexives incluses) :

$$\forall s, s' \in S \quad \mathbf{Q}_a(s,s') = \sum_{\substack{\zeta \in T \\ \bullet\zeta = s \\ \zeta\bullet = s' \\ act(\zeta) = a}} ev(\zeta)$$

De même on montre dans l'annexe C.3.5 que l'espérance conditionnelle de $N_v$ sachant $\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop$ est égale a l'expression analytique suivante :

$$\mathbb{E}[N_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots(\mathbf{f}(v,t_i)\mathbf{Q}_{a_i} + \mathbf{d}(t_i)\mathbf{Q}_{v,a_i})\mathbf{d}(t_{i+1})\mathbf{Q}_{a_{i+1}}\ldots\mathbf{d}(t_{m-1})\mathbf{1}}{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{d}(t_{m-1})\mathbf{1}}$$

où $\mathbf{f}$ est une matrice de fonctions exprimables comme la solution d'un systèmes d'équations différentielles ordinaires et $\mathbf{Q}_{v,a}$ est la matrice contenant les taux de transitions de toutes les transitions d'action $a \in Act$ et de taux $v \in V$ :

$$\mathbf{Q}_{v,a}(s,s') = \sum_{\substack{\zeta \in T \\ \bullet\zeta = s \\ \zeta\bullet = s' \\ act(\zeta) = a \\ rate(\zeta) = v}} ev(\zeta)$$

On va maintenant donner la définition des matrices de fonctions $\mathbf{d}$, $\mathbf{e}$ et $\mathbf{f}$ :

$$\begin{aligned}
\mathbf{d}(t) &= e^{t\mathbf{Q}_\tau} \\
\mathbf{e}(v,t) &= \textstyle\int_0^t e^{u\mathbf{Q}_\tau}\mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau}\,du \\
\mathbf{f}(v,t) &= \textstyle\int_0^t e^{u\mathbf{Q}_\tau}\mathbf{Q}_{v,\tau} e^{(t-u)\mathbf{Q}_\tau}\,du
\end{aligned}$$

où $\mathbf{Q}_\tau$ est la matrice contenant la somme des taux de transitions invisibles entre deux états moins la somme du taux de sorti de l'état sur la diagonale de la matrice :

$$\mathbf{Q}_\tau(s,s') = \begin{cases} \sum\limits_{\substack{\zeta\in T \\ \bullet\zeta=s \\ \zeta\bullet=s' \\ act(\zeta)=\tau}} ev(\zeta) - \sum\limits_{\substack{\zeta\in T \\ \bullet\zeta=s}} ev(\zeta) & \text{si } s=s' \\ \sum\limits_{\substack{\zeta\in T \\ \bullet\zeta=s \\ \zeta\bullet=s' \\ act(\zeta)=\tau}} ev(\zeta) & \text{sinon} \end{cases}$$

$\mathbf{C}_v : S \times S \mapsto \mathbb{N}$ est la matrice diagonale contenant pour chaque état le nombre de transitions sortantes de taux $v$ :

$$\mathbf{C}_v(s,s') = \begin{cases} |\{\zeta \in \mathbb{T}(v) | \bullet\zeta = s\}| & \text{si } s=s' \\ 0 & \text{sinon} \end{cases}$$

Pour résoudre les matrices de fonctions $\mathbf{d}$, $\mathbf{e}$ et $\mathbf{f}$ on vérifie qu'elles sont solutions des systèmes d'équations différentielles ordinaires suivants (voir l'*annexe* C.2 pour la vérification) :

$$\begin{array}{lll} \dot{\mathbf{d}}(t) = \mathbf{d}(t)\mathbf{Q}_\tau & \text{avec} & \mathbf{d}(0) = \mathbf{I} \\ \dot{\mathbf{e}}(v,t) = \mathbf{e}(v,t)\mathbf{Q}_\tau + \mathbf{d}(t)\mathbf{C}_v & \text{avec} & \mathbf{e}(v,0) = \mathbf{O} \\ \dot{\mathbf{f}}(v,t) = \mathbf{f}(v,t)\mathbf{Q}_\tau + \mathbf{d}(t)\mathbf{Q}_{v,\tau} & \text{avec} & \mathbf{f}(v,0) = \mathbf{O} \end{array}$$

Enfin on peut remarquer que la constante $r_{stop}$ n'apparaît pas dans les expressions analytiques des espérances conditionnelles de $Z_v$ et $N_v$. Ce qui confirme l'hypothèse d'indépendance de leur calcul par rapport au modèle de l'observateur.

### 6.3.2 La complexité de l'algorithme

Afin de calculer les espérances conditionnelles de $Z_v$ et de $N_v$ on a besoin de résoudre les fonctions $\mathbf{d}$, $\mathbf{e}$ et $\mathbf{f}$ pour chacun des temps $t_1$ à $t_m$ (pour chaque exécution partiellement observable de taille $m$). Une fois le résultat de ces fonctions connu le calcul s'opère suivant les définitions des expressions analytiques de ces espérances. Il faut résoudre $\mathbf{e}$ et $\mathbf{f}$ pour chaque $v \in V$, soit $2 \times |V|$, plus le système $\mathbf{d}$, soit en tout $2 \times |V| + 1$ systèmes différentiels. Chaque système consiste en des produits de matrices de complexité $|S|^3$. On note $I$ le nombre d'itérations des étapes $E$ et $M$ et $C$ le nombre de pas de calcul nécessaires pour résoudre les systèmes différentiels jusqu'à la plus grande valeur de temps, Alors la complexité en temps de l'algorithme est de l'ordre de :

$$I \times C \times |V| \times |S|^3$$

ce qui correspond à la complexité de l'algorithme précédant assortie du coefficient multiplicateur $|V|$. Une fois les fonctions $\mathbf{d}$, $\mathbf{e}$ et $\mathbf{f}$ connues on peut calculer les espérances conditionnelles de $Z_v$ et de $N_v$ mais pour cela on a besoin de mémoriser les résultats de

ces fonctions ($2 \times |V| + 1$ matrices carrées de dimension $S$) pour chaque instant $t_1$ à $t_m$ de chaque exécution partiellement observable. On suppose qu'il y a $n$ exécutions partiellement observables, chacune en moyenne de taille $m$, ainsi la complexité spatiale est de l'ordre de :

$$n \times m \times |V| \times |S|^2$$

### 6.3.3 Expérimentations

L'algorithme n'ayant pour le moment pas été implanté de façon optimale en temps et en espace dans EMPEPA, on propose simplement dans cette section d'illustrer son utilité à travers deux exemples simples de quelques états.

**Exemple 1 : retrouver les valeurs initiales d'un modèle**

Soir $V = \{x, y, z\}$ l'ensemble des variables du problème. Le modèle PEPA considéré est le suivant :

$$P_1 \stackrel{def}{=} (a, x).P_1 + (\tau, x).P_2$$
$$P_2 \stackrel{def}{=} (b, y).P_2 + (a, z).P_1$$

Le terme racine est $P_1$. Les variables ont été initialisées avec les valeurs :

$$x = 1 \quad y = 2 \quad z = 3$$

On a généré par simulation (à l'aide du logiciel EMPEPA) quatre exécutions partielles de tailles respectives 350, 8, 250 et 450. Ces tailles ont été choisies aléatoirement par EMPEPA. Voici par exemple des préfixes de chaque exécution (la deuxième exécution est complète) :

$$init \xrightarrow{0.786} b \xrightarrow{0.216} a \xrightarrow{0.746} a \xrightarrow{1.093} a \xrightarrow{1.362} a \xrightarrow{0.017} a \xrightarrow{0.068} a \xrightarrow{0.377} b \xrightarrow{0.046} b \xrightarrow{0.152} a \ldots stop$$
$$init \xrightarrow{0.769} b \xrightarrow{0.014} a \xrightarrow{0.019} a \xrightarrow{0.867} a \xrightarrow{0.789} a \xrightarrow{0.023} a \xrightarrow{0.006} a \xrightarrow{0.144} stop$$
$$init \xrightarrow{0.09} a \xrightarrow{0.063} a \xrightarrow{0.241} a \xrightarrow{0.364} a \xrightarrow{0.206} a \xrightarrow{1.355} a \xrightarrow{1.531} a \xrightarrow{0.893} a \xrightarrow{0.830} a \xrightarrow{0.824} b \ldots stop$$
$$init \xrightarrow{0.334} a \xrightarrow{1.065} a \xrightarrow{0.287} a \xrightarrow{0.312} a \xrightarrow{0.142} a \xrightarrow{0.667} a \xrightarrow{0.367} a \xrightarrow{0.39} a \xrightarrow{1.278} a \xrightarrow{2.543} a \ldots stop$$

Ensuite on a valué les variables de la manière suivante :

$$x = 3 \quad y = 1 \quad z = 2$$

et demandé à EMPEPA de trouver les valeurs de $x, y$ et $z$ qui maximisent la vraisemblance de ces quatre exécutions partielles. Après 10 itérations (quelques dizaines de secondes) EMPEPA a retourné les affectations suivantes :

$$x = 1.006635 \quad y = 2.025398 \quad z = 2.996108$$

C'est-à-dire, à quelques erreurs près, les valeurs qui avaient été choisies pour générer les quatre exécutions partielles.

**Exemple 2 : ne rien observer apporte aussi de la connaissance**

Cette technique permet de prendre en compte l'absence d'observation et c'est ce qu'illustre l'exemple suivant. On modélise un système composé de deux ordinateurs et d'un réseau. Il y a un ordinateur source, modélisé par le terme PEPA $Source$, qui envoie périodiquement un message sur le réseau à destination d'un autre ordinateur, modélisé par le terme PEPA $Dest$. Le réseau, modélisé par le terme $Net$, transporte le message depuis la source jusqu'à la destination suivant une simple distribution exponentielle. Le réseau a une certaine probabilité de tomber en panne et une probabilité nulle d'être réparé. La panne a pour conséquence d'arrêter les communications. L'ensemble des variables du problème est :

$$V = \{s, r, p\}$$

La variable $s$ correspond au taux de transition de l'envoie du message depuis la source. La variable $r$ correspond au taux de transition de la transmission du message sur le réseau. La variable $p$ correspond au taux de panne du réseau. Il y a deux actions, $i$ lorsque le message rentre dans le réseau, $o$ lorsque le message sort du réseau. Le modèle est défini à l'aide des trois termes PEPA suivants :

$$\begin{array}{lll} Source & \stackrel{def}{=} & (i, s).Source \\ Net & \stackrel{def}{=} & (i, \top).((o, r).Net + (\tau, p).nil) \\ Dest & \stackrel{def}{=} & (o, \top).Dest \end{array}$$

et du terme racine ci-dessous :

$$Source \underset{\{i\}}{\bowtie} Net \underset{\{o\}}{\bowtie} Dest$$

L'aspect intéressant de cet exemple est qu'il n'est pas possible de faire la distinction entre un retard du message et son absence due à une panne du réseau uniquement sur la base des observations $i$ et $o$. En revanche, il est possible de tenir le raisonnent suivant : plus on attend longtemps sans nouvelle, plus la chance que le réseau soit en panne est grande. Les résultats qui suivent montrent que l'algorithme reste efficace face à ce genre de situation.

Les variables de $V$ ont été initialisées avec les valeurs suivantes :

$$ev(s) = 1 \quad ev(r) = 2 \quad ev(p) = 0.1$$

On a généré 200 exécutions partiellement observables, puis changé les valeurs des variables par :

$$ev(s) = 3 \quad ev(r) = 1 \quad ev(p) = 2$$

Après 30 itérations (quelques dizaines de secondes) l'algorithme à convergé vers les valeurs suivantes, proche des valeurs initiales :

$$ev(s) = 1.02 \quad ev(r) = 1.98 \quad ev(p) = 0.101$$

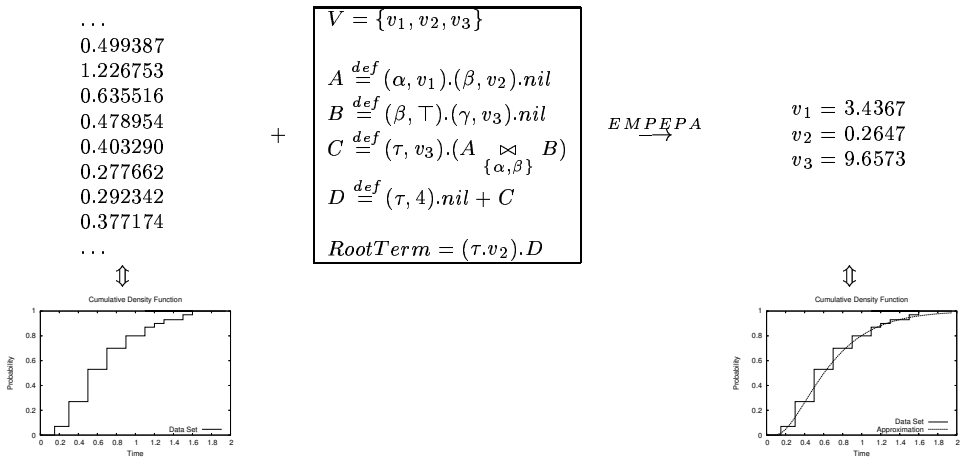



Figure 6.14: *Exemple d'utilisation de EMPEPA avec comme observations des temps absorption.*

## 6.4 EMPEPA

EMPEPA (provenant de la concaténation de EM et de PEPA) est un outil, développé dans le cadre de cette thèse, permettant de trouver les valeurs des taux inconnues d'un modèles PEPA de sorte que celui-ci maximise la vraisemblance d'un ensemble d'observations. La fonction de cet outil est donc d'améliorer le réalisme d'un modèle PEPA en accord avec des observations de la réalité. Ces observations peuvent être :

1. des temps d'absorption, voir la *section* 6.2 pour les détails concernant la description de ce problème.
2. des exécutions partiellement observables, voir la *section* 6.3 pour les détails concernant la description de ce problème.

La *figure* 6.14 donne un schémas d'utilisation de EMPEPA pour approcher une distribution décrite par un ensemble de temps d'absorption. A gauche de la flèche étiquetée EMPEPA sont indiquées les entrées de l'algorithme :

1. Une liste de temps d'absorption (en bas de cette liste correspond la distribution cumulative discrète de cet ensemble de temps d'absorption).
2. Une distribution de type phase décrite en PEPA contenant les variables $v_1$ à $v_3$.

A droite de la flèche est indiquée la sortie de l'algorithme : l'affectation de toutes les variables de sorte que la distribution décrite par le modèle PEPA approche au mieux la distribution des données observées. La *figure* 6.15 donne un schémas d'utilisation de EMPEPA sur un terme PEPA sans état absorbant et dont les observations sont des exécutions partiellement observables. A gauche de la flèche étiqueté EMPEPA sont indiquées les entrées de l'algorithme :

1. Une liste d'exécutions partiellement observables.
2. Un modèle PEPA sans état absorbant contenant les variables de $v_1$ à $v_3$.

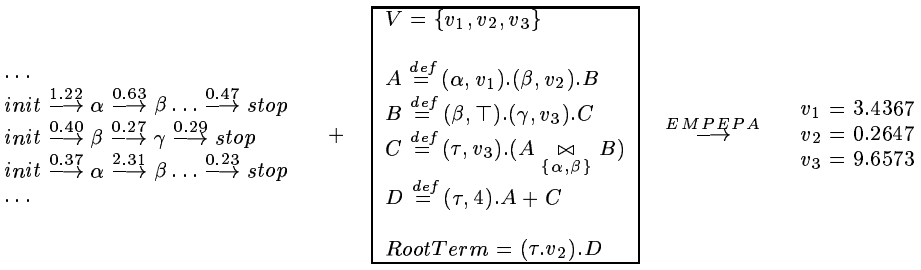



Figure 6.15: *Schémas d'un exemple d'utilisation de EMPEPA avec comme observations des exécutions partiellement observables.*

A droite de la flèche la sortie : l'affectation de toutes les variables de sorte que le modèle PEPA maximise l'ensemble des exécutions partiellement observables. EMPEPA a été codé en langage *C* respectant la norme ANSI de 1999. Il se compile et s'installe sans difficulté sur toute plateforme compatible UNIX grâce à l'utilisation des outils Automake et Autoconf. Une seule bibliothèque non standard (mais se trouvant sous forme de pacquage sur la plus pas des distributions Linux) est requise :

– la GNU Scientific Library (GSL)

En plus d'implanter les algorithmes décrits aux *sections* 6.2 et 6.3, il offre les fonctions de base permettant de simuler un modèle PEPA et d'écrire dans un fichier les résultats de ces simulations dans la même syntaxe que celle des observations, ce qui a permis de tester la validité de l'implantation.

Grâce à l'utilisation de la bibliothèque GSL, EMPEPA dispose de neuf méthodes de résolution pour calculer les espérances des étapes $E$ de l'algorithme, plus une dixième méthode, également développée dans le cadre de cette thèse, permettant d'approcher ces espérances par simulation (voir la *section* 6.4.1 pour plus de détails). Les neuf méthodes de résolution sont listées ci-dessous :

1. Runge-Kutta (2,3)
2. Runge-Kutta d'ordre 4
3. Runge-Kutta-Fehlberg (4,5)
4. Runge-Kutta Cash-Karp (4,5)
5. Runge-Kutta Prince-Dormand (8,9)
6. Runge-Kutta à points Gaussien d'ordre 2 implicite
7. Runge-Kutta à points Gaussien d'ordre 4 implicite
8. M=1 implicit Gear method
9. M=2 implicit Gear method

Les *annexes* D.1 et D.2 contiennent les détails concernant la syntaxe des modèles PEPA et des observations, et les différentes options disponibles en ligne de commande. EMPEPA

est un projet Open-source sous licence GPL hébergé par Sourceforge et en libre téléchargement à l'adresse internet indiquée ci-dessous.

`http ://sourceforge.net/projects/empepa`

### 6.4.1 Approcher $\mathbb{E}[\mathbf{Z}_v|\mathbf{Y}=\mathbf{y}]$ et $\mathbb{E}[\mathbf{N}_v|\mathbf{Y}=\mathbf{y}]$ par simulation

On propose, dans EMPEPA, une méthode supplémentaire afin de calculer, ou plus exactement d'approcher, les espérances conditionnelles de $\mathbf{Z}_v$ et $\mathbf{N}_v$ par des simulations. L'idée consiste à lancer un certain nombre de simulations et à compter les temps de séjours dans chaque état et le nombre de fois que chaque transition a été prise.

Une difficulté supplémentaire vient du fait que les espérances à approcher sont conditionnelles. Dès lors seules les simulations vérifiant ces conditions peuvent être retenue pour l'approximation de ces espérances. Malheureusement, la mesure de probabilité nulle des conditions impliquées ici, $\mathbf{Y}=\mathbf{y}$, ne laisse aucun espoir d'obtenir des simulations les respectant, c'est-à-dire ayant leur temps d'absorption exactement sur l'un des $y_i$ de $\mathbf{y}$.

Afin de contourner cette difficulté on a utilisé la propriété de Markov du processus et considéré que toute simulation arrivant au moins dans un certain état non absorbant à une date $y_i$ possède une certaine densité de probabilité de prendre une transition vers l'état absorbant. Cette densité de probabilité permet de pondérer l'importance des espérances de $\mathbf{Z}_v$ et $\mathbf{N}_v$ comptées pendant cette simulation par rapport aux autres simulations.

Le problème de cette méthode est que plus la probabilité d'atteindre un état non absorbant dans un temps $y_i$ de $\mathbf{y}$ sera faible plus le nombre de simulations nécessaire pour obtenir une bonne approximation devra être grand. La *table* 6.1 récapitule dans quelles situations cette méthode peut ou ne peut pas se révéler intéressante. De plus, afin de réussir à utiliser cette méthode, les conditions de la colonne de gauche doivent être vérifiées à chaque itération de l'algorithme, sous peine de ne jamais passer la prochaine itération car l'algorithme cherche en vain des simulations pertinentes pour ses approximations.

Dans la pratique cette méthode s'est révélée très efficace quand elle a été appliquée à des exemples construits artificiellement de sorte que les conditions de la colonne de gauche de la *table* 6.1 soient respectées. Dans une utilisation réaliste, telle celle qui est présentée dans le troisième exemple de la *section* 6.2.1, on n'a pas réussi à profiter de cette méthode car les conditions de la colonne de gauche n'étaient pas réalisées (des millions de simulations n'ont pas été suffisantes pour atteindre un état non absorbant quand il s'agissait des plus grandes valeurs de $\mathbf{y}$). Il est probable que, dans la plus part des cas réalistes, cette méthode ne soit pas applicable, à moins d'avoir, par chance, les conditions de la colonne de gauche réunies, auquel cas cette méthode pourra se révéler plus rapide qu'une méthode basée sur la résolution de systèmes d'équations différentielles ordinaires.

| Situations où cette méthode est intéressante | Situations où cette méthode n'est pas intéressante |
|---|---|
| Le modèle a une bonne probabilité d'atteindre tous les états et les transitions qui sont impliqués dans le calcul des espérances conditionnelles. | Certaine exécutions impliquées dans le calcul de certaines espérances conditionnelles ont une probabilité d'occurrence trop petite. |
| Les temps d'absorption sont suffisamment petits de sorte que la probabilité donnée par le modèle d'atteindre un état non absorbant à un temps d'absorption $y_i$ de $\mathbf{y}$ n'est pas négligeable. | La probabilité d'atteindre un état non absorbant dans un temps $y_i$ de $\mathbf{y}$ est trop petite. |

Table 6.1: *Tableau récapitulatif des conditions requises pour le bon fonctionnement de la méthode de calcul des espérances conditionnelles par simulations.*

## 6.5 Conclusion du chapitre

### 6.5.1 Résumé

Les méthodes d'approximations permettent d'obtenir des modèles plus réalistes que ceux construits manuellement. Lesquels manquent en contre partie de structure et d'intelligibilité. Afin de remédier à cela, on propose d'adapter la méthode d'approximation de Soren Asmussen, Olle Nerman et Marita Olsson [57] pour lui permettre d'opérer sur l'algèbre PEPA. Laquelle permet d'imposer une structure au modèle, les rendant plus intelligibles et offrant davantage de souplesse pour les utiliser et les transformer. On définit ensuite une généralisation de cet algorithme permettant de l'utiliser sur la base d'exécutions partiellement observables au lieu de temps d'absorption et sur des modèles PEPA avec ou sans état absorbant.

### 6.5.2 Limites et améliorations possibles

Ces deux algorithmes permettent de travailler sur des modèles PEPA dont les synchronisations se produisent exclusivement entre transitions actives et transition passives. De plus, leur complexité et la puissance actuelle des ordinateurs ne permet pas de les utiliser sur des modèles de plus de quelques centaines d'états (contre des millions d'états pour les algorithmes de model checking). Il serait donc intéressant d'envisager de les optimiser. Voici trois idées d'optimisations possibles (mais non certaines) :

– Utiliser la connaissance des calculs des espérances d'une itération à l'autre, (ce qui

aura peut-être pour conséquence de transférer la complexité du temps vers celle de l'espace).

- Réduire le nombre des données d'observations (et en particulier les grandes valeurs de temps d'absorption car elles augmentent le nombre de pas nécessaire pour la résolution des systèmes différentiels) pour ne garder que les plus significatives.
- Transformer les temps d'absorption (ou les temps d'observations entre deux actions) en moyenne, écart type, etc, c'est-à-dire en considérant les moments de la distribution des données au lieu des données brutes. L'avantage de considérer les moments est qu'il est possible de borner leur nombre utiles car une chaîne de Markov fini ne possède qu'un nombre fini de moments.

# Chapter 7

# Conclusion

Évaluer les performances d'une simulation distribuée temps réelle par l'utilisation du modèle checking probabiliste pose :

1. le problème de la modélisation probabiliste du système dans un modèle accepté par un model checker probabiliste,
2. le problème de la formalisation des contraintes de performance à évaluer dans une logique acceptée par un model checker probabiliste.

La principale difficulté rencontrée au cours des travaux de thèse fut celle de produire un modèle probabiliste réaliste. La formalisation des contraintes de performance ne pas posant de difficultés particulières, en tout cas pas dans cette étude.

L'utilisation d'algorithmes d'approximations a pu permettre de générer des modèles réalistes, aussi bien concernant la trajectoire du processus en fonction du temps qu'en fonction de différents paramètres (tailles des messages, qualité du réseau, rapidité des machines, etc). En revanche les modèles obtenus par ces algorithmes sont difficilement compréhensibles et manipulables par l'humain en charge de faire la modélisation.

Afin de remédier à ce problème, un algorithme permettant d'approcher des distributions de type phase à partir de temps d'absorption a été adapté pour fonctionner sur des modèles décrits en PEPA. La description du modèle en PEPA permet d'imposer une structure au modèle, de le rendre plus compréhensible qu'une chaîne de Markov et plus facile à manipuler. De cette manière il devient, par exemple, facile de remplacer un composant par un autre composant. L'algorithme a ensuite été généralisé afin de pouvoir opérer sur des modèles PEPA sans état absorbant et dont les données d'observation sont des exécutions partiellement observables. Ces deux algorithmes admettent les limites suivantes :

1. Ils ne fonctionnent que sur des termes PEPA dont les synchronisations s'effectuent entre transitions actives et passives.
2. Leur complexité algorithmique ne permet pour le moment pas de les utiliser sur des modèles de plus de quelques centaines d'états.

3. Ces algorithmes ne devinent pas la structure du modèle PEPA, seulement les valeurs des variables placées à l'intérieur du modèle PEPA prédéfini par l'utilisateur.

Les améliorations possibles associées à ces limites sont :

1. Étendre ces deux algorithmes à tous types de synchronisations.
2. Optimiser les calculs des espérances ou trouver une autre méthode plus efficace que la méthode EM pour ce problème.
3. Générer intégralement le modèle PEPA le plus vraisemblable permettant d'expliquer un ensemble d'exécutions partiellement observables.

La limite la plus difficile à surmonter est de loin la dernière. C'est aussi la plus motivante. Cela permettrait de déléguer à l'ordinateur la lourde tâche de modélisation, habituellement réservée au concepteur (ou au thésard...). Il faut noter que bien souvent un concepteur humain ne modélise pas un processus sur la seule base d'un ensemble d'observations. Il possède en général un ensemble d'hypothèses qui permettent de restreindre l'ensemble des modèles candidats pour expliquer ces observations. Une telle amélioration a donc davantage d'intérêt si elle est associée avec la prise en compte de ces hypothèses.

# Bibliography


[1] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Verifying continuous-time markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102, pages 269–276, USA, 1996. Springer Verlag.

[2] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for probabilistic real-time systems. In *Automata, Languages and Programming*, pages 115–126, 1991.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[4] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time markov chains by transient analysis. In *CAV*, pages 358–372, 2000.

[5] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–161, 1999.

[6] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3) :125–155, 1998.

[7] B. Baynat. *Théorie des files d'attente - des chaînes de Markov aux réseaux à forme produit.* Hermes Science, 2000.

[8] B. Béard, M. Bidoit, F. Laroussine, A. Petit, and P. Schnoebelen. *Vérification de logiciels : Techniques et outils du model-checking.* Vuibert informatique, 1999.

[9] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *In proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04).* LNCS 3185, 2004.

[10] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable ? In *Proc. 12th Int. Conf. Computer Aided Verification (CAV'2000), Chicago, IL, USA, July 2000*, volume 1855, pages 464–479. Springer, 2000.

[11] H. Bowman, J. Bryans, and J. Derrick. A model checking algorithm for stochastic systems, 2000.

[12] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, 1986.

[13] B. Bréholée and P. Siron. Certi : Evolutions of the ONERA RTI prototype. In *Fall 2002 Simulation Interoperability Workshop*, 2002.

[14] E. Cinlar. *Introduction to Stochastic Processes.* Prentice-Hall Inc., 1975.

[15] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Springer, editor, *Proc. Logics of Programs Workshop*, volume 131, pages 52–71, Yorktown Heights, New York, May 1981.

[16] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5) :1512–1542, September 1994.

[17] P. D'Argenio, J. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems, 1998.

[18] B. d'Ausbourg, J.-L. Bussenot, and P. Siron. Perfosim : A performance evaluation tool for HLA distributed simulations. In *Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'02)*, Fort Worth, Texas, USA, October 2002.

[19] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Soc. Ser. B.*, 39 :1–38, 1977.

[20] DMSO. High level architecture run-time infrastructure programmer's guide 1.3 version 5, RTI 1.3 distribution. Technical report, Departement of Defense, 1998.

[21] DMSO. High level archtecture interface specification, version 1.3. Technical report, Departement of Defense, 1998.

[22] J. L. Doob. *Stochastic Processes.* Wiley, New York, 1953.

[23] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proc. 14th ACM Symp. Theory of Computing (STOC'82)*, pages 169–180, San Francisco, CA, May 1982.

[24] N. Geisweiller. Approximation de distributions générales par des termes de l'algèbre PEPA. In *Formalisation des Activités Concurrentes*, 2005.

[25] N. Geisweiller. Fitting phase type distributions within PEPA model. In *Process Algebra and Stochastically Timed Activities*, 2005.

[26] N. Geisweiller. Toward fitting phase type distributions within PEPA model. In *Performability Modeling of Computer and Communication Systems*, 2005.

[27] N. Geisweiller and J. Bonté. Performance evaluation of a real-time simulation architecture using probabilistic model checking. In *Practical Applications of Stochastic Modelling*, 2004.

[28] V. Girardin and N. Limnios. *Probabilités.* Vuibert " Les Grands Cours ", 2001.

[29] M. Gross and A. Lentin. *Notions sur les Grammaires Formelles.* Gauthier-Villars, 1970.

[30] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5) :512–535, 1994.

[31] J. Herbrand. Recherche sur la théorie de la démonstration, 1930.

[32] J. Hillston. Compositional markovian modelling using a process algebra, 1995.

[33] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[34] C. Hoare. Communicating sequential processes. In *Comm. ACM*, pages 666–677, 1978.

[35] K. Matthes. Zur theorie der bedienungsprozesse. In *3rd Prague Conf. on Information Theory, Stat. Dec. Fns. and Random Processes*, pages 513–528, 1962.

[36] S. G. Krantz. Jensen's inequality. *Handbook of Complex Variables*, page 118, 1999.

[37] J.-L. Krivine. *Théorie des ensembles.* Cassini, 1998.

[38] V. Kulkarni. *Modeling and Analysis of Stochastic Systems.* Chapman and Hall, 1995.

[39] M. Kwiatkowska, G. Norman, and D. Parker. PRISM : Probabilistic symbolic model checker. In *Computer Performance Evaluation / TOOLS*, pages 200–204, 2002.

[40] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Lecture Notes in Computer Science*, 1601 :75–95, 1999.

[41] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. *Lecture Notes in Computer Science*, 1877 :123+, 2000.

[42] G. G. I. Lopez, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions : Model checking semi-markov chains. In *PAPM-PROBMIV*, pages 57–70, 2001.

[43] A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[44] M. Metivier. *Probabilités : Dix Leçons d'Introduction.* Ellipse, 1987.

[45] R. Milner. A calculus of communicating systems. In Springer, editor, *Lecture Notes in Computer Science*, volume 92, Berlin, 1980.

[46] A. Monfort. *Cours de Probabilités.* Economica, 1996.

[47] E. Nagel. *Le Théorème de Gödel.* Seuil, 1997.

[48] M. Narasimha, R. Cleaveland, and P. Iyer. Probabilistic temporal logics via the modal mu-calculus. In *Foundations of Software Science and Computation Structure*, pages 288–305, 1999.

[49] M. Neuts. Matrix-geometric solution in stochastic models : An algorithmic approach. *The Johns Hopkins University Press*, 1981.

[50] D. J. Newman and A. R. Reddy. Rational approximation to ex and tro related functions. *J. Approximation Theory*, 25 :21–30, 1979.

[51] J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.

[52] M. Olsson. The EMpht-programme. Technical report, Department of Mathematics, Chalmers Uniuversity of Technology, Göteborg, 1998.

[53] D. Parker, G. Norman, and M. Kwiatkowska. PRISM user's guide, 2004.

[54] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[55] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13 :45–60, 1981.

[56] R. Smullyan. *Les Théorèmes d'incomplétude de Gödel*. DUNOD, 1992.

[57] M. O. Soren Asmussen, Olle Nerman. Fitting phase-type distribution via the EM algorithm. *Scandinavian Journal of Statistics*, 23 :419–441, 1996.

[58] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model Checking for Real-Time Systems. In *7th. Symposium of Logics in Computer Science*, pages 394–406, Santa-Cruz, California, 1992. IEEE Computer Scienty Press.

[59] S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[60] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, B :133–191, 1990.

[61] A. Wald. *Sequential Analysis*. WILEY, 1947.

[62] C. Wu. On the convergence properties of the EM algorithm. *Ann. Statist.*, 11 :95–103, 1983.

[63] H. L. S. Younes. Ymer : A statistical model checker.

[64] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 223–235, Copenhagen, Denmark, July 2002. Springer.

[65] S. Yovine. Kronos : A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1/2) :123–133, October 1997.

# Appendix A

# Probabilité

Dans chacune des sections suivantes de cette annexe $X = (X_t)_{t\in\mathbb{R}}$ désigne une CMTC d'espace d'état $S$, de distribution initiale $\pi$ et de générateur infinitésimal $\mathbf{Q}$.

## A.1 Définition de la trajectoire d'une CMTC

La trajectoire de $X$ s'exprime à l'aide d'un système d'équations différentielles ordinaire dont la variable est le temps $t$. Pour trouver l'expression de ce système on doit chercher la limite de $\frac{\mathbb{P}(X_{t+\Delta t}=s)-\mathbb{P}(X_t=s)}{\Delta t}$ quand $\Delta t$ tend vers 0 :

$$\begin{aligned}
\mathbb{P}(X_{t+\Delta t}=s) &= \mathbb{P}(X_{t+\Delta t}=s|X_t=s)\times\mathbb{P}(X_t=s) \\
&\quad +\sum_{s'\neq s}\mathbb{P}(X_{t+\Delta t}=s|X_t=s')\times\mathbb{P}(X_t=s') \\
&= (1+\mathbf{Q}(s,s)\Delta t+o(\Delta t))\times\mathbb{P}(X_t=s) \\
&\quad +\sum_{s'\neq s}\mathbf{Q}(s',s)\Delta t+o(\Delta t))\times\mathbb{P}(X_t=s') \\
\frac{\mathbb{P}(X_{t+\Delta t}=s)-\mathbb{P}(X_t=s)}{\Delta t} &= \frac{(\mathbf{Q}(s,s)\Delta t+o(\Delta t))\times\mathbb{P}(X_t=s)}{\Delta t} \\
&\quad +\frac{\sum_{s'\neq s}(\mathbf{Q}(s',s)\Delta t+o(\Delta t))\times\mathbb{P}\{X_t=s'\}}{\Delta t}
\end{aligned}$$

On note $p_s(t)$ la probabilité $\mathbb{P}(X_t=s)$ et $\mathbf{p}(t)$ le vecteur ligne $(p_s(t))_{s\in S}$. Si on fait tendre $\Delta t$ vers 0 on obtient le système différentiel ordinaire suivant :

$$\forall s\in S\quad \dot{p_s}(t)=\frac{\partial p_s(t)}{\partial t}=\sum_{s'\in S}p_{s'}(t)\mathbf{Q}(s',s)$$

Ceci s'exprime sous forme du produit vecteur-matrice ci-dessous :

$$\dot{\mathbf{p}}(t)=\mathbf{p}(t)\mathbf{Q}$$

On peut vérifier que la solution de ce système différentiel est :

$$\mathbf{p}(t) = \pi e^{t\mathbf{Q}}$$

Où $e^{t\mathbf{Q}}$ est une exponentiation de matrice, $e^{t\mathbf{Q}} = \sum_{j=0}^{\infty} \frac{(t\mathbf{Q})^j}{j!}$.

## A.2 La Probabilité de rester dans un même état

Soit $C_{t_1,t_2}$ la variable aléatoire qui correspond au nombre de transitions tirées entre l'instant $t_1$ et l'instant $t_2$ du processus $X$. Tout d'abord on remarque que :

$$\lim_{\Delta t \to 0} \mathbb{P}(X_{t+\Delta t} = s | X_t = s) = \lim_{\Delta t \to 0} \mathbb{P}(X_{t+\Delta t} = s, C_{t,t+\Delta t} = 0 | X_t = s)$$

On cherche maintenant à exprimer la limite de $\frac{\mathbb{P}(X_{t+\Delta t}=s, C_{0,t+\Delta t}=0) - \mathbb{P}(X_t=s, C_{0,t}=0)}{\Delta t}$ quand $\Delta t$ tend vers 0 :

$$\begin{aligned}
\mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0) &= \mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0, C_{0,t} = 0) \\
&= \mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0, X_t = s, C_{0,t} = 0) \\
&\quad + \mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0, X_t \neq s, C_{0,t} = 0) \\
&= \mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0 | X_t = s, C_{0,t} = 0) \\
&\quad \times \mathbb{P}(X_t = s, C_{0,t} = 0) \\
&= \mathbb{P}(X_{t+\Delta t} = s, C_{0,t+\Delta t} = 0 | X_t = s) \\
&\quad \times \mathbb{P}(X_t = s, C_{0,t} = 0)
\end{aligned}$$

On note $\dot{p}_{Cs}(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}(X_{t+\Delta t}=s, C_{0,t+\Delta t}=0) - \mathbb{P}(X_t=s, C_{0,t}=0)}{\Delta t}$, et on déduit par un calcul analogue à celui de $\dot{p}_s(t)$ que $\dot{p}_{Cs}(t) = p_{Cs}(t)\mathbf{Q}(s,s)$. La solution de ce système différentiel ordinaire d'ordre 1 est $p_{Cs}(t) = p_{Cs}(0)e^{\mathbf{Q}(s,s)t}$.

## A.3 La probabilité d'atteindre un état donné sachant qu'il y a transition

On formalise cette probabilité de la manière suivante :

$$P(X_{t+\Delta t} = s' | X_{t+\Delta t} \neq s, X_t = s)$$

L'utilisation de la propriété de Markov de $X$ permet de l'exprimer de la façon suivante :

$$\begin{aligned}
P(X_{t+\Delta t} = s' | X_{t+\Delta t} \neq s, X_t = s) &= \frac{P(X_{t+\Delta t} = s', X_{t+\Delta t} \neq s, X_t = s)}{P(X_{t+\Delta t} \neq s, X_t = s)} \\
&= \frac{P(X_{t+\Delta t} = s', X_t = s)}{P(X_{t+\Delta t} \neq s, X_t = s)} \\
&= \frac{P(X_{t+\Delta t} = s' | X_t = s) \times P(X_t = s)}{P(X_{t+\Delta t} \neq s | X_t = s) \times P(X_t = s)} \\
&= \frac{P(X_{t+\Delta t} = s' | X_t = s)}{P(X_{t+\Delta t} \neq s | X_t = s)} \\
&= \frac{\mathbf{Q}(s, s') \times \Delta t + o_1(\Delta t)}{-\mathbf{Q}(s, s) \times \Delta t + o_2(\Delta t)} \\
&= \frac{\mathbf{Q}(s, s') + \frac{o_1(\Delta t)}{\Delta t}}{-\mathbf{Q}(s, s) + \frac{o_2(\Delta t)}{\Delta t}}
\end{aligned}$$

Donc si $\Delta t \to 0$ la probabilité est égale à $\frac{\mathbf{Q}(s,s')}{-\mathbf{Q}(s,s)}$

## A.4 CMTC et CMTD associée

Il est possible d'approcher une CMTC de distribution initiale $\pi$ et de générateur infinitésimal $\mathbf{Q}$ par une CMTD de distribution initiale identique et une matrice de transition $\mathbf{P} = \epsilon\mathbf{Q}$, où $\epsilon$ est un réel positif représentant le temps écoulé entre 2 transitions de la CMTD associée. On montre donc que la limite suivante est vérifiée pour tout $t \in \mathbb{R}_+$ :

$$\forall t \in \mathbb{R}_+ \quad \lim_{\epsilon \to 0} \pi(\mathbf{I} + \epsilon\mathbf{Q})^{[t/\epsilon]} = \pi e^{t\mathbf{Q}}$$

En effet :

$$\lim_{\epsilon \to 0} \frac{e^{\epsilon\mathbf{Q}} - \mathbf{I}}{\epsilon} = \mathbf{Q}$$

Aussi :

$$\begin{aligned}
\lim_{\epsilon \to 0} \pi(\mathbf{I} + \epsilon\mathbf{Q})^{[t/\epsilon]} &= \lim_{\epsilon \to 0} \pi(\mathbf{I} + \epsilon\frac{e^{\epsilon\mathbf{Q}} - \mathbf{I}}{\epsilon})^{[t/\epsilon]} \\
&= \lim_{\epsilon \to 0} \pi(e^{\epsilon\mathbf{Q}})^{[t/\epsilon]} \\
&= \lim_{\epsilon \to 0} \pi(e^{\epsilon\mathbf{Q}})^{t/\epsilon - \alpha} \qquad \text{avec } 0 \leq \alpha \leq 1 \\
&= \lim_{\epsilon \to 0} \pi(e^{t\mathbf{Q} - \epsilon\alpha\mathbf{Q}}) \\
&= \pi e^{t\mathbf{Q}}
\end{aligned}$$

# Appendix B

# Model Checking Probabiliste

## B.1 Résolution de l'opérateur Until non borné de PCTL

Le système d'équations linéaires 4.1 *page* 41 définit les probabilités suivantes :

$$\forall s \in S' \subseteq S,\ p_s = \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq \infty} \Psi_2} | X_0 = s)$$

Les deux équations suivantes s'obtiennent en partitionnant la probabilité suivant la variable $X_1$ et en appliquant la formule de Bayes :

$$\begin{aligned}
\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq \infty} \Psi_2} | X_0 = s) &= \sum_{s' \in S} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2}, X_1 = s' | X_0 = s) \\
&= \sum_{s' \in S} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_1 = s', X_0 = s) \times \mathbb{P}(X_1 = s' | X_0 = s)
\end{aligned}$$

On applique ensuite la propriété de Markov et on utilise le fait que $X$ est homogène dans le temps :

$$\begin{aligned}
&= \sum_{s' \in S} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_1 = s') \times \mathbb{P}(X_1 = s' | X_0 = s) \\
&= \sum_{s' \in S} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_0 = s') \times \mathbb{P}(X_1 = s' | X_0 = s)
\end{aligned}$$

On décompose la somme $\sum_{s' \in S}$ en trois sommes car $S = S' + Sat(\psi_2) + S''$ :

$$\begin{aligned}
= &\sum_{s' \in S'} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_0 = s') \times \mathbb{P}(X_1 = s' | X_0 = s) \\
&+ \sum_{s' \in Sat(\psi_2)} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_0 = s') \times \mathbb{P}(X_1 = s' | X_0 = s) \\
&+ \sum_{s' \in S''} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq \infty} \psi_2} | X_0 = s') \times \mathbb{P}(X_1 = s' | X_0 = s)
\end{aligned}$$

Enfin, puisque $\mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq\infty} \psi_2} | X_0 = s') = 1$ quand $s' \in Sat(\psi_2)$ et $\mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq\infty} \psi_2} | X_0 = s') = 0$ quand $s' \in S''$ on obtient :

$$\begin{aligned}\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq\infty} \Psi_2} | X_0 = s) &= \sum_{s' \in S'} \mathbb{P}(\sigma_{\psi_1 \mathcal{U}^{\leq\infty} \psi_2} | X_0 = s') \times \mathbb{P}(X_1 = s' | X_0 = s) \\ &\quad + \sum_{s' \in Sat(\psi_2)} \mathbb{P}(X_1 = s' | X_0 = s)\end{aligned}$$

## B.2 Résolution de l'opérateur Until borné pour CSL

On définit la variable aléatoire $T_0$ qui indique le temps de séjour dans le premier état de $X$ avant la première transition. On s'intéresse à $\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s)$ pour tout $s \in S'$, voir la *figure* 4.2 de la *page* 42 pour la définition de S'.

$$\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) = \int_0^\infty \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s) dt'$$

On sépare l'intégrale en deux intégrales de 0 à $t$ et de $t$ à l'infini.

$$\begin{aligned}\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) &= \int_0^t \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s) dt' \\ &\quad + \int_t^\infty \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s) dt'\end{aligned}$$

Puisque $s \in S'$ il est clair que $\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s)$ est nul pour tout $t' > t$, la seconde intégrale vaut donc 0.

$$\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) = \int_0^t \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s) dt'$$

On partitionne $\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t' | X_0 = s)$ suivant $X_{t'} = s'$ pour tout $s' \in S$ :

$$\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) = \int_0^t \sum_{s' \in S} \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2}, T_0 = t', X_{t'} = s' | X_0 = s) dt'$$

Puis on utilise le théorème de Bayes et la propriété de Markov pour obtenir :

$$\begin{aligned}\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) &= \int_0^t \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_{t'} = s', T_0 = t', X_0 = s) \\ &\quad \times \mathbb{P}(X_{t'} = s', T_0 = t' | X_0 = s) dt' \\ &= \int_0^t \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_{t'} = s') \times \mathbb{P}(X_{t'} = s', T_0 = t' | X_0 = s) dt'\end{aligned}$$

$X$ est homogène dans le temps, $\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_{t'} = s') = \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t-t'} \Psi_2} | X_0 = s')$. On remplace $\mathbb{P}(X_{t'} = s', T_0 = t' | X_0 = s)$ par son expression numérique (voir l'*annexe* A.2) pour obtenir enfin :

$$\mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t} \Psi_2} | X_0 = s) = \int_0^t e^{\mathbf{Q}(s,s)t'} \times \mathbf{Q}(s, s') \times \mathbb{P}(\sigma_{\Psi_1 \mathcal{U}^{\leq t-t'} \Psi_2} | X_0 = s')$$

# Appendix C

# L'algorithme EM

## C.1 Les fonctions de vraisemblance

Dans la suite on suppose que tout réel élevé à la puissance de 0 est 1, c'est-à-dire que $v^0 = 1 \ \forall v \in \mathbb{R}$.

### C.1.1 La densité de probabilité de $X = x$

Soit $x$ une exécution d'une distribution de type phase de générateur infinitésimal $\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \mathbf{t} \\ 0 \dots 0 & 0 \end{pmatrix}$. On suppose que l'exécution $x$ prend un nombre $m$ fini de transitions avant d'atteindre l'état absorbant. $x = s_1 \xrightarrow{t_1} \dots s_m \xrightarrow{t_m} nil$ dénote une telle exécution, $s_1$ est l'état l'initial $s_\star$ et $t_i$ est le temps de séjour dans l'état $s_i$ pour tout $i = 1, \dots, m$. La probabilité $\mathbb{P}(X = s_1 \xrightarrow{t_1} \dots s_m \xrightarrow{t_m} nil)$ étant de mesure nulle on admet la notation suivante :

$$\begin{aligned}
\mathbb{P}(X = s_1 \xrightarrow{t_1} \dots s_m \xrightarrow{t_m} nil) &= \frac{\partial^m \mathbb{P}(s_1 \xrightarrow{t_1} \dots s_m \xrightarrow{t_m} nil)}{\partial t_1 \dots \partial t_m} \\
&= e^{\mathbf{T}(s_1,s_1)t_1} \times \mathbf{T}(s_1, s_1) \times \frac{\mathbf{T}(s_1, s_2)}{-\mathbf{T}(s_1, s_1)} \\
&\quad \times e^{\mathbf{T}(s_2,s_2)t_2} \times \mathbf{T}(s_2, s_2) \times \frac{\mathbf{T}(s_2, s_3)}{-\mathbf{T}(s_2, s_2)} \\
&\quad \vdots \\
&\quad \times e^{\mathbf{T}(s_m,s_m)t_m} \times \mathbf{T}(s_m, s_m) \times \frac{\mathbf{t}(s_m)}{-\mathbf{T}(s_m, s_m)}
\end{aligned}$$

Ce qui se simplifie par :

$$\begin{aligned}\mathbb{P}(X = x) &= \mathbf{T}(s_1, s_2) \times e^{\mathbf{T}(s_1,s_1)t_1} \\ &\quad \times \mathbf{T}(s_2, s_3) \times e^{\mathbf{T}(s_2,s_2)t_2} \\ &\quad \vdots \\ &\quad \times \mathbf{t}(s_m) \times e^{\mathbf{T}(s_m,s_m)t_m}\end{aligned}$$

### C.1.2 La densité de probabilité de $Y = y$

$\mathbb{P}(Y = y)$ étant de mesure nulle on admet la notation suivante :

$$\begin{aligned}\mathbb{P}(Y = y) &= \frac{\partial \mathbb{P}(Y = y)}{\partial y} \\ &= \pi_{s_\star} e^{y\mathbf{T}}\mathbf{t}\end{aligned}$$

### C.1.3 La densité de probabilité de $\mathfrak{X} = \mathfrak{x}$

Soit $G = (S, Act, s_\star, T)$ un graphe de dérivations absorbant sans transitions passives avec un ensemble de variable $V$. Soit $\mathbf{E} : S \mapsto \mathbb{R}_+$ un vecteur contenant pour chaque $s \in S_{ph}$ le taux de probabilité de prendre une transition (y compris une transition réflexive) :

$$\text{Pour } s \in S_{ph}, \quad \mathbf{E}(s) = \sum_{\substack{\zeta \in T, \\ \bullet\zeta = s}} ev(\zeta)$$

Soit $\mathfrak{X}$ le processus stochastique décrivant complètement le graphe de dérivations $G$. Une exécution $\mathfrak{x}$ de $\mathfrak{X}$ se note :

$$\mathfrak{x} = s_1 \overset{(t_1,\zeta_1)}{\longrightarrow} \dots s_i \overset{(t_i,\zeta_i)}{\longrightarrow} \dots s_m$$

La probabilité $\mathbb{P}(\mathfrak{X} = \mathfrak{x})$ étant de mesure nulle on admet la notation suivante :

$$\mathbb{P}(\mathfrak{X} = \mathfrak{x}) = \frac{\partial^{m-1}\mathbb{P}(\mathfrak{X} = s_1 \overset{(t_1,\zeta_1)}{\longrightarrow} \dots s_i \overset{(t_i,\zeta_i)}{\longrightarrow} \dots s_m)}{\partial t_1 \dots \partial t_{m-1}}$$

dont l'expression est la suivante :

$$\begin{aligned}\mathbb{P}(\mathfrak{X} = \mathfrak{x}) &= ev(\zeta_1) \times e^{-\mathbf{E}(s_1)t_1} \\ &\quad \times ev(\zeta_2) \times e^{-\mathbf{E}(s_2)t_2} \\ &\quad \vdots \\ &\quad \times ev(\zeta_m) \times e^{-\mathbf{E}(s_m)t_m}\end{aligned}$$

## C.2 Systèmes différentiels

**Pour** $\mathbf{c}^s(t) = \int_0^t \pi_\star e^{u\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du$

Nous allons vérifier que $(\mathbf{c}^s)_{s\in S_{ph}}$ est bien solution du système $\forall s \in S_{ph} \quad \dot{\mathbf{c}}^s(t) = \mathbf{T}\mathbf{c}^s(t) + a_s(t)\mathbf{t}$ avec $\mathbf{c}^s(0) = 0$. Nous allons dériver $\mathbf{c}^s$ :

$$\begin{aligned}
\dot{\mathbf{c}}^s(t) &= \lim_{h\to 0} \frac{\int_0^{t+h} \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t+h-u)\mathbf{T}} \mathbf{t} du - \int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du}{h} \\
&= \lim_{h\to 0} \frac{\int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t+h-u)\mathbf{T}} \mathbf{t} du + \int_t^{t+h} \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t+h-u)\mathbf{T}} \mathbf{t} du - \int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du}{h} \\
&= \lim_{h\to 0} \frac{\int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{h\mathbf{T}} e^{(t-u)\mathbf{T}} \mathbf{t} du + \int_t^{t+h} \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t+h-u)\mathbf{T}} \mathbf{t} du - \int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du}{h} \\
&= \lim_{h\to 0} \frac{e^{h\mathbf{T}} - \mathbf{I}}{h} \int_0^t \pi_{s\star} e^{t\mathbf{T}} \pi'_s e^{(t-u)\mathbf{T}} \mathbf{t} du + \pi_{s\star} e^{t\mathbf{T}} \pi'_s \mathbf{t}
\end{aligned}$$

Puisque $\lim_{h\to 0} \frac{e^{h\mathbf{T}}-\mathbf{I}}{h} = \mathbf{T}$ on vérifie bien que $\dot{\mathbf{c}}^s(t) = \mathbf{T}\mathbf{c}^s(t) + a_s(t)\mathbf{t}$, de plus l'intégrale entre 0 et 0 est nulle donc $\mathbf{c}^s(0) = 0$.

**Pour** $\mathbf{d}(t) = e^{t\mathbf{Q}_\tau}$

$$\dot{\mathbf{d}}(t) = \frac{\partial \mathbf{d}(t)}{\partial t} = e^{t\mathbf{Q}_\tau} \mathbf{Q}_\tau$$

aussi

$$\dot{\mathbf{d}}(t) = \mathbf{d}(t)\mathbf{Q}_\tau \quad \text{avec} \quad \mathbf{d}(0) = \mathbf{I}$$

**Pour** $\mathbf{e}(v,t) = \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} du$

$$\begin{aligned}
\dot{\mathbf{e}}(v,t) &= \lim_{h\to 0} \frac{\int_0^{t+h} e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t+h-u)\mathbf{Q}_\tau} du - \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} du}{h} \\
&= \lim_{h\to 0} \frac{\int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} e^{h\mathbf{Q}_\tau} du + \int_t^{t+h} e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t+h-u)\mathbf{Q}_\tau} du - \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} du}{h} \\
&= \lim_{h\to 0} \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} \frac{e^{h\mathbf{Q}_\tau} - \mathbf{I}}{h} du + \frac{\int_t^{t+h} e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t+h-u)\mathbf{Q}_\tau} du}{h} \\
&= \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} \mathbf{Q}_\tau du + e^{t\mathbf{Q}_\tau} \mathbf{C}_v \\
&= \int_0^t e^{u\mathbf{Q}_\tau} \mathbf{C}_v e^{(t-u)\mathbf{Q}_\tau} du \mathbf{Q}_\tau + \mathbf{d}(t)\mathbf{C}_v
\end{aligned}$$

Nous obtenons donc :

$$\dot{\mathbf{e}}(v,t) = \mathbf{e}(v,t)\mathbf{Q}_\tau + \mathbf{d}(t)\mathbf{C}_v \quad \text{avec} \quad \mathbf{e}(v,0) = \mathbf{O}$$

**Pour** $\mathbf{f}(v,t) = \int_0^t e^{u\mathbf{Q}_\tau}\mathbf{Q}_{v,\tau}e^{(t-u)\mathbf{Q}_\tau}du$

On reprend le même raisonnement qu'avec $\mathbf{e}(v,t)$ et on obtient :

$$\dot{\mathbf{f}}(v,t) = \mathbf{f}(v,t)\mathbf{Q}_\tau + \mathbf{d}(t)\mathbf{Q}_{v,\tau} \quad \text{avec} \quad \mathbf{f}(v,0) = \mathbf{O}$$

## C.3 Calcul des espérances conditionnelles

### C.3.1 Espérance conditionnelle du temps de séjour dans un état sachant un temps d'absorption

Soit $X=(X_t)_{t\in\mathbb{R}_+}$ une distribution de type phase de distribution initiale $\pi$ de générateur infinitésimal de phases $\mathbf{T}$ et de vecteur de sortie $\mathbf{t}$. Soit $Z_s$, pour tout $s \in S_{ph}$ l'ensemble des variables aléatoires désignant le temps total de séjour dans l'état $s$ du processus $X$, sachant que le temps d'absorption est $y$.

$$[Z_s|Y=y] = \int_0^\infty \mathbb{P}(X_t = s|Y=y)dt$$

Après $y$ le processus est dans l'état absorbant $nil$, puisque $nil \notin S_{ph}$ on intègre seulement entre 0 et $y$. L'équation suivante est obtenue en utilisant la propriété de Markov :

$$[Z_s|Y=y] = \frac{\int_0^y \mathbb{P}(X_t=s)\mathbb{P}(Y=y|X_t=s)dt}{\mathbb{P}(Y=y)}$$

Ensuite on remplace ces probabilités par leur expressions analytiques en utilisant la définition d'une chaîne de Markov (voir la *section* 3.2.3) :

$$\begin{aligned}[Z_s|Y=y] &= \frac{\int_0^y \pi e^{t\mathbf{T}}\pi_s' \pi_s e^{(y-t)\mathbf{T}}\mathbf{t}dt}{\pi e^{y\mathbf{T}}\mathbf{t}} \\ &= \frac{\pi_s \int_0^y \pi e^{t\mathbf{T}}\pi_s' e^{(y-t)\mathbf{T}}\mathbf{t}dt}{\pi e^{y\mathbf{T}}\mathbf{t}}\end{aligned}$$

Finalement :

$$[Z_s|Y=y] = \frac{c_s^s(y)}{\pi\mathbf{b}(y)}$$

### C.3.2 Espérance conditionnelle du temps de séjour dans un état connaissant une exécution partiellement observable

Soit $G = (S, Act, s_\star, T)$ un graphe de dérivations complété de l'état $s_{stop}$ et des transitions $(stop, r_{stop})$ allant vers cet état. Soit $V$ l'ensemble des variables de $G$.

$$\mathbb{E}[Z_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{\zeta \in \mathbb{T}(v)} \int_0^\infty \mathbb{P}(X_t = \bullet\zeta \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop) dt$$

Soient $\theta_i = \sum_{j=1}^{i} t_j$ pour tout $i = 1, \ldots, m$, $\theta_0 = 0$. $\mathbb{P}(\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$ est une notation pour :

$$\frac{\partial \ldots \partial \mathbb{P}(\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)}{\partial t_1 \ldots \partial t_m}$$

$$=$$

$$\lim_{h_1 \to 0} \ldots \lim_{h_m \to 0} \frac{\mathbb{P}\left( \begin{array}{l} C_0^{Act \cup \{stop\}} = C_{\theta_1 - h_1}^{Act \cup \{stop\}}, C_{\theta_1 - h_1}^{a_1} \neq C_{\theta_1}^{a_1}, \\ \vdots \\ C_{\theta_{m-1}}^{Act \cup \{stop\}} = C_{\theta_m - h_m}^{Act \cup \{stop\}}, C_{\theta_m - h_m}^{stop} \neq C_{\theta_m}^{stop}) \end{array} \right)}{h_1 \times \ldots \times h_m}$$

Afin d'éviter des notations trop encombrantes on dénotera l'évènement :

$$\begin{array}{l} \lim_{h_1 \to 0} C_0^{Act \cup \{stop\}} = C_{\theta_1 - h_1}^{Act \cup \{stop\}}, C_{\theta_1 - h_1}^{a_1} \neq C_{\theta_1}^{a_1}, \\ \vdots \\ \lim_{h_m \to 0} C_{\theta_{m-1}}^{Act \cup \{stop\}} = C_{\theta_m - h_m}^{Act \cup \{stop\}}, C_{\theta_m - h_m}^{stop} \neq C_{\theta_m}^{stop} \end{array}$$

de la façon suivante :

$$\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \xrightarrow{stop} \theta_m$$

De la même manière quand on voudra décrire que l'exécution ne prend pas de transition avec actions visibles entre $\theta$ et $\theta'$ on utilisera la notation :

$$\theta \xrightarrow{\tau^*} \theta'$$

ce qui correspond à l'évènement $C_\theta^{Act \cup \{stop\}} = C_{\theta'}^{Act \cup \{stop\}}$. On cherche l'expression analytique de l'espérance conditionnelle de $Z_v$ :

$$\mathbb{E}[Z_v^{\theta_{i-1}, \theta_i} \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{\zeta \in \mathbb{T}(v)} \int_0^\infty \mathbb{P}(X_t = \bullet\zeta | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop) dt$$

On décompose l'intégrale :

$$\sum_{\zeta \in \mathbb{T}(v)} \sum_{i=1}^{m} \int_{\theta_{i-1}}^{\theta_i} \mathbb{P}(X_t = \bullet\zeta | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop) dt$$

et on cherche l'expression $\mathbb{P}(X_t = s | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$ pour $\theta_{i-1} \leq t < \theta_i$ et $s \in S$. L'égalité qui suit résulte de l'utilisation de la formule de Bayes :

$$\mathbb{P}(X_t = s | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$$

$$=$$

$$\frac{\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \theta_{i-1} \xrightarrow{\tau^*} t, X_t = s) \mathbb{P}(\mathfrak{Y} = t \xrightarrow{a_i} \ldots \xrightarrow{stop} \theta_m | \mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \theta_{i-1} \xrightarrow{\tau^*} t, X_t = s)}{\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \xrightarrow{stop} \theta_m)}$$

On simplifie l'expression grâce à la propriété de Markov :

$$\frac{\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \theta_{i-1} \xrightarrow{\tau^*} t, X_t = s) \mathbb{P}(\mathfrak{Y} = t \xrightarrow{a_i} \ldots \xrightarrow{stop} \theta_m | X_t = s)}{\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \xrightarrow{stop} \theta_m)}$$

L'expression du numérateur s'obtient par le même type de manipulations, formule de Bayes et propriété de Markov :

$$\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{stop} \theta_m)$$

$$=$$

$$\sum_{s_m \in S} \mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{a_{m-1}} \theta_{m-1}, X_t = s_m) \mathbb{P}(\mathfrak{Y} = \theta_{m-1} \xrightarrow{stop} \theta_m | X_t = s_m)$$

On applique le même raisonnement à $\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{a_{m-1}} \theta_{m-1}, X_t = s_m)$, puis au membre gauche du résultat obtenu et ainsi de suite :

$$\begin{aligned}
\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{stop} \theta_m) &= \sum_{s_1 \in S} \mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1, X_{\theta_1} = s_1) \\
&\times \sum_{s_2 \in S} \mathbb{P}(\mathfrak{Y} = \theta_1 \xrightarrow{a_2} \theta_2, X_{\theta_2} = s_2 | X_{\theta_1} = s_1) \\
&\vdots \\
&\times \mathbb{P}(\mathfrak{Y} = \theta_{m-1} \xrightarrow{stop} \theta_m | X_{\theta_{m-1}} = s_{m-1})
\end{aligned}$$

Par le même raisonnement on déduit les deux égalités suivantes :

$$\begin{aligned}\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \theta_{i-1} \xrightarrow{\tau^*} t, X_t = s) &= \sum_{s_1 \in S} \mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1, X_{\theta_1} = s_1) \\ &\times \sum_{s_2 \in S} \mathbb{P}(\mathfrak{Y} = \theta_1 \xrightarrow{a_2} \theta_2, X_{\theta_2} = s_2 | X_{\theta_1} = s_1) \\ &\vdots \\ &\times \mathbb{P}(\mathfrak{Y} = \theta_{i-1} \xrightarrow{\tau^*} t, X_t = s | X_{\theta_{i-1}} = s_{i-1})\end{aligned}$$

$$\begin{aligned}\mathbb{P}(\mathfrak{Y} = t \xrightarrow{a_i} \ldots \xrightarrow{stop} \theta_m | X_t = s) &= \sum_{s_i \in S} \mathbb{P}(\mathfrak{Y} = t \xrightarrow{a_i} \theta_i, X_{\theta_i} = s_i | X_t = s) \\ &\times \sum_{s_{i+1} \in S} \mathbb{P}(\mathfrak{Y} = \theta_i \xrightarrow{a_{i+1}} \theta_{i+1}, X_{\theta_{i+1}} = s_{i+1} | X_{\theta_i} = s_i) \\ &\vdots \\ &\times \sum_{s_{m-1} \in S} \mathbb{P}(\mathfrak{Y} = \theta_{m-1} \xrightarrow{stop} \theta_m | X_{\theta_{m-1}} = s_{m-1})\end{aligned}$$

On peut déduire les expressions analytiques des trois probabilités suivantes en utilisant la propriété de Markov de $\mathfrak{X}$ :

$$\mathbb{P}(\mathfrak{Y} = \theta \xrightarrow{\tau^*} \theta', X_{\theta'} = s' | X_\theta = s) = \pi_s e^{(\theta'-\theta)\mathbf{Q}_\tau^{stop}} \pi'_{s'}$$

$$\mathbb{P}(\mathfrak{Y} = \theta \xrightarrow{a} \theta', X_{\theta'} = s' | X_\theta = s) = \pi_s e^{(\theta'-\theta)\mathbf{Q}_\tau^{stop}} \mathbf{Q}_a \pi'_{s'}$$

$$\mathbb{P}(\mathfrak{Y} = \theta \xrightarrow{stop} \theta' | X_\theta = s) = \pi_s e^{(\theta'-\theta)\mathbf{Q}_\tau^{stop}} \mathbf{q}_{stop}$$

où $\mathbf{Q}_a$ est une matrice contenant les taux de transitions de prendre une transition de nom d'action $a \in Act$ entre deux états (comme définit à la *page* 104 *section* 6.3.1), et $\mathbf{Q}_\tau^{stop}$ est la matrice du générateur infinitésimal de $G$ considérant que seules des transitions d'action invisible sont tirées (c'est-à-dire à l'exclusion des actions dont le nom appartient à $Act$ ou est égale à $stop$) :

$$\mathbf{Q}_\tau^{stop}(s, s') = \begin{cases} -r_{stop} + \sum_{\substack{\zeta \in T \\ \bullet\zeta = s \\ \zeta\bullet = s' \\ act(\zeta) = \tau}} ev(\zeta) - \sum_{\bullet\zeta = s} ev(\zeta) & \text{si } s = s' \\ \sum_{\substack{\zeta \in T \\ \bullet\zeta = s \\ \zeta\bullet = s' \\ act(\zeta) = \tau}} ev(\zeta) & \text{sinon} \end{cases}$$

On peut déjà remarquer que :

$$\mathbf{Q}_\tau^{stop} = \mathbf{Q}_\tau - r_{stop}\mathbf{I} \tag{C.1}$$

où $\mathbf{Q}_\tau$ est définit à la *page* 104 *section* 6.3.1. Ce détail aura son importance quand nous montrerons que l'expression de l'espérance conditionnelle ne dépend pas de $r_{stop}$. Nous pouvons maintenant formuler l'expression analytique de l'espérance conditionnelle de $Z_v$ sachant $\mathfrak{Y}$ :

$$\mathbb{E}[Z_v \,|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{\zeta \in \mathbb{T}(v)} \sum_{i=1}^{m} \int_{\theta_{i-1}}^{\theta_i} \frac{\begin{array}{l} \sum_{s_1 \in S} \pi_{s_\star} e^{t_1 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_1} \pi'_{s_1} \\ \times \sum_{s_2 \in S} \pi_{s_1} e^{t_2 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_2} \pi'_{s_2} \\ \vdots \\ \times \pi_{s_{i-1}} e^{(t-\theta_{i-1}) \mathbf{Q}_\tau^{stop}} \pi'_s \\ \times \sum_{s_i \in S} \pi_{\bullet\zeta} e^{(\theta_i - t) \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_i} \pi'_{s_i} \\ \times \sum_{s_{i+1} \in S} \pi_{s_i} e^{t_{i+1} \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_{i+1}} \pi'_{s_{i+1}} \\ \vdots \\ \times \pi_{s_{m-1}} e^{t_m \mathbf{Q}_\tau^{stop}} \mathbf{q}_{stop} \end{array}}{\begin{array}{l} \sum_{s_1 \in S} \pi_{s_\star} e^{t_1 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_1} \pi'_{s_1} \\ \times \sum_{s_2 \in S} \pi_{s_1} e^{t_2 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_2} \pi'_{s_2} \\ \vdots \\ \times \pi_{s_{m-1}} e^{t_m \mathbf{Q}_\tau^{stop}} \mathbf{q}_{stop} \end{array}} dt$$

Que l'on peut réécrire de la manière suivante :

$$\sum_{\zeta \in \mathbb{T}(v)} \sum_{i=1}^{m} \int_{\theta_{i-1}}^{\theta_i} \frac{\begin{array}{l} \pi_{s_\star} e^{t_1 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_1} \\ \times \sum_{s_1 \in S} \pi'_{s_1} \pi_{s_1} e^{t_2 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_2} \\ \vdots \\ \times \sum_{s_{i-1} \in S} \pi'_{s_{i-1}} \pi_{s_{i-1}} e^{(t-\theta_{i-1}) \mathbf{Q}_\tau^{stop}} \pi'_{\bullet\zeta} \\ \times \pi_s e^{(\theta_i - t) \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_i} \\ \sum_{s_i \in S} \pi'_{s_i} \pi_{s_i} e^{t_{i+1} \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_{i+1}} \\ \vdots \\ \times \sum_{s_m \in S} \pi'_{s_m} \pi_{s_m} e^{t_m \mathbf{Q}_\tau^{stop}} \mathbf{q}_{stop} \end{array}}{\begin{array}{l} \pi_{s_\star} e^{t_1 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_1} \\ \times \sum_{s_1 \in S} \pi'_{s_1} \pi_{s_1} e^{t_2 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_2} \\ \vdots \\ \times \sum_{s_m \in S} \pi'_{s_m} \pi_{s_m} e^{t_m \mathbf{Q}_\tau^{stop}} \mathbf{q}_{stop} \end{array}} dt$$

Étant donnée que $\sum_{s\in S}\pi'_s\pi_s$ est égal à la matrice identité on peut simplifier l'expression par :

$$\sum_{\zeta\mathbb{T}(v)}\sum_{i=1}^{m}\int_{\theta_{i-1}}^{\theta_i}\frac{\begin{array}{l}\pi_{s_\star}e^{t_1\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_2}\times\ldots\times e^{(t-\theta_{i-1})\mathbf{Q}_\tau}\pi'_\zeta\\ \times\pi_s e^{(\theta_i-t)\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_i}\times e^{t_{i+1}\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_{i+1}}\times\ldots\times e^{t_m\mathbf{Q}_\tau^{stop}}\mathbf{q}_{stop}\end{array}}{\pi_{s_\star}e^{t_1\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_2}\times\ldots\times e^{t_m\mathbf{Q}_\tau^{stop}}\mathbf{q}_{stop}}dt$$

On peut réécrire l'expression de la manière suivante (on modifie la portée de l'intégrale de sorte qu'elle ne comprenne que les terme qui dépendent de $t$, et on élimine la première somme en utilisant la matrice $\mathbf{C}_v$ définie à la *page* 104 *section* 6.3.1) :

$$\sum_{i=1}^{m}\frac{\begin{array}{l}\pi_{s_\star}e^{t_1\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_2}\times\ldots\times e^{t_{i-1}\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_{i-1}}\\ \times\int_{\theta_{i-1}}^{\theta_i}e^{(t-\theta_{i-1})\mathbf{Q}_\tau^{stop}}\mathbf{C}_v e^{(\theta_i-t)\mathbf{Q}_\tau^{stop}}dt\\ \times\mathbf{Q}_{a_i}\times e^{t_{i+1}\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_{i+1}}\times\ldots\times e^{t_m-\mathbf{Q}_\tau^{stop}}\mathbf{q}_{stop}\end{array}}{\pi_{s_\star}e^{t_1\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau^{stop}}\mathbf{Q}_{a_2}\times\ldots\times e^{t_m\mathbf{Q}_\tau^{stop}}\mathbf{q}_{stop}}$$

L'équation C.1 de la *page* 133 nous permet de réécrire $e^{t_i\times\mathbf{Q}_\tau^{stop}}$ de la façon suivante :

$$e^{t_i\times\mathbf{Q}_\tau^{stop}}=e^{t_i\times\mathbf{Q}_\tau-t_i\times r_{stop}\times\mathbf{I}}=e^{t_i\times\mathbf{Q}_\tau}\times e^{-t_i\times r_{stop}\times\mathbf{I}}$$

De plus, on peut remarquer que $e^{-t_i\times r_{stop}\times\mathbf{I}}$ est égal à $e^{-t_i\times r_{stop}}\times\mathbf{I}$ car l'exponentiation d'une matrice diagonale est la matrice diagonale constituée de l'exponentielle de chacune des composantes. De la même manière on peut vérifier que :

$$\int_{\theta_{i-1}}^{\theta_i}e^{(t-\theta_{i-1})\mathbf{Q}_\tau^{stop}}\mathbf{C}_v e^{(\theta_i-t)\mathbf{Q}_\tau^{stop}}dt=e^{-t_i\times r_{stop}}\times\int_{\theta_{i-1}}^{\theta_i}e^{(t-\theta_{i-1})\mathbf{Q}_\tau}\mathbf{C}_v e^{(\theta_i-t)\mathbf{Q}_\tau}dt$$

Il y a donc $m$ termes $e^{-t_i\times r_{stop}}$ au numérateur et au dénominateur de la fraction, ces termes se compensent et disparaissent de l'expression :

$$\sum_{i=1}^{m}\frac{\begin{array}{l}\pi_{s_\star}e^{t_1\mathbf{Q}_\tau}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau}\mathbf{Q}_{a_2}\times\ldots\times e^{t_{i-1}\mathbf{Q}_\tau}\mathbf{Q}_{a_{i-1}}\\ \times\int_{\theta_{i-1}}^{\theta_i}e^{(t-\theta_{i-1})\mathbf{Q}_\tau}\mathbf{C}_v e^{(\theta_i-t)\mathbf{Q}_\tau}dt\\ \times\mathbf{Q}_{a_i}\times e^{t_{i+1}\mathbf{Q}_\tau}\mathbf{Q}_{a_{i+1}}\times\ldots\times e^{t_m-\mathbf{Q}_\tau}\mathbf{q}_{stop}\end{array}}{\pi_{s_\star}e^{t_1\mathbf{Q}_\tau}\mathbf{Q}_{a_1}\times e^{t_2\mathbf{Q}_\tau}\mathbf{Q}_{a_2}\times\ldots\times e^{t_m\mathbf{Q}_\tau}\mathbf{q}_{stop}}$$

Enfin on remplace les termes de l'expression par les fonctions $\mathbf{d}$, $\mathbf{e}$ et $\mathbf{f}$ (voir l'*annexe* C.2) :

$$\sum_{i=1}^{m}\frac{\pi_{s_\star}\mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{e}(v,t_i)\mathbf{Q}_{a_i}\mathbf{d}(t_{i+1})\mathbf{Q}_{a_{i+1}}\ldots\mathbf{d}(t_m)\mathbf{q}_{stop}}{\pi_{s_\star}\mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{d}(t_m)\mathbf{q}_{stop}}$$

Nous avons posé comme hypothèse que le moment où l'observateur stoppe l'observation est indépendant du processus observé $\mathfrak{X}$ ce qui a pour conséquence que le vecteur $\mathbf{q}_{stop}$ a

la même valeur pour toutes ses composantes, aussi $\mathbf{q}_{stop} = r_{stop} \times \mathbf{1}$ où $\mathbf{1}$ est le vecteur contenant la valeur 1 dans chacune de ses composantes. Étant au numérateur et au dénominateur de la fraction la constante $r_{stop}$ s'élimine. Nous n'avons donc pas besoin de $\mathbf{q}_{stop}$ pour exprimer l'espérance de $Z_v$ :

$$\mathbb{E}[Z_v | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{e}(v,t_i)\mathbf{Q}_{a_i}\mathbf{d}(t_{i+1})\mathbf{Q}_{a_{i+1}}\ldots\mathbf{d}(t_m)\mathbf{1}}{\pi_{s_\star}\mathbf{d}(t_1)\mathbf{Q}_{a_1}\mathbf{d}(t_2)\mathbf{Q}_{a_2}\ldots\mathbf{d}(t_m)\mathbf{1}}$$

### C.3.3 Espérance conditionnelle du nombre de transitions tirées entre deux états sachant un temps d'absorption

On considère d'abord le cas où la transition arrive dans un état $s' \in S_{ph}$.

$$\begin{aligned}
\mathbb{E}[N_{s,s'}|Y=y] &= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \mathbb{P}(X_{i\Delta t} = s, X_{(i+1)\Delta t} = s' | Y = y)dt \\
&= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \frac{\mathbb{P}(X_{i\Delta t} = s, X_{(i+1)\Delta t} = s', Y = y)}{\mathbb{P}(Y=y)}dt
\end{aligned}$$

On utilise la formule de Bayes pour décomposer la probabilité suivante :

$$\mathbb{P}(X_{i\Delta t} = s, X_{(i+1)\Delta t} = s', Y = y)$$

$$=$$

$$\mathbb{P}(X_t = s)\mathbb{P}(X_{(i+1)\Delta t} = s' | X_{i\Delta t} = s')\mathbb{P}(Y = y | X_{(i+1)\Delta t} = s')$$

Et on remplace les probabilités par leurs expressions analytiques :

$$(\pi e^{i\Delta t \mathbf{T}} \pi'_s)(\mathbf{T}(s,s')\Delta t + o(\Delta t))(\pi_{s'} e^{(y - i\Delta t - \Delta t)\mathbf{T}}\mathbf{t})$$

Les troisième et quatrième égalités s'obtiennent en utilisant la propriété de Markov du processus. Si on fait tendre $\Delta t$ vers 0 on obtient alors :

$$\begin{aligned}
\mathbb{E}[N_{s,s'}|Y=y] &= \frac{\int_0^y \pi e^{t\mathbf{T}}\pi'_s \mathbf{T}(s,s')\pi_{s'} e^{(y-t)\mathbf{T}}\mathbf{t}dt}{\mathbb{P}(Y \in dy)} \\
&= \frac{\int_0^y \pi e^{t\mathbf{T}}\pi'_s \mathbf{T}(s,s')\pi_{s'} e^{(y-t)\mathbf{T}}\mathbf{t}dt}{\mathbb{P}(Y \in dy)} \\
&= \frac{\mathbf{T}(s,s')\pi_{s'}\int_0^y \pi e^{t\mathbf{T}}\pi'_s e^{(y-t)\mathbf{T}}\mathbf{t}dt}{\mathbb{P}(Y \in dy)} \\
&= \frac{\mathbf{T}(s,s')c^s_{s'}(y)}{\pi\mathbf{b}(y)}
\end{aligned}$$

Dans le cas où $(s, s') \in S_{Ph} \times \{nil\}$, l'espérance conditionnelle du nombre de fois où la transition $(s, s')$ a été tirée revient à considérer la probabilité conditionnelle que la dernière transition vers l'état absorbant au temps $y$ part de l'état $s$ :

$$\begin{aligned}
\mathbb{E}[N_{s,nil} \mid Y = y] &= \mathbb{P}(X_y = s | Y = y) \\
&= \frac{\mathbb{P}(X_y = s)\mathbb{P}(Y \in dy \mid X_y = s)}{\mathbb{P}(Y \in dy)} \\
&= \frac{(\pi e^{y\mathbf{T}} \pi'_s)(\pi_s \mathbf{t})}{\pi e^{y\mathbf{T}} \mathbf{t}} \\
&= \frac{\mathbf{t}(s) a_s(y)}{\pi \mathbf{b}(y)}
\end{aligned}$$

On obtient donc finalement :

$$\mathbb{E}[\mathbf{N}_{s,s'} | \mathbf{Y} = \mathbf{y}] = \sum_{i=1}^{n} \frac{\mathbf{Q}(s, s')}{\pi_\star \mathbf{b}(y_i)} \times \begin{cases} c^s_{s'}(y_i) & \text{si } s' \neq nil \\ a_s(y_i) & \text{si } s' = nil \end{cases}$$

avec $\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \mathbf{t} \\ 0 \ldots 0 & 0 \end{pmatrix}$.

### C.3.4 Espérance conditionnelle du nombre de transitions tirées de taux $v$ connaissant un temps d'absorption

Soient $N_\zeta$, $|T|$ variables aléatoires définissant le nombre total de fois que le processus $\mathfrak{X}$ tire la transition $\zeta \in T$. On suppose dans un premier temps que $\zeta\bullet \neq nil$.

$$\begin{aligned}
\mathbb{E}[N_\zeta | Y = y] &= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \mathbb{P}(C^\zeta_{i\Delta t + \Delta t} \neq C^\zeta_{i\Delta t} | Y = y) \\
&= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \frac{\mathbb{P}(C^\zeta_{i\Delta t + \Delta t} \neq C^\zeta_{i\Delta t}, Y = y)}{\mathbb{P}(Y = y)} \\
&= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \frac{\mathbb{P}(X_{i\Delta t + \Delta t} = \zeta\bullet, X_{i\Delta t} = \bullet\zeta, C^\zeta_{i\Delta t + \Delta t} \neq C^\zeta_{i\Delta t}, Y = y)}{\mathbb{P}(Y = y)}
\end{aligned}$$

On utilise la propriété de Markov de $\mathfrak{X}$ :

$$\begin{aligned}
&= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \frac{\begin{array}{l}\mathbb{P}(X_{i\Delta t} = \bullet\zeta) \\ \times \mathbb{P}(X_{i\Delta t + \Delta t} = \zeta\bullet, C^\zeta_{i\Delta t + \Delta t} \neq C^\zeta_{i\Delta t} | X_{i\Delta t} = \bullet\zeta) \\ \times \mathbb{P}(Y = y \mid X_{i\Delta t + \Delta t} = \zeta\bullet)\end{array}}{\mathbb{P}(Y = y)} \\
&= \lim_{\Delta t \to 0} \sum_{i=0}^{[y/\Delta t]-1} \frac{\sum_{\zeta \in \mathbb{T}(v)} (\pi_{s_\star} e^{i\Delta t \mathbf{T}} \pi'_{\bullet\zeta})(ev(\zeta)\Delta t + o(\Delta t))(\pi_{\zeta\bullet} e^{(y - i\Delta t - \Delta t)\mathbf{T}} \mathbf{t})}{\mathbb{P}(Y = y)}
\end{aligned}$$

On considère la limite quand $\Delta t$ tend vers 0 :

$$\begin{aligned}\mathbb{E}[N_\zeta|Y=y] &= \frac{\int_0^y \pi_{s_\star} e^{t\mathbf{T}} \pi'_{\bullet\zeta} ev(rate(\zeta)) \pi_{\zeta\bullet} e^{(y-t)\mathbf{T}} \mathbf{t} dt}{\mathbb{P}(Y=y)} \\ &= \frac{ev(\zeta)\pi_{\zeta\bullet} \int_0^y \pi_{s_\star} e^{t\mathbf{T}} \pi'_{\bullet\zeta} e^{(y-t)\mathbf{T}} \mathbf{t} dt}{\mathbb{P}(Y=y)} \\ &= \frac{c_{\zeta\bullet}^{\bullet\zeta}(y)}{\pi_{s_\star}\mathbf{b}(y)}\end{aligned}$$

Dans le cas où $\zeta\bullet = nil$, l'espérance conditionnelle du nombre de fois où $\zeta$ est tirée revient à considérer la probabilité conditionnelle que $\zeta$ soit la dernière transition tiré au temps $y$ :

$$\mathbb{E}[N_\zeta|Y=y] = \mathbb{P}(C_y^\zeta \neq C_{y+\epsilon}^\zeta | Y=y)$$

avec $\epsilon > 0$. L'équation suivante est obtenue en utilisant la propriété de Markov :

$$\begin{aligned} &= \frac{\mathbb{P}(X_y = \bullet\zeta)\mathbb{P}(Y=y \mid X_y = \bullet\zeta, C_y^\zeta \neq C_{y+\epsilon}^\zeta)}{\mathbb{P}(Y=y)} \\ &= \frac{(\pi_\star e^{y\mathbf{T}} \pi'_s) ev(\zeta)}{\pi e^{y\mathbf{T}}\mathbf{t}} \\ &= \frac{ev(\zeta)a_s(y)}{\pi\mathbf{b}(y)}\end{aligned}$$

Il s'ensuit donc :

$$\mathbb{E}[N_v|Y=y] = \frac{\sum_{\zeta\in\mathbb{T}(v)} ev(v)}{\pi_{s_\star}\mathbf{b}(y)} \times \begin{cases} c_{\zeta\bullet}^{\bullet\zeta}(y) & \text{if } \zeta\bullet \neq nil \\ a_{\zeta\bullet}(y) & \text{if } \zeta\bullet = nil \end{cases}$$

### C.3.5 Espérance conditionnelle du nombre de transitions tirées de taux $v$ connaissant une exécution partiellement observables

$$\mathbb{E}[N_v|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\lim_{\epsilon\to 0} \sum_{i=0}^{\infty} \mathbb{P}(C_{i\epsilon}^v \neq C_{(i+1)\epsilon}^v | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$$

On partitionne $\mathbb{P}(C_{i\epsilon}^v \neq C_{(i+1)\epsilon}^v|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$ suivant les variables aléatoires $X_{i\epsilon}$ et $X_{(i+1)\epsilon}$ :

$$\mathbb{P}(C_{i\epsilon}^v \neq C_{(i+1)\epsilon}^v|\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$$

$$=$$

$$\sum_{s\in S}\sum_{s'\in S}\mathbb{P}(X_{i\epsilon}=s, C^v_{i\epsilon}\neq C^v_{(i+1)\epsilon}, X_{(i+1)\epsilon}=s'|\mathfrak{Y}=init\xrightarrow{t_1}a_1\ldots\xrightarrow{t_m}stop)$$

On va développer la probabilité ci-dessus de manière à pouvoir en donner une expression analytique. On choisi un $i$ en particulier, et on décompose cette probabilité suivante deux cas, ou bien 1) la transition[1] de taux $v$ entre l'instant $i\epsilon$ et l'instant $(i+1)\epsilon$ n'est pas observable (i.e. d'action $\tau$) ou bien 2) elle est observable (i.e. d'action appartenant à $Act$) :

1. La transition de taux $v$ prise entre l'instant $i\epsilon$ et l'instant $(i+1)\epsilon$ n'est pas observable (i.e. d'action $\tau$) :

   $$\exists j,\ \theta_{j-1}\leq i\epsilon \text{ et } (i+1)\epsilon<\theta_j$$

   On peut exprimer $\mathfrak{Y}=init\xrightarrow{t_1}a_1\ldots\xrightarrow{t_m}stop$ par la notation suivante (voir l'*annexe* C.3.2) :

   $$\mathfrak{Y}=0\xrightarrow{a_1}\theta_1\ldots\theta_{j-1}\xrightarrow{\tau*}i\epsilon\xrightarrow{\tau*}(i+1)\epsilon\xrightarrow{a_j}\theta_j\ldots\theta_{m-1}\xrightarrow{stop}\theta_m$$

   Ce qui est équivalent à la notation :

   $$\begin{array}{l}\mathfrak{Y}=0\xrightarrow{a_1}\theta_1\ldots\theta_{j-1}\xrightarrow{\tau*}i\epsilon,\\ C^{Act\cup\{stop\}}_{i\epsilon}=C^{Act\cup\{stop\}}_{(i+1)\epsilon},\\ \mathfrak{Y}=(i+1)\epsilon\xrightarrow{a_j}\theta_j\ldots\theta_{m-1}\xrightarrow{stop}\theta_m\end{array}$$

   La formule de Bayes ainsi que la propriété de Markov nous permettent ensuite d'obtenir l'égalité suivante (le raisonnement est analogue à ceux de l'*annexe* C.3.2) :

   $$\mathbb{P}(X_{i\epsilon}=s, C^v_{i\epsilon}\neq C^v_{(i+1)\epsilon}, X_{(i+1)\epsilon}=s'|\mathfrak{Y}=init\xrightarrow{t_1}a_1\ldots\xrightarrow{t_m}stop)$$

   $$=$$

   $$\frac{\begin{array}{l}\mathbb{P}(\mathfrak{Y}=0\xrightarrow{a_1}\theta_1\ldots\theta_{j-1}\xrightarrow{\tau*}i\epsilon, X_{i\epsilon}=s)\\ \times\mathbb{P}(C^v_{i\epsilon}\neq C^v_{(i+1)\epsilon}, C^{Act\cup\{stop\}}_{i\epsilon}=C^{Act\cup\{stop\}}_{(i+1)\epsilon}, X_{(i+1)\epsilon}=s'|X_{i\epsilon}=s)\\ \times\mathbb{P}(\mathfrak{Y}=(i+1)\epsilon\xrightarrow{a_j}\theta_j\ldots\theta_{m-1}\xrightarrow{stop}\theta_m|X_{(i+1)\epsilon}=s')\end{array}}{\mathbb{P}(\mathfrak{Y}=init\xrightarrow{t_1}a_1\ldots\xrightarrow{t_m}stop)}$$

   Avant de donner son expression analytique on passe au deuxième cas.

2. La transition de taux $v$ prise entre l'instant $i\epsilon$ et l'instant $(i+1)\epsilon$ est observable (i.e. d'action appartenant à $Act$) :

   $$\exists j,\ i\epsilon<\theta_j<(i+1)\epsilon$$

[1] En fait plusieurs transitions peuvent être prises entre deux instants, mais comme on considère la limite quand la différence de ces deux instants tend vers 0, il est acceptable de ne parler que d'une seule transition.

*et*

*l'observation $a_j$ correspond à une transition de taux $v$*

Ceci est exprimé par l'événement limite suivant :

$$\begin{array}{l}
\lim_{h_1\to 0} C_0^{Act\cup\{stop\}} = C_{\theta_1-h_1}^{Act\cup\{stop\}}, C_{\theta_1-h_1}^{a_1} \neq C_{\theta_1}^{a_1}, \\
\vdots \\
\lim_{h_j\to 0} C_{j-1}^{Act\cup\{stop\}} = C_{\theta_j-h_j}^{Act\cup\{stop\}}, C_{\theta_j-h_j}^{a_j} \neq C_{\theta_j}^{a_j}, C_{\theta_j-h_j}^{V\setminus\{v\}} \neq C_{\theta_j}^{V\setminus\{v\}}, \\
\vdots \\
\lim_{h_m\to 0} C_{\theta_{m-1}}^{Act\cup\{stop\}} = C_{\theta_m-h_m}^{Act\cup\{stop\}}, C_{\theta_m-h_m}^{stop} \neq C_{\theta_m}^{stop}
\end{array}$$

Afin de simplifier les notations nous noterons cet évènement :

$$\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \theta_{j-1} \xrightarrow{a_j,v} \theta_j \ldots \xrightarrow{stop} \theta_m$$

Lequel sera équivalent à la notation :

$$\begin{array}{l}
\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \theta_{j-1} \xrightarrow{\tau*} i\epsilon, \\
\mathfrak{Y} = i\epsilon \xrightarrow{a_j,v} \theta_j \xrightarrow{\tau*} (i+1)\epsilon, \\
\mathfrak{Y} = (i+1)\epsilon \xrightarrow{a_{j+1}} \theta_{j+1} \ldots \theta_{m-1} \xrightarrow{stop} \theta_m
\end{array}$$

De la même manière on utilise la formule de Bayes et la propriété de Markov pour obtenir l'égalité suivante :

$$\mathbb{P}(X_{i\epsilon} = s, C_{i\epsilon}^v \neq C_{(i+1)\epsilon}^v, X_{(i+1)\epsilon} = s' | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)$$

$$=$$

$$\frac{\begin{array}{l}
\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \theta_{j-1} \xrightarrow{\tau*} i\epsilon, X_{i\epsilon} = s) \\
\times \mathbb{P}(C_{i\epsilon}^v \neq C_{(i+1)\epsilon}^v, \mathfrak{Y} = i\epsilon \xrightarrow{a_j,v} \theta_j \xrightarrow{\tau*} (i+1)\epsilon, X_{(i+1)\epsilon} = s' | X_{i\epsilon} = s) \\
\times \mathbb{P}(\mathfrak{Y} = (i+1)\epsilon \xrightarrow{a_j} \theta_j \ldots \theta_{m-1} \xrightarrow{stop} \theta_m | X_{(i+1)\epsilon} = s')
\end{array}}{\mathbb{P}(\mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop)}$$

Les expressions analytiques de $\mathbb{P}(\mathfrak{Y} = t \xrightarrow{a_i} \ldots \xrightarrow{stop} \theta_m | X_t = s)$ et $\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \theta_1 \ldots \xrightarrow{stop} \theta_m)$ ont déjà été définies dans l'annexe C.3.2. De façon analogue on peut montrer que :

$$\begin{array}{rcl}
\mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{\tau^*} i\epsilon, X_{i\epsilon} = s) & = & \sum_{s''\in S} \mathbb{P}(\mathfrak{Y} = 0 \xrightarrow{a_1} \ldots \xrightarrow{a_{j-1}} \theta_{j-1}, X_{\theta_{j-1}} = s'') \\
& & \times \mathbb{P}(\mathfrak{Y} = \theta_{j-1} \xrightarrow{\tau^*} i\epsilon, X_{i\epsilon} = s | X_{\theta_{j-1}} = s'')
\end{array}$$

$$=$$

$$\pi_{s_\star} e^{t_1 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_1} e^{t_2 \mathbf{Q}_\tau^{stop}} \mathbf{Q}_{a_2} \ldots \mathbf{Q}_{a_{j-1}} e^{(i\epsilon-\theta_{j-1})\mathbf{Q}_\tau^{stop}} \pi'_s$$

Enfin il nous reste enfin à exprimer sous forme analytique les probabilités suivantes :

$$\mathbb{P}(C^v_{i\epsilon} \neq C^v_{(i+1)\epsilon}, C^{Act\cup\{stop\}}_{i\epsilon} = C^{Act\cup\{stop\}}_{(i+1)\epsilon}, X_{(i+1)\epsilon} = s' | X_{i\epsilon} = s)$$

et de :

$$\mathbb{P}(C^v_{i\epsilon} \neq C^v_{(i+1)\epsilon}, \mathfrak{Y} = i\epsilon \xrightarrow{a_j,v} \theta_j \xrightarrow{\tau*} (i+1)\epsilon, X_{(i+1)\epsilon} = s' | X_{i\epsilon} = s)$$

L'égalité qui suit vient simplement des définitions de $C^v_t$ et de $C^{Act\cup\{stop\}}_t$ et de la propriété de Markov de $\mathfrak{X}$ :

$$\mathbb{P}(C^v_{i\epsilon} \neq C^v_{(i+1)\epsilon}, C^{Act\cup\{stop\}}_{i\epsilon} = C^{Act\cup\{stop\}}_{(i+1)\epsilon}, X_{(i+1)\epsilon} = s' | X_{i\epsilon} = s)$$

$$=$$

$$\sum_{\substack{\zeta\in\mathbb{T}(v),\\ \zeta\bullet=s',\\ act(\zeta)=\tau}} ev(\zeta)\epsilon + o(\epsilon)$$

que l'on peut simplifier en utilisant la définition de $\mathbf{Q}_{v,\tau}$ :

$$= \mathbf{Q}_{v,\tau}(s, s')\epsilon + o(\epsilon)$$

Précisons que l'expression ci-dessus désigne bien une probabilité tandis que celle ci-dessous va désigner une densité de probabilité. Maintenant on traite la deuxième égalité :

$$\mathbb{P}(C^v_{i\epsilon} \neq C^v_{(i+1)\epsilon}, \mathfrak{Y} = i\epsilon \xrightarrow{a_j,v} \theta_j \xrightarrow{\tau*} (i+1)\epsilon, X_{(i+1)\epsilon} = s' | X_{i\epsilon} = s)$$

$$=$$

$$\pi_s e^{(\theta_j - i\epsilon)\mathbf{Q}^{stop}_\tau} \mathbf{Q}_{a_j,v} e^{((i+1)\epsilon - \theta_j)\mathbf{Q}^{stop}_\tau} \pi'_{s'}$$

On fait tendre $\epsilon$ vers 0 ce qui permet d'écrire l'expression analytique complète de $N_v$ :

$$\mathbb{E}[N_v | \mathfrak{Y} = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\begin{array}{l}\pi_{s_\star} e^{t_1\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{a_1} e^{t_2\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{a_2}\ldots \\ ((\int_{\theta_{i-1}}^{\theta_i} e^{(t-\theta_{i-1})\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{v,\tau}e^{(\theta_i - t)\mathbf{Q}^{stop}_\tau} dt\ \mathbf{Q}_{a_i}) + e^{t_i\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{v,a_i}) \\ e^{t_{i+1}\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{a_{i+1}}\ldots e^{t_{m-1}\mathbf{Q}^{stop}_\tau}\mathbf{q}_{stop}\end{array}}{\pi_{s_\star} e^{t_1\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{a_1} e^{t_2\mathbf{Q}^{stop}_\tau}\mathbf{Q}_{a_2}\ldots e^{t_{m-1}\mathbf{Q}^{stop}_\tau}\mathbf{q}_{stop}}$$

De la même manière que pour l'espérance conditionnelle de $\mathbf{Z}_v$, l'hypothèse d'indépendance entre l'observateur et le processus observé permet de remplacer le vecteur $\mathbf{q}_{stop}$ par le vecteur $\mathbf{1}$ et la matrice $\mathbf{Q}^{stop}_\tau$ par $\mathbf{Q}_\tau$ :

$$\mathbb{E}[N_v | Y = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\begin{array}{l}\pi_{s_\star} e^{t_1 \mathbf{Q}_\tau} \mathbf{Q}_{a_1} e^{t_2 \mathbf{Q}_\tau} \mathbf{Q}_{a_2} \ldots \\ ((\int_{\theta_{i-1}}^{\theta_i} e^{(t-\theta_{i-1})\mathbf{Q}_\tau} \mathbf{Q}_{v,\tau} e^{(\theta_i - t)\mathbf{Q}_\tau} dt \ \mathbf{Q}_{a_i}) + e^{t_i \mathbf{Q}_\tau} \mathbf{Q}_{v,a_i}) \\ e^{t_{i+1}\mathbf{Q}_\tau} \mathbf{Q}_{a_{i+1}} \ldots e^{t_{m-1}\mathbf{Q}_\tau} \mathbf{q}_{stop}\end{array}}{\pi_{s_\star} e^{t_1 \mathbf{Q}_\tau} \mathbf{Q}_{a_1} e^{t_2 \mathbf{Q}_\tau} \mathbf{Q}_{a_2} \ldots e^{t_{m-1}\mathbf{Q}_\tau} \mathbf{q}_{stop}}$$

Enfin on remplace les termes $e^{t_i \mathbf{Q}_\tau}$ par $\mathbf{d}(t_i)$ et les termes $\int_{\theta_{i-1}}^{\theta_i} e^{(t-\theta_{i-1})\mathbf{Q}_\tau} \mathbf{Q}_{v,\tau} e^{(\theta_i - t)\mathbf{Q}_\tau} dt$ par $\mathbf{f}(v, t_i)$ :

$$\mathbb{E}[N_v \,|Y = init \xrightarrow{t_1} a_1 \ldots \xrightarrow{t_m} stop]$$

$$=$$

$$\sum_{i=1}^{m} \frac{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1} \mathbf{d}(t_2)\mathbf{Q}_{a_2} \ldots (\mathbf{f}(v,t_i)\mathbf{Q}_{a_i} + \mathbf{d}(t_i)\mathbf{Q}_{v,a_i})\mathbf{d}(t_{i+1})\mathbf{Q}_{a_{i+1}} \ldots \mathbf{d}(t_{m-1})\mathbf{1}}{\pi_{s_\star} \mathbf{d}(t_1)\mathbf{Q}_{a_1} \mathbf{d}(t_2)\mathbf{Q}_{a_2} \ldots \mathbf{d}(t_{m-1})\mathbf{1}}$$

## C.4 Étape de Maximisation

### C.4.1 La Maximisation de $L(\mathbf{x})$

Les *équations* 4.2 à la *page* 60 et 4.4 à la *page* 61 impliquent directement la définition de la vraisemblance de $\mathbf{x}$ suivante :

$$\begin{aligned} L(\mathbf{x}) = \ & \prod_{i=1}^{n} \ [\prod_{s \in S_{ph}} e^{\mathbf{T}(s,s)\times z_s^i}] \\ & \times \ [\prod_{\substack{s,s' \in S_{ph}, \\ s' \neq s}} \mathbf{T}(s,s')^{n_{s,s'}^i}] \\ & \times \ [\prod_{s \in S_{ph}} \mathbf{t}(s)^{n_{s,nil}^i}] \end{aligned}$$

où $z_s^i$ est le temps de séjour dans l'état $s \in S_{ph}$ de la $i^{ème}$ exécution et $n_{s,s'}^i$ le nombre de fois que la transition entre $s \in S_{ph}$ et $s' \in S$ est tirée par la $i^{ème}$ exécution. Le produit d'exponentielles se transforme en l'exponentielle de la somme des arguments des exponentielles, on déduit donc :

$$\begin{aligned} L(\mathbf{x}) \ = \ & [\prod_{s \in S_{ph}} e^{\mathbf{T}(s,s)\times \mathbf{z}_s}] \\ & \times \ [\prod_{\substack{s,s' \in S_{ph}, \\ s' \neq s}} \mathbf{T}(s,s')^{\mathbf{n}_{s,s'}}] \\ & \times \ [\prod_{s \in S_{ph}} \mathbf{t}(s)^{\mathbf{n}_{s,nil}}] \end{aligned}$$

avec $\mathbf{z}_s$ le temps de séjour total de toutes les exécutions de $\mathbf{x}$ dans l'état $s \in S_{ph}$ :

$$\forall s \in S_{ph} \quad \mathbf{z}_s = \sum_{i=1}^{n} z_s$$

et $\mathbf{n}_{s,s'}$ le nombre total de fois que la transition partant de $s \in S_{ph}$ et arrivant dans $s' \in S$ est tirée par l'ensemble des exécutions de $\mathbf{x}$ :

$$\forall s \in S_{ph} \forall s' \in S \quad \mathbf{n}_{s,s'} = \sum_{i=1}^{n} n_{s,s'}$$

On va ensuite considérer la dérivée partielle de $log(L(x))$ par rapport à un coefficient de $\mathbf{T}$ choisi arbitrairement, noté $\mathbf{T}(s,s')$, et chercher en quels points cette dérivée partielle s'annule. On commence par simplifier $L(\mathbf{x})$ en mettant de côté tous les termes ne contenant pas $\mathbf{T}(s,s')$ :

$$L(\mathbf{x}) = C \times e^{\mathbf{T}(s,s)\times \mathbf{z}_s} \times \mathbf{T}(s,s')^{n_{s,s'}}$$

où $C$ est une quantité qui ne dépend pas de $\mathbf{T}(s,s')$. Aussi de la même façon (et considérant que $\mathbf{T}(s,s) = -\mathbf{t}(s) - \sum_{\substack{s' \in S \\ s' \neq s}} \mathbf{T}(s,s')$) on met de côté tous les termes différents de $\mathbf{T}(s,s')$ dans $\mathbf{T}(s,s)$ et on obtient :

$$L(\mathbf{x}) = C' \times e^{-\mathbf{T}(s,s')\times \mathbf{z}_s} \times \mathbf{T}(s,s')^{n_{s,s'}}$$

On étudie maintenant le logarithme de $L(\mathbf{x})$ (car la fonction logarithme est monotone croissante) :

$$log(L(\mathbf{x})) = C'' - \mathbf{T}(s,s') \times \mathbf{z}_s + \mathbf{n}_{s,s'} \times log(\mathbf{T}(s,s'))$$

$C'$ et $C''$ sont des quantités qui ne dépendent pas de $\mathbf{T}(s,s')$. Puis on résout la dérivée partielle de $log(L(\mathbf{x}))$ suivant $\mathbf{T}(s,s')$ :

$$\frac{\partial log(L(\mathbf{x}))}{\partial \mathbf{T}(s,s')} = -\mathbf{z}_s + \frac{\mathbf{n}_{s,s'}}{\mathbf{T}(s,s')}$$

Il est maintenant possible de connaître le tableau de variations de $log(L(\mathbf{x}))$ en fonction de $\mathbf{T}(s,s')$ :

| $\mathbf{T}(s,s')$ | $< \frac{\mathbf{n}_{s,s'}}{\mathbf{z}_s}$ | $= \frac{\mathbf{n}_{s,s'}}{\mathbf{z}_s}$ | $> \frac{\mathbf{n}_{s,s'}}{\mathbf{z}_s}$ |
|---|---|---|---|
| $\frac{\partial log(L(\mathbf{x}))}{\partial \mathbf{T}(s,s')}$ | $+$ | $0$ | $-$ |
| $log(L(\mathbf{x}))$ | $\nearrow$ | | $\searrow$ |

Le même raisonnement tient pour les autres coefficients de $\mathbf{T}$ et de $\mathbf{t}$.

### C.4.2 La Maximisation de $L(\mathfrak{x})$

On montre dans cette annexe quelles affectations doivent être faites pendant l'étape de maximisation de l'algorithme. L'*annexe* C.1.3 nous donne la fonction de vraisemblance (la densité de probabilité d'avoir $\mathfrak{X} = \mathfrak{x}$) suivante :

$$\begin{aligned} L(\mathfrak{x}) &= ev(\zeta_1) \times e^{-\mathbf{E}(s_1)t_1} \\ &\quad \times ev(\zeta_2) \times e^{-\mathbf{E}(s_2)t_2} \\ &\quad \vdots \\ &\quad \times ev(\zeta_m) \times e^{-\mathbf{E}(s_m)t_m} \end{aligned}$$

Si $n_\zeta$ est le nombre total de fois que l'exécution $\mathfrak{x}$ a tiré la transition $\zeta \in T$ et $z_s$ le temps total de séjours de $\mathfrak{x}$ dans l'état $s \in S_{ph}$ alors la densité de probabilité de $\mathfrak{X} = \mathfrak{x}$ peut s'écrire (on utilise le fait que l'exponentielle transforme la somme en produit et on pose la convention que $x^y = 1$ quand $y = 0$) :

$$L(\mathfrak{x}) = [\prod_{s \in S_{ph}} e^{-\mathbf{E}(s) \times z_s}] \times [\prod_{\zeta \in T} ev(\zeta)^{n_\zeta}]$$

On considérer ensuite la dérivé de $log(L(\mathfrak{x}))$ suivant $v \in V$, notée $\frac{\partial log(L(\mathfrak{x}))}{\partial v}$, pour trouver son maximum. Avant de chercher $\frac{\partial log(L(\mathfrak{x}))}{\partial v}$ on va simplifier $L(\mathfrak{x})$. Si $n_v$ est le nombre total de fois où $\mathfrak{x}$ a tiré une transition de taux $v$ alors $n_v = \sum_{\zeta \in \mathbb{T}(v)} n_\zeta$ et nous pouvons écrire $L(\mathfrak{x})$ de la façon suivante :

$$L(\mathfrak{x}) = C \times [\prod_{s \in S_{ph}} e^{-\mathbf{E}(s) \times z_s}] \times ev(v)^{n_v}$$

ou de la façon suivante (en utilisant le fait que l'exponentielle transforme le produit en somme) :

$$L(\mathfrak{x}) = C \times e^{-\sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s} \times ev(v)^{n_v}$$

où $C$ est un réel qui ne dépend pas de $v$. On démontre ensuite le *lemme* suivant :

**Lemme C.1** *$\sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s$ peut s'écrire sous la forme $D + ev(v) \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta}$ où $D$ ne dépend pas de $v$.*

**Preuve**

Par induction sur la taille de $\mathbb{T}(v)$ :
– Si $\mathbb{T}(v)$ est vide alors $\sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s$ ne dépend pas de $v$, donc

$$D = \sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s \quad \text{et} \quad v \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta} = 0$$

– Soit $G'$ le graphe de dérivation $G$ avec une transition $\zeta$ de taux $v$ en moins. Aussi $\mathbb{T}'(v)$ correspond à $\mathbb{T}(v)$ moins la transition $\zeta$. On définit $\mathbf{E}' : S \mapsto \mathbb{R}_+$ le vecteur des taux de probabilité de sortie de chaque état de $S_{ph}$ de $G'$. D'après la définition de $\mathbf{E}(s)$ on peut déduire que :

$$\sum_{s \in S_{ph}} \mathbf{E}(s) - \sum_{s \in S_{ph}} \mathbf{E}'(s) = ev(\zeta) = ev(v)$$

ainsi que :

$$\sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s = \sum_{s \in S_{ph}} \mathbf{E}'(s) \times z_s + ev(v) \times z_{\bullet\zeta}$$

On suppose (hypothèse d'induction) que $\sum_{s \in S_{ph}} \mathbf{E}'(s) \times z_s$ peut s'écrire sous la forme $D + ev(v) \sum_{\zeta \in \mathbb{T}'(v)} z_{\bullet\zeta}$. On peut donc conclure que :

$$\sum_{s \in S_{ph}} \mathbf{E}(s) \times z_s = D + ev(v) \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta}$$

□

Le lemme précédant nous permet d'exprimer $L(\mathfrak{x})$ de la façon suivante :

$$L(\mathfrak{x}) = C \times e^{-D - ev(v) \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta}} \times ev(v)^{n_v}$$

On considère le logarithme de $L(\mathfrak{x})$ (car la fonction logarithme est monotone croissante) :

$$log(L(\mathfrak{x})) = C' - ev(v) \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta} + n_v log(ev(v))$$

où $C'$ est une grandeur qui ne dépend pas de $v$. On résout maintenant la dérivée partielle du logarithme de $L(\mathfrak{x})$ :

$$\frac{\partial log(L(\mathfrak{x}))}{\partial v} = - \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta} + \frac{n_v}{ev(v)}$$

On pose $n_v = \sum_{\zeta \in \mathbb{T}(v)} z_{\bullet\zeta}$. Il est maintenant facile de connaître le tableau de variations de $log(L(\mathfrak{x}))$ en fonction de $ev(v)$ :

| $ev(v)$ | $< \frac{n_v}{z_v}$ | $= \frac{n_v}{z_v}$ | $> \frac{n_v}{z_v}$ |
|---|---|---|---|
| $\frac{\partial log(L(\mathfrak{x}))}{\partial ev(v)}$ | $+$ | $0$ | $-$ |
| $log(L(\mathfrak{x}))$ | ↗ | | ↘ |

# Appendix D

# EMPEPA

## D.1 Descriptif de l'utilisation du logiciel EMPEPA et de ces options en ligne de commande

Les fonctions du logiciel EMPEPA sont invoquées par la ligne de commande :

`empepa [options] fichier1 [fichier2]`

Le logiciel EMPEPA demande toujours au moins un fichier en entrée, `fichier1`, contenant un modèle PEPA, dont la syntaxe est décrite à l'*annexe* D.2.1. Suivant les options sélectionnées il attend en entrée un second fichier, `fichier2`, décrivant les données d'observations, lesquelles suivent la syntaxe décrite dans l'*annexe* D.2.2. Les options sont les suivantes :

`-h` ou `-help`

permet d'afficher quelques lignes d'explications sur l'utilisation de EMPEPA et sur ces options.

`-aALGO`

où `ALGO` désigne le schémas de résolution des systèmes différentielles ordinaires de l'algorithme. `ALGO` peut prendre les mots clef suivants :

1. `rk2` pour Runge-Kutta (2,3),
2. `rk4` pour Runge-Kutta d'ordre 4,
3. `rkf45` pour Runge-Kutta-Fehlberg (4,5),
4. `rkck` pour Runge-Kutta Cash-Karp (4,5),
5. `rk8pd` pour Runge-Kutta Prince-Dormand (8,9),
6. `rk2imp` pour Runge-Kutta à points Gaussien d'ordre 2 implicite,
7. `rk4imp` pour Runge-Kutta à points Gaussien d'ordre 4 implicite,
8. `gear1` pour M=1 implicit Gear method,

9. `gear2` pour M=2 implicit Gear method,
10. `simul` pour la méthode d'approximation par simulation.

Par défaut le schémas sélectionné est `rk4`.

`-rREEL_POSITIF`

où `REEL_POSITIF` est un réel positif désignant la précision de calcul de la méthode de résolution choisie. Par défaut cette valeur est de 1e-12.

`-iITERATIONS`

où `ITERATIONS` désigne le nombre d'itérations de l'algorithme. Par défaut ce nombre est de 10.

`-s` ou `-simulate`

Cette option permet d'utiliser les fonctions de simulation de EMPEPA. Dans ce cas aucun fichier `fichier2` n'est attendu en ligne de commande. En outre, si cette option est ajouté alors EMPEPA attend l'option suivante :

`-oFICHIER_OBSERVATIONS`

où `FICHIER_OBSERVATIONS` est le nom de fichier dans lequel EMPEPA doit écrire les observations obtenues par simulation. Si l'option :

`-srSTOP_RATE`

est ajouté alors EMPEPA inscrit une liste d'exécutions partiellement observables dans `FICHIER_OBSERVATIONS` avec comme temps moyen pour chaque exécution partiellement observable de 1/`STOP_RATE`. Sinon EMPEPA inscrit des temps d'absorption (attention, dans ce cas le modèle PEPA d'entrée doit être absorbant). L'option suivante :

`-nsNBR_SIMUL`

définit le nombre d'observations à inscrire dans le fichier `FICHIER_OBSERVATIONS`, c'est-à-dire le nombre de temps d'absorption ou le nombre d'exécution partiellement observables. Dans le cas où EMPEPA est utilisé pour maximiser les observations et que la méthode de calcul des espérances choisi est `simul` alors `NBR_SIMUL` désigne le nombre minimum de simulations nécessaires pour faire l'approximation de ces espérances.

Ci-dessous, voici un exemple d'une commande qui exécute EMPEPA avec la méthode numérique Runge-Kutta Prince-Dormand (8,9) et avec la précision $10^{-10}$ pour maximiser la vraisemblance des observations contenu dans le fichier observation.data considérant le modèle PEPA contenu dans le fichier model.pepa :

```
empepa -ark8pd -r1e-10 observations.data model.pepa
```

L'exemple suivant permet d'obtenir 100 exécutions partiellement observables de taux d'arrêt 0.01 par simulation du fichier model.pepa. Les observations sont inscrites dans le fichier observations.data :

```
empepa -s -sn100 -sr0.01 -oobservations.data model.pepa
```

## D.2 Syntaxe des modèles PEPA et observations en entrées de EMPEPA

Les non-terminaux sont indiqués en italique tandis que les terminaux sont indiqué en police courrier. Les tokens sont indiqués en caractère italique majuscule. Le mot vide est symbolisé par $\epsilon$. On utilise deux tokens typés : *ID* représentant tout identificateur et *RP* représentant tout réel positif.

$$ID \quad ::= \quad [\texttt{A..z}]([\texttt{A..z}]|\_)^*$$

$$RP \quad ::= \quad [\texttt{0..9}]^+(.[\texttt{0..9}]^*|\epsilon)(\texttt{e}(\texttt{-}|\texttt{+})[\texttt{0..9}]^+|\epsilon)$$

### D.2.1 Syntaxe des modèles PEPA

On présente ici la syntaxe utilisée dans EMPEPA pour décrire une modèle PEPA. Cette syntaxe est similaire à la syntaxe utilisée dans le model checker PRISM avec en plus le mot chef `nil` pour désigner le processus inactif et une liste de déclarations de variables avec ou sans valeurs d'initialisations.

*PEPAModèle* ::= *DeclVariables DeclConstantes DeclTermes TermeRacine*

*DeclVariables* ::= `var` *VarList* `;`

*VarList* ::= *VarList* `,` *Variable* | $\epsilon$

*Variable* ::= *ID* | *ID* `=` *PR*

*DeclConstantes* ::= `const` *ConstList* `;`

*ConstList* ::= *ConstList* `,` *ID* `=` *RP* | $\epsilon$

*DeclTermes* ::= *DeclTermes* ♯ *ID* = *PEPATerme* ; | $\epsilon$

*PEPATerme* ::= *PEPATerme* / { *IDList* } | *PEPATerme* + *PEPATerme*
| ( *Action* , *Taux* ) . *PEPATerme* | *ID* | nil | ( *PEPATerme* )

*Action* ::= tau | *ID*

*Taux* ::= *RP* | *ID* | infty

*TermeRacine* ::= *PEPATerme* | *TermeRacine* < *IDList* > *TermeRacine*
| ( *TermeRacine* )

*IDList* ::= *IDList* , *ID* | $\epsilon$

Ci-dessous voici un exemple simple d'un modèle PEPA dans cette syntaxe :

```
var x=2.1e-3, y ;

const z=4.2 ;

♯P1=(a,x).(b,z).P1+(c,y).P1 ;
♯P2=((b,x).(a,z).P2)/{b} ;

P1<a>P2
```

### D.2.2 Syntaxe des observations

On présente ici la syntaxe des observations en entrée de EMPEPA. Il y a deux types possibles d'observations : *TempsAbsorption* représentant un ensemble de temps d'absorption et *ExecPartObsList* représentant un ensemble d'exécutions partiellement observables. Le symbole $\hookleftarrow$ signifie un ou plusieurs retours chariots.

*TempsAbsorption* ::= *TempsAbsorption Abs* | *Abs*

*Abs* ::= *RP* $\hookleftarrow$ | *RP RP* $\hookleftarrow$

*ExecPartObsList* ::= *ExecPartObsList ExecPartObs* | *ExecPartObs*

*ExecPartObs* ::= `init` *TempsObsList* `-` *RP* `>` `stop`

*TempsObsList* ::= *TempsObsList* `-` *RP* `>` *ID* | $\epsilon$

Ci-dessous voici un exemple d'un ensemble de quatre temps d'absorption. Le deuxième temps d'absorption 2.4e-1 a été observé deux fois et le temps d'absorption 5.9 trois fois.

1.2e-2
2.4e-1 2
4.3
5.9 3

L'exemple qui suit correspond à un ensemble de 4 exécutions partiellement observables :

```
init -0.3>a -1.2>b -2.3e-1>stop
init -1.2>stop
init -2.1>c -3.1e-6>a -4.34>b -4.8>b -0.2>stop
init -1.1>c -2.7>a -3.3e-3>b -2.3>stop
```

**Résumé :** L'objectif défini dans cette thèse est d'utiliser les récentes techniques de model checking probabiliste afin de vérifier des contraintes de performances sur des architectures de simulations distribuées. L'aspect probabiliste de l'approche permet à la fois de réduire la complexité du modèle et d'exprimer des contraintes de performances plus souples. Dans cette thèse nous avons dans un premier temps tenté d'utiliser le model checking probabiliste afin de formuler et de vérifier de telles contraintes de performances. Nous avons utilisé le model checker PRISM et confronté les résultats obtenus avec des données réelles provenant de simulations distribuées. Ceci a permis de révéler une difficulté fondamentale de cette démarche : il est difficile de choisir les coefficients du modèle pour que les résultats obtenus par le model checker coïncident avec la réalité. En clair, le modèle doit être réaliste.

Afin de surmonter cette difficulté nous avons d'abord utilisé des algorithmes d'approximation qui ont permis d'obtenir des modèles plus réalistes mais sans structure, les rendant difficilement compréhensibles et manipulables. Afin de remédier à ce problème nous avons adapté un de ces algorithmes d'approximation, permettant d'induire des distributions de type phase à partir d'un ensemble de temps d'absorption, pour lui permettre d'opérer sur des modèles décrit dans l'algèbre PEPA, un langage de modélisation basé sur une approche compositionnelle. L'algorithme a ensuite été étendu afin de fonctionner avec des exécutions partiellement observables au lieu de temps d'absorption, permettant ainsi de travailler sur des modèles PEPA sans état absorbant. Les expérimentations s'appuient sur le logiciel EMPEPA, distribué librement en Open-source, développé dans le cadre des travaux de thèse, et implantant en particulier ces deux algorithmes.

*Mots clefs :* Simulation distribuée, model checking probabiliste, chaînes de Markov, algorithme EM, algèbre stochastique PEPA.

**Abstract :** The goal defined in this thesis is to use the recent techniques of probabilistic model checking in order to check performances constraints over distributed simulation architecture. The probabilistic aspect of this approach allows both to reduce the complexity of the model and to express finer performance constraints. In this thesis we firstly attempt to use probabilistic model checking to formalize and verify such a constraints. The probabilistic model checker PRISM has been used and its results have been compared to real data coming from distributed simulations. This permitted to reveal a fundamental difficulty of this approach : it is difficult to choose the coefficients of the model so that the results obtained by the model checker coincide with reality. Clearly, the model has to be realistic.

To overcome this difficulty we have firstly used fitting algorithms allowing to obtain more realistic model but without structure, making them hard to understand and to handle. To remedy this problem we have adapt one fitting algorithm, permitting to induce phase type distributions according to absorption times, to make it operate on PEPA, a modeling language based on a compositional approach. Secondly, the algorithm has been extended to operate with partially observable executions instead of absorption times, allowing to work on non-absorbing PEPA model. Experimentation has been driven using the tool EMPEPA, an Open-source program, developed in the context of the thesis work, and implementing these two algorithms.

*Key words :* Distributed simulation, probabilistic model checking, Markov chains, EM algorithm, stochastic algebra PEPA.


10, avenue Édouard-Belin - BP 54032 - 31055 Toulouse cedex 4 - France
Tél : 33 (0)5 62 17 80 80 - Fax : 33 0(5) 62 17 83 30 - http://www.supaero.fr
Établissement public à caractère administratif sous tutelle du Ministère de la Défense