Chapitre 8

Vérification de systèmes probabilisés : méthodes et outils

8.1. Introduction

Pendant longtemps, la vérification fonctionnelle et l'évaluation de performance d'un système ou d'une application se présentaient comme deux étapes distinctes du processus de développement. Chacune disposait de ses modèles et de ses méthodes. Or depuis une quinzaine d'années, de nombreux travaux se situent à la croisée des deux thématiques et sont regroupées sous l'étiquette de vérification probabiliste ou de manière plus appropriée de vérification de systèmes probabilisés.

Ce nouvel axe de recherche répond au besoin des modélisateurs. Ceux-ci souhaitent par exemple évaluer la probabilité de satisfaction d'un propriété exprimée par une formule logique. Ils désirent aussi analyser un système présentant simultanément des aspects non déterministes et probabilistes. L'objectif de ce chapitre est de présenter au lecteur trois thèmes de recherche importants liés à cet axe : la définition de modèles stochastiques de haut niveau, la vérification de chaînes de Markov et celle des processus de décision markoviens.

Nous suivrons toujours l'organisation suivante : la présentation détaillée d'une approche, un panorama des approches alternatives et la description d'un outil d'analyse.

La première partie du chapitre est constituée de rappels indispensables sur les processus stochastiques et les chaînes de Markov. Puis la deuxième partie est consacrée

Chapitre rédigé par S. HADDAD et P. MOREAUX.

aux modèles de haut niveau. La partie suivante étudie la vérification des chaînes de Markov et la chapitre se conclut par une présentation de la vérification des processus de décision markoviens.

8.2. Evaluation de performance de modèles markoviens

8.2.1. *Un modèle stochastique de systèmes à événements discrets*

Nous supposons connues du lecteur les bases de la théorie des probabilités. Pour plus de détails, on pourra se reporter aux ouvrages suivants : [FOA 98, FOA 02] en français ou [FEL 68, FEL 71, TRI 82] en anglais.

Notations

– $\Pr(E)$ désigne la probabilité d'un événement E et $\Pr(A \mid B)$ la probabilité de A sachant B.

– L'adverbe $presque$, dans des expressions telles que *presque partout* ou *presque sûrement*, signifie pour un ensemble de probabilité 1.

– $\mathbb{R}$ (resp. $\mathbb{R}^+, \mathbb{R}^{+*}$) désigne les réels (resp. les réels non négatifs, strictement positifs). Si x est un réel alors $\lfloor x \rfloor$ désigne sa partie entière.

– Si $E \subseteq \mathbb{R}$ alors $Inf(E)$ (resp. $Sup(E)$) désigne la borne inférieure (resp. supérieure) de E.

Une exécution d'un SED se caractérise par une suite (*a priori* infinie) d'événements $\{e_1, e_2, \ldots\}$ séparés par des intervalles de temps. Seuls les événements peuvent changer l'état du système.

Formellement, le comportement stochastique d'un SED est déterminé par deux familles de variables aléatoires :

– $X_0, \ldots, X_n, \ldots$ à valeurs dans l'espace (discret) des états du système, noté S. Dans la suite sauf mention explicite, nous supposerons que cet espace est fini. X_0 représente l'état initial du système et X_n $(n > 0)$ l'état courant après le $n^{ième}$ événement. L'occurrence d'un événement ne modifie pas nécessairement l'état du système, par conséquent X_{n+1} peut être égal à X_n.

– $T_0, ..., T_n, ...$ à valeurs dans $\mathbb{R}^+$ où T_0 représente l'intervalle de temps avant le premier événement et T_n $(n > 0)$ représente l'intervalle de temps entre le $n^{ième}$ et le $(n+1)^{ième}$ événement. Notons que cet intervalle peut être nul (*e.g.* une suite d'instructions considérées comme instantanées au regard de transactions de base de données avec des entrées/sorties).

Lorsque la distribution initiale X_0 est concentrée en un état s, on dira que le processus démarre en s (*i.e.* $\Pr(X_0 = s) = 1$).

A priori, aucune restriction n'est imposée sur ces familles de variables aléatoires. Cependant, pour les catégories de processus que nous étudierons, un SED ne peut exécuter une infinité d'actions en un temps fini. Autrement dit :

$$\sum_{n=0}^{\infty} T_n = \infty \; presque \; sûrement \tag{8.1}$$

Cette propriété nous autorise à définir l'état du système à tout instant. Soit $N(\tau)$, la variable aléatoire définie par :

$$N(\tau) =_{def} Inf(\{n \mid \sum_{k=0}^{n} T_k > \tau\})$$

D'après l'équation [8.1], $N(\tau)$ est définie *presque partout*. Comme on peut le voir sur la figure 8.1, $N(\tau)$ présente des sauts d'amplitude supérieure à 1. L'état $Y(\tau)$ du système à l'instant τ, est alors simplement $X_{N(\tau)}$. Il est important de noter que $Y(\tau)$ n'est pas équivalent au processus stochastique, mais qu'il permet, dans la plupart des cas, de procéder aux analyses standard. Le schéma de la figure 8.1 présente une *réalisation* possible du processus et illustre l'interprétation de chacune des variables aléatoires introduites plus haut. Dans cet exemple, le processus est initialement dans l'état s_4 et y reste jusqu'à l'instant τ_0 où il passe dans l'état s_6. À l'instant $\tau_0 + \tau_1$, le système visite successivement en un temps nul, les états s_3 et s_{12} avant d'atteindre l'état s_7 où il séjourne un certain temps. L'observation $Y(\tau)$ en temps continu occulte les états évanescents s_3 et s_{12} du processus.

L'évaluation de performance d'un SED conduit à deux types d'analyse :

– L'étude du comportement transitoire , c'est à dire l'obtention de mesures en fonction du temps écoulé depuis l'état initial. Cette étude vise les phases d'initialisation des systèmes et les systèmes à états terminaux. Parmi les domaines d'application, on peut citer l'analyse de fiabilité et de sûreté de fonctionnement [LAP 95, MEY 80, TRI 92].

– L'étude du comportement stationnaire du système. Pour de nombreuses applications, ce qui intéresse le modélisateur est le comportement du système une fois la phase initiale passée, lorsqu'il se stabilise.

Ceci suppose bien entendu qu'un tel comportement stationnaire existe. Ce qui se résume, en notant $\boldsymbol{\pi}(\tau)$ la distribution de $Y(\tau)$, par :

$$\lim_{\tau \to \infty} \boldsymbol{\pi}(\tau) = \boldsymbol{\pi} \tag{8.2}$$

où $\boldsymbol{\pi}$ est aussi une distribution, appelée *distribution stationnaire*.

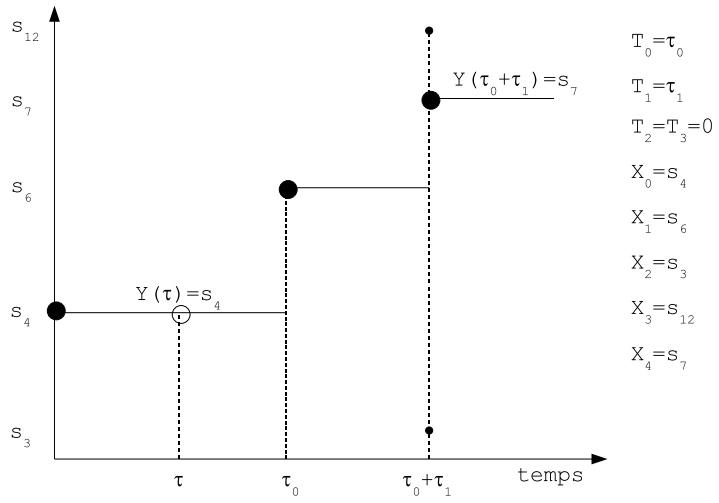

Figure 8.1: Une réalisation du processus stochastique

Les distributions transitoires ou stationnaires ne sont qu'un moyen de calculer des *indices de performance*. Par exemple, la probabilité stationnaire qu'un serveur soit opérationnel, la probabilité qu'à l'instant τ une connexion soit établie ou le nombre moyen de clients d'un service sont de tels indices .

Afin de raisonner de manière générique sur les SED on supposera donné dans la suite un ensemble de fonctions définies sur l'ensemble des états et à valeurs dans $\mathbb{R}$. Ainsi une fonction f peut être vue comme un indice de performance et étant donnée une distribution $\boldsymbol{\pi}$, la quantité $\sum_{s \in S} \boldsymbol{\pi}(s) \cdot f(s)$ représente la mesure de cet indice.

Lorsque l'indice est une fonction à valeurs dans $\{0, 1\}$, on peut l'assimiler à une proposition atomique satisfaite en un état si la fonction vaut 1. Dans la suite on notera $\mathcal{P}$, l'ensemble des propositions atomiques et $s \vDash \phi$, avec s un état et ϕ une proposition atomique, le fait que s vérifie ϕ. Dans ce cas étant donnée une distribution $\boldsymbol{\pi}$, la quantité $\sum_{s \vDash \phi} \boldsymbol{\pi}(s)$ représente la mesure de cet indice.

8.2.2. *Chaînes de Markov à temps discret*

Présentation

Une chaîne de Markov à temps discret (en anglais Discrete Time Markov Chain ou *DTMC*) possède les caractéristiques suivantes :

– L'intervalle de temps entre les instants T_n est une constante de valeur 1

– L'état suivant un état atteint ne dépend que de cet état et les probabilités de transition restent constantes[1] au cours du temps :

$$\Pr(X_{n+1} = s_j \mid X_0 = s_{i_0}, ..., X_n = s_i) =$$

$$\Pr(X_{n+1} = s_j \mid X_n = s_i) = p_{ij} =_{def} \mathbf{P}[i, j]$$

Nous utiliserons indifféremment les deux notations pour les transitions d'état.

Comportement transitoire et stationnaire d'une DTMC

Dans ce paragraphe nous rappelons des résultats classiques en fournissant des justifications intuitives qui ne sauraient constituer des preuves mathématiques.

L'analyse du comportement transitoire ne présente pas de difficulté. Les changements d'état se font aux instants $\{1, 2, \ldots\}$. Etant données une distribution initiale $\boldsymbol{\pi}_0$ et la matrice de transition $\mathbf{P}$, alors $\boldsymbol{\pi}_n$ la distribution de X_n (*i.e.* l'état de la chaîne à l'instant n) est donnée par la formule $\boldsymbol{\pi}_n = \boldsymbol{\pi}_0 \cdot \mathbf{P}^n$ qui s'obtient à l'aide d'une récurrence élémentaire.

L'analyse du comportement asympotique des DTMC (pour un ensemble d'états quelconque) conduit à la classification suivante des états :

– Un état s est *transitoire* si la probabilité d'y revenir est inférieure à 1. Par conséquent, sa probabilité d'occurrence $\Pr(X_n = s)$ tend vers 0 lorsque n tend vers l'infini. Un état est appelé *récurrent* s'il n'est pas transitoire.

– Un état est *récurrent nul* si la durée moyenne du retour à cet état est infinie. Intuitivement, une fois atteint, cet état apparaîtra à des intervalles dont la durée moyenne tendra vers l'infini et par conséquent sa probabilité d'occurrence tendra aussi vers 0. Ce raisonnement intuitif est mathématiquement justifié.

– Un état est *récurrent non nul* si la durée moyenne du retour à cet état est finie. Si une distribution stationnaire existe alors elle est concentrée sur les états récurrents non nuls.

Nous allons détailler cette analyse dans le cas d'un espace d'états fini. Considérons le graphe construit de la manière suivante :

– l'ensemble des sommets est l'ensemble des états de la chaîne ;

– il y a un arc de s_i à s_j si $p_{ij} > 0$.

---

1. d'où le terme de chaîne *homogène* utilisé dans les études sur les chaînes de Markov en toute généralité

Etudions les composantes fortement connexes (c.f.c.) de ce graphe. Si une c.f.c. a un arc sortant, alors nécessairement, les états de cette c.f.c. sont transitoires. A l'inverse, tous les états d'une c.f.c. puits (*i.e.* sans arc sortant) sont récurrents non nuls. Dans le cas extrême où une c.f.c. puits est réduit à un état s (*i.e.* $\mathbf{P}[s,s] = 1$), on dit que s est un état *absorbant*.

Lorsque le graphe est fortement connexe, la chaîne est dite *irréductible*. Dans le cas général, chaque c.f.c. puits constitue une sous-chaîne irréductible.

Etudions l'existence d'une distribution stationnaire dans le cas d'une chaîne irréductible. Remarquons d'abord qu'elle n'est pas toujours garantie. Ainsi, une chaîne constituée de deux états s_0 et s_1, de distribution initiale concentrée en un état et où $p_{0,1} = p_{1,0} = 1$, alterne entre les deux états et ne converge donc pas vers une distribution stationnaire. En généralisant, une chaîne irréductible est dite *périodique* de période $k > 1$ si on peut partitionner les états en sous-ensembles $S_0, S_1, \ldots, S_{k-1}$ tels que des états de S_i on accède, en un pas, exclusivement aux états de $S_{(i+1) \mod k}$.

Il s'avère qu'une chaîne irréductible et apériodique (dite alors *ergodique*) admet une distribution stationnaire et que celle-ci est *indépendante de la distribution initiale*. Le calcul de cette distribution est relativement facile. En effet, on a $\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_n \cdot \mathbf{P}$. En passant à la limite (justifiée), on obtient $\boldsymbol{\pi} = \boldsymbol{\pi} \cdot \mathbf{P}$. De plus, $\boldsymbol{\pi}$ est la seule distribution solution de :

$$\mathbf{X} = \mathbf{X} \cdot \mathbf{P} \tag{8.3}$$

Remarquons qu'une distribution initiale, solution de cette équation, est *invariante* : quelque soit l'instant d'observation la distribution courante est identique à la distribution initiale. Afin de résoudre l'équation [8.3], on peut procéder à un calcul direct en complétant par l'équation de normalisation $\mathbf{X} \cdot \mathbf{1}^T = 1$ où $\mathbf{1}^T$ désigne le vecteur colonne composé de 1. Mais les calculs itératifs sont plus intéressants si l'espace d'états est de taille importante. Le plus simple consiste à itérer $\mathbf{X} \leftarrow \mathbf{X} \cdot \mathbf{P}$ [STE 94].

Intéressons-nous maintenant au cas général en supposant uniquement que les c.f.c. puits (notées $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$) sont apériodiques de distributions stationnaires $\{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k\}$. Dans ce cas, la chaîne admet aussi une distribution stationnaire (qui cette fois-ci dépend de la distribution initiale). Cette distribution est donnée par la formule $\boldsymbol{\pi} = \sum_{i=1}^{k} \Pr(\text{d'atteindre } \mathcal{C}_i) \cdot \boldsymbol{\pi}_i$. Il reste donc à calculer la probabilité d'atteindre une c.f.c. puits. On évalue cette quantité en partant d'un état fixé puis on la conditionne suivant la distribution initiale : $\Pr(\text{d'atteindre } \mathcal{C}_i) = \sum_{s \in S} \boldsymbol{\pi}_0(s) \cdot \boldsymbol{\pi}'_{\mathcal{C}_i}(s)$ où $\boldsymbol{\pi}'_{\mathcal{C}_i}(s) = \Pr(\text{d'atteindre } \mathcal{C}_i \mid X_0 = s)$. Soit $\mathbf{P}_{T,T}$ la sous-matrice de transition de la chaîne restreinte aux états transitoires et soit $\mathbf{P}_{T,i}$ la sous-matrice de transition de la chaîne des états transitoires vers les états de $\mathcal{C}_i$, alors $\boldsymbol{\pi}'_{\mathcal{C}_i} = (\sum_{n \geq 0} (\mathbf{P}_{T,T})^n) \cdot \mathbf{P}_{T,i} \cdot \mathbf{1}^T = (\mathbf{I} - \mathbf{P}_{T,T})^{-1} \cdot \mathbf{P}_{T,i} \cdot \mathbf{1}^T$. La première égalité s'obtient en conditionnant l'accessibilité de $\mathcal{C}_i$ par la longueur possible du chemin qui y conduit tandis que la seconde se vérifie immédiatement.

8.2.3. *Chaînes de Markov à temps continu*

Présentation

Une chaîne de Markov à temps continu (en anglais Continuous Time Markov Chain ou *CTMC*) a les caractéristiques suivantes :

– L'intervalle de temps entre les instants T_n est une variable aléatoire exponentielle négative dont le taux ne dépend que de l'état X_n. Autrement dit

$$\Pr(T_n \leq \tau \,|\, X_0 = s_{i_0}, ..., X_n = s_i, T_0 \leq \tau_0, ..., T_{n-1} \leq \tau_{n-1}) =$$
$$\Pr(T_n \leq \tau \,|\, X_n = s_i) = 1 - e^{\lambda_i \cdot \tau}$$

– L'état suivant un état courant ne dépend que de cet état et les probabilités de transition restent constantes[2] au cours du temps :

$$\Pr(X_{n+1} = s_j \,|\, X_0 = s_{i_0}, ..., X_n = s_i, T_0 \leq \tau_0, ..., T_{n-1} \leq \tau_{n-1}) =$$
$$\Pr(X_{n+1} = s_j \,|\, X_n = s_i) = p_{ij} =_{def} \mathbf{P}[i,j]$$

La chaîne discrète définie par la matrice $\mathbf{P}$ est appelée *chaîne incluse*. Elle observe les changements d'état de la CTMC sans tenir compte du temps écoulé. Un état de la CTMC est absorbant s'il est absorbant pour la DTMC incluse.

Comportement transitoire et stationnaire d'une CTMC

Dans les chaînes de Markov à temps continu, en raison de l'absence de mémoire de la loi exponentielle, l'évolution du SED à tout instant est uniquement conditionnée par son état courant.

Plus précisément, le processus se caractérise par sa distribution initiale $\boldsymbol{\pi}(0)$, la matrice $\mathbf{P}$ et les λ_i. Appelons $\boldsymbol{\pi}(\tau)$ la distribution de $Y(\tau)$ et $\pi_k(\tau) = \boldsymbol{\pi}(t)(s_k)$. Si δ est petit, entre τ et $\tau + \delta$ la probabilité de l'occurrence de plus d'un événement est négligeable et la probabilité d'occurrence d'un changement d'état de k à k' est approximativement égale à $\lambda_k \cdot \delta \cdot p_{kk'}$ (par définition de la loi exponentielle).

$$\pi_k(\tau + \delta) \approx \pi_k(\tau) \cdot (1 - \lambda_k \cdot \delta) + \sum_{k' \neq k} \pi_{k'}(\tau) \cdot \lambda_{k'} \cdot \delta \cdot p_{k'k}$$

D'où

$$\frac{\pi_k(\tau + \delta) - \pi_k(\tau)}{\delta} \approx \pi_k(\tau) \cdot (-\lambda_k) + \sum_{k' \neq k} \pi_{k'}(\tau) \cdot \lambda_{k'} \cdot p_{k'k}$$

2. ici aussi, on parle de chaîne *homogène*

Et finalement :

$$\frac{d\pi_k}{d\tau} = \pi_k(\tau) \cdot (-\lambda_k) + \sum_{k' \neq k} \pi_{k'}(\tau) \cdot \lambda_{k'} \cdot p_{k'k}$$

Définissons la matrice $\mathbf{Q}$ par : $q_{kk'} = \lambda_k \cdot p_{kk'}$ pour $k \neq k'$ et $q_{kk} = -\lambda_k (= -\sum_{k' \neq k} q_{kk'})$. Alors l'équation précédente se réécrit :

$$\frac{d\boldsymbol{\pi}}{d\tau} = \boldsymbol{\pi} \cdot \mathbf{Q} \tag{8.4}$$

La matrice $\mathbf{Q}$ est appelée le *générateur infinitésimal* de la CTMC. D'après l'équation [8.4], celui-ci spécifie complètement l'évolution de celle-ci.

Si cette équation établit le caractère sans mémoire d'une CTMC, elle ne fournit pas un moyen pratique de calculer le comportement transitoire de la chaîne. Afin d'y parvenir, nous décrivons une deuxième CTMC équivalente à la première du point de vue probabiliste (une technique introduite dans [JEN 53] et connue sous le nom d'*uniformisation*). Choisissons une valeur $\mu \geq Sup(\{\lambda_i\})$. Quelque soit un état atteint, la durée qui précède le prochain changement d'état suit une loi exponentielle de paramètre (uniforme) μ. Le changement d'état est quant à lui conduit par la matrice de transition $\mathbf{P}^\mu$ définie par $\forall i \neq j, \mathbf{P}^\mu[s_i, s_j] = (\mu)^{-1} \cdot \lambda_i \cdot \mathbf{P}[s_i, s_j]$. Le calcul (immédiat) du générateur infinitésimal de cette deuxième chaine montre qu'il est égal à celui de la première chaîne. On a donc affaire au même processus stochastique. La distribution transitoire $\boldsymbol{\pi}(\tau)$ s'obtient de la façon suivante. On calcule la probabilité d'être en s à l'instant τ sachant qu'il y a eu n changements d'états dans l'intervalle $[0, \tau]$. Cette probabilité est donnée par la chaîne incluse et plus précisément par $\boldsymbol{\pi}(0) \cdot (\mathbf{P}^\mu)^n$. Puis on «déconditionne» en calculant la probabilité de n changements sachant que l'intervalle entre deux changements suit la loi exponentielle. Cette probabilité est donnée par $e^{-\mu \cdot \tau} \cdot (\mu \cdot \tau)^n / n!$. On obtient donc :

$$\boldsymbol{\pi}(\tau) = \boldsymbol{\pi}(0) \cdot (e^{-\mu \cdot \tau} \sum_{n \geq 0} \frac{(\mu \cdot \tau)^n (\mathbf{P}^\mu)^n}{n!})$$

Dans la pratique, la somme infinie ne pose pas de problème car cette somme converge très rapidement et la sommation peut être stoppée dès que la précision requise est supérieure à $e^{-\mu \cdot \tau} \cdot (\mu \cdot \tau)^n / n!$.

Examinons maintenant le comportement asympotique d'une CTMC. La manière la plus simple d'analyser ce comportement consiste à étudier la chaîne incluse. Comme nous l'avons observé lors de l'approche par uniformisation, celle-ci n'est pas unique. Intéressons-nous à une DTMC obtenue avec un choix de $\mu > Sup(\{\lambda_i\})$. Dans ce cas, tout état s vérifie $\mathbf{P}^\mu[s, s] > 0$ et par conséquent chaque c.f.c. puits de cette

chaîne est ergodique. Ceci implique qu'elle admet une distribution stationnaire. Cette distribution mesure la probabilité stationnaire d'occurrence d'un état. Mais puisque la description (uniforme) de la chaîne implique un temps de séjour moyen identique dans chacun des états ($1/\mu$), elle nous fournit aussi la distribution stationnaire de la CTMC.

Dans le cas particulier (mais fréquent) où la chaine incluse est ergodique, cette distribution est obtenue par résolution de l'équation $\mathbf{X} = \mathbf{X} \cdot \mathbf{P}^{\mu}$. Nous remarquons que $\mathbf{P}^{\mu} = \mathbf{I} + (1/\mu)\mathbf{Q}$. Donc la distribution est aussi l'unique solution de l'équation :

$$\mathbf{X} \cdot \mathbf{Q} = 0 \quad \text{et} \quad \mathbf{X} \cdot \mathbf{1}^{T} = 1 \tag{8.5}$$

Par analogie, on dit alors que la CTMC est ergodique.

8.3. Modèles stochastiques de haut niveau

La notion de processus stochastique fournit un cadre probabiliste aux SED et les chaînes de Markov en constituent une catégorie dont l'analyse est réalisable. Cependant le modélisateur désire disposer d'un modèle de spécification de plus haut niveau dont la sémantique donnée par un processus stochastique permet ensuite une analyse quantitative. Il s'avère que la sémantique stochastique d'un modèle de haut niveau soulève des problèmes spécifiques. L'objectif de cette section est de montrer comment identifier et résoudre ces problèmes.

Nous avons choisi d'illustrer ces aspects sémantiques à l'aide du modèle des réseaux de Petri. Nous introduirons d'abord un modèle général puis un sous-modèle markovien afin de montrer comment déterminer les caractéristiques de la CTMC associée. Nous supposons connu du lecteur le modèle des réseaux de Petri ordinaires. A défaut, on pourra consulter [DIA 01].

8.3.1. *Les réseaux de Petri à lois générales*

L'aspect stochastique des réseaux de Petri est introduit en considérant qu'une transition possède un *délai de franchissement* variable résultant d'un tirage aléatoire dans une distribution donnée (à valeurs dans $\mathbb{R}^{+}$) et que les conflits de franchissement sont résolus par un tirage probabiliste dépendant du *poids* des transitions. Les différentes familles de réseaux de Petri stochastiques sont définies en restreignant la nature de ces distributions. Mais pour l'immédiat, nous ne faisons aucune hypothèse sur ces distributions.

Définition 1. *Un réseau de Petri stochastique (marqué) (*RDPS*)* $N = (P, T, Pre, Post, \Phi, w, m_0)$ *est défini par :*

– P*, l'ensemble fini des places ;*

– T, *avec* $P \cap T = \emptyset$, *l'ensemble fini des transitions ;*

– Pre *(resp.* $Post$*), la matrice d'incidence arrière (resp. avant) de* $P \times T$ *vers* $\mathbb{N}$ *;*

– Φ, *une fonction de* T *vers l'ensemble des distributions de v.a. à valeurs dans* $\mathbb{R}^+$, *la loi du délai des transitions ;*

– w, *une fonction de* T *vers* $\mathbb{R}^{+*}$, *le poids des transitions ;*

– m_0, *un vecteur de* $\mathbb{N}^P$, *le marquage initial.*

L'introduction des distributions et des poids ne suffit pas à spécifier le processus stochastique associé au RDPS. Nous allons successivement étudier les problèmes liés à cette spécification.

Remarque : La plupart des paramètres de ce processus peuvent être rendus dépendants du marquage courant. Par souci de simplification, nous ne le mentionnerons pas dans la discussion qui suit.

Politique de choix Etant donné le marquage initial, il s'agit de déterminer la prochaine transition à franchir parmi les transitions franchissables. Deux possibilités s'offrent à nous :

– un choix probabiliste suivant une distribution associée aux poids des transitions franchissables. On parle alors de *présélection* car ce choix a lieu avant le tirage aléatoire du délai ;

– à l'inverse, un tirage aléatoire indépendant pour chacun des délais suivi du choix du plus court délai. En cas de délais égaux, on effectue également un choix probabiliste selon les poids, appelé *post-sélection*.

La deuxième solution est systématiquement retenue car d'une part elle correspond à une modélisation plus naturelle et d'autre part parce qu'à l'aide de transitions immédiates (voir la section 8.3.4) la préselection est simulable par la post-sélection. Notons qu'excepté si les distributions des délais sont continues, il importe de spécifier les distributions des sélections (ici par le poids des transitions).

Politique de service Si, en un marquage m, une transition t a un degré de franchissement $e = \lfloor Inf(\{m(p)/Pre[p,t]\}) \rfloor > 1$, on peut considérer que le marquage *fournit* e clients à la transition vue comme un serveur. Aussi lors du tirage aléatoire du délai, trois options sont possibles selon l'événement modélisé :

– un seul tirage est effectué, la transition n'effectue qu'un service à la fois (mode *single-server*) ;

– e tirages sont effectués, la transition est un serveur *réentrant* (mode *infinite-server*) ;

– $Inf(\{e, deg(t)\})$ tirages sont effectués, la transition pouvant assurer au plus $deg(t)$ services simultanés ; ce cas généralise les deux précédents (avec $deg(t) = 1$ ou ∞) (mode *multiple-server*). Le modélisateur doit alors spécifier $deg(t)$ pour chaque transition.

Politique de mémoire Une fois la transition t franchie, quelle est l'influence d'un tirage non retenu d'une transition t' sur ses franchissements futurs ?

La première possibilité consiste à oublier le tirage effectué. Si la transition t' reste franchissable, elle donne lieu à un nouveau tirage aléatoire (mode *resampling memory*). Avec une telle sémantique, t pourrait modéliser un phénomène de panne d'un service spécifié par t'.

La deuxième possibilité consiste à mémoriser le tirage décrémenté du tirage retenu, uniquement si t' reste franchissable (mode *enabling memory PRD* (*Preemptive Repeat Different*)). Si t' devient infranchissable, ce mécanisme modélise un *time-out* (t') désarmé par t.

La troisième possibilité est équivalente pour une transition encore franchissable mais conserve le tirage inchangé si t' devient infranchissable. Ce tirage sera utilisé à nouveau lorsque t' redevient franchissable (mode *enabling memory PRI* (*Preemptive Repeat Identical*)). Une transition t' devenue infranchissable pourrait modéliser un travail avorté par t pour être recommencé plus tard.

La quatrième possibilité consiste à mémoriser le tirage décrémenté du tirage retenu. Une transition t' devenue infranchissable pourrait modéliser un travail suspendu par t (mode *age memory* aussi appelé *PRS* (*Preemptive ReSume*)).

Pour compléter cette politique, nous devons prendre en compte le cas des transitions réentrantes (*multiple-server*), ce qui nécessite parfois de choisir quels tirages conserver, suspendre ou oublier. La solution la plus simple est une politique *FIFO* des tirages. Le dernier tirage effectué est le premier suspendu ou oublié. D'autres politiques (comme suspendre ou oublier le client le moins engagé) ne sont pas nécessairement compatibles avec certaines méthodes d'analyse.

Il est clair qu'une fois définies ces trois politiques, le processus stochastique est déterminé sans ambiguïté. Nous allons maintenant restreindre la nature des distributions des délais.

8.3.2. *Les réseaux de Petri à lois exponentielles*

Dans le modèle de base [FLO 85, MOL 81] (appelé en anglais *Stochastic Petri Net* ou *SPN*), chaque transition t a un délai exponentiel de taux $\lambda(t)$ (en supposant une énumération de T, on notera $\lambda_k = \lambda(t_k)$).

Examinons le processus stochastique engendré par un tel réseau en mode *single-server*. Soit m un marquage donné, $t_1, \ldots, t_k$ les transitions franchissables à partir de m. On vérifie que :

– le temps de séjour en m suit une loi exponentielle de taux $\lambda_1 + \ldots + \lambda_k$;

– la probabilité de franchir t_i avant les autres transitions est égale à $\frac{\lambda_i}{\sum \lambda_j}$ et qu'elle est indépendante du temps de séjour dans le marquage ;

– la distribution du temps résiduel du délai de franchissement de t_i si t_j est franchie est égale à la distribution initiale (loi sans mémoire).

Autrement dit, seul le nouveau marquage détermine le comportement futur du processus stochastique. On a donc affaire à une chaîne de Markov à temps continu, isomorphe au graphe d'accessibilité du réseau, dont tous les paramètres sont donnés par les états (*i.e.* les marquages). Ce raisonnement se généralise aux autres modes de fonctionnement. Sous réserve que ce graphe soit fini, l'analyse transitoire et stationnaire de ce modèle s'effectue alors comme indiqué en section 8.2.3.

8.3.3. *Indices de performance des réseaux de Petri*

Puisque l'état d'un réseau de Petri est un marquage, les fonctions associées aux indices s'expriment comme des expressions numériques où les variables sont les marquages possibles des places (x_p représentant le marquage de la place p). Par exemple, supposons l'existence d'une place cli indiquant le nombre de clients et d'une place $panne$ indiquant le fait que le serveur soit en panne. Alors l'expression x_{cli} fournit le nombre de clients tandis que $x_{cli} \geq 1 \wedge x_{panne} = 1$ représente le fait qu'au moins un client attende le service alors qu'il est en panne.

Notons que, moyennant des raisonnements *ad hoc*, il est possible de calculer des indices relatifs aux transitions. Par exemple, supposons que t représente l'arrivée d'un client et que l'on veuille calculer la probabilité qu'un client en arrivant trouve le serveur en panne. Il nous faut identifier les marquages où t est franchissable. Pour chacun de ces marquages m, nous calculons la probabilité que t soit la prochaine transition franchie. Dans les réseaux de Petri à distributions exponentielles, cette probabilité que nous noterons $\pi_{tir}(m,t)$, s'obtient comme le ratio de son taux sur la somme des taux des transitions franchissables en m. Soit $\boldsymbol{\pi}$ la distribution pour laquelle on veut mesurer l'indice, alors la valeur recherchée est :

$$\frac{\sum_{m \xrightarrow{t} m' \wedge m'(off)=1} \boldsymbol{\pi}(m) \cdot \pi_{tir}(m,t)}{\sum_{m \xrightarrow{t} m'} \boldsymbol{\pi}(m) \cdot \pi_{tir}(m,t)}$$

8.3.4. *Panorama des modèles et des méthodes d'evaluation de performance*

Le domaine de l'évaluation de performance est extrêmement vaste en raison d'un long passé lié aux débuts des télécommunications. Nous nous limiterons donc à un survol en renvoyant à des ouvrages de référence le lecteur soucieux d'approfondir tel ou tel sujet particulier.

Les premiers modèles d'évaluation sont les files d'attente, puis les réseaux de files d'attente [KLE 75, KLE 76]. Ce modèle met principalement l'accent sur le type de loi envisagé, la nature et le nombre des clients, la discipline de service et le routage entre files. Cependant, s'il est adapté à l'étude des réseaux de télécommunications, il se révèle inapproprié lorsque les systèmes (tels les environnements informatiques) comportent des synchronisations entre composants. On peut pallier à cette insuffisance en ajoutant des mécanismes *ad hoc* mais il vaut mieux «probabiliser» les modèles qualitatifs des systèmes concurrents.

Les réseaux de Petri stochastiques bénéficient de l'expérience de nombreuses modélisations en réseaux de Petri ordinaires [Ajm 95]. Sa règle de franchissement élémentaire permet facilement de le transformer en modèle stochastique comme nous l'avons fait au paragraphe précédent. De plus, les réseaux de Petri de haut-niveau qui incorporent la structuration des données et le paramétrage des actions ont aussi servi de support à une version stochastique [CHI 93b].

Les algèbres de processus présentent l'avantage d'intégrer dans leur définition les aspects compositionnels et sont donc adaptées à un processus de modélisation hiérarchique. Aussi, elles ont également fait l'objet d'une extension stochastique [HIL 96]. Contrairement aux réseaux de Petri, la sémantique stochastique de tels modèles pose des problèmes subtils comme par exemple la spécification quantitative du mécanisme de synchronisation d'actions.

Les méthodes d'évaluation sont généralement classées selon la complexité du calcul. Nous suivrons cet ordre en nous limitant à des références relatives au modèle des réseaux de Petri par souci de concision.

Lorsque la structure d'un modèle est particulièrement simple, il est possible d'obtenir une formule exprimant la distribution stationnaire en fonction des paramètres numériques. Dans le cas d'une formule obtenue à partir de formules associées au composant du système, on parle de forme produit. Dans le cadre des réseaux de Petri, une forme produit est difficile à obtenir en raison des synchronisations effctuées par le franchissement d'une transition. Cependant pour certaines sous-classes de réseaux de Petri de telles formules ont été établies [HEN 90, HAD 05].

Dans le cas général, il est nécessaire de générer la CTMC associée au réseau de Petri stochastique et de calculer sa distribution stationnaire. Lorsque la taille de la

chaîne est trop importante, une analyse de la structure du réseau permet de calculer des bornes [CHI 93a] ou de concevoir des méthodes approchées [CAM 94]. Dans le cas de réseaux de haut niveau, les techniques d'agrégation *a priori* génèrent une CTMC plus petite mais équivalente à la CTMC du réseau [CHI 93b]. Lorsque le réseau de Petri est non borné (*i.e.* le marquage des places du réseau peut être aussi grand que possible), la CTMC associée est infinie. Cependant, si une seule place est non bornée, il est encore possible d'obtenir la distribution stationnaire [HAV 95].

Si les lois exponentielles conviennent pour la modélisation d'événements dont la distribution temporelle n'est pas connue, certaines opérations ont une durée comprise dans un intervalle de temps ou même assimilable à une constante. Dans ce cas, le choix d'une loi exponentielle conduit à des mesures très approximatives. Aussi une étude des réseaux comprenant des transitions à loi déterministe (appelée aussi loi de Dirac). a été entreprise [Ajm 87, LIN 98]. De nombreuses approches alternatives ont ensuite été proposées selon la nature des lois et l'occurence des transitions correspondantes [DON 98, GER 99, LIN 98, LIN 99].

8.3.5. *L'outil GreatSPN*

GreatSPN [Ajm 95] est l'un des plus importants outils d'analyse qualitative et quantitative de réseaux de Petri, développé depuis le début des années 1980 par l'équipe d'évaluation de performance de l'Université de Turin. Prenant en compte initialement les réseaux de Petri stochastiques, il a progressivement intégré les extensions sémantiques de ce modèle relatives à la distribution des transitions. Il est également le seul logiciel à gérer le modèle de réseau de Petri stochastique bien formé, un modèle de haut niveau (coloré) qui tire avantage des symétries de comportement afin d'obtentir une CTMC agrégée à partir de la définition du réseau. GreatSPN est disponible auprès des auteurs, libre de droits pour un usage académique. Le logiciel possède une interface graphique pour la définition du réseau, des propriétés complémentaires non graphiques (durées, etc.) et le lancement des calculs. La plupart des résultats sont aussi disponibles dans l'interface graphique.

• Les modèles supportés - GreatSPN gère d'une part les réseaux de Petri non colorés dont les transitions ont des distributions exponentielles négatives (SPN de base), à loi de type phase (Erlang, etc.), déterministes (constantes) et à durée nulle (transitions immédiates).
GreatSPN prend en compte le modèle de haut niveau (S)WN ((Stochastic) Well formed Net), le seul modèle de réseau stochastique coloré largement utilisé. Les paramètres de ce modèle se saisissent également *via* l'interface graphique.

• Analyse qualitative des réseaux de Petri - L'outil intègre la plupart des méthodes d'analyse standard de réseaux de Petri : détermination du caractère borné, calcul des invariants, des trappes et siphons, etc. Dans le cas où il est fini, GreatSPN calcule et

stocke sur disque le graphe d'accessibilité du réseau. Pour les WN, les calculs sont effectués au niveau d'un graphe agrégé (le graphe symbolique d'accessibilité).

• Analyse de performance des réseaux de Petri stochastiques - GreatSPN calcule les distributions stationnaires et transtoires des modèles finis. L'utilisateur peut également définir des indices de performance complexes que l'outil calculera sur demande. Le logiciel fournit également les distributions et indices de performances des SWN, au niveau agrégé et si besoin au niveau non agrégé.
Malgré une optimisation de la représentation des états (les marquages du réseau), la taille du graphe d'accessibilité empêche la résolution exacte de très grands modèles. Aussi GreatSPN comporte un simulateur stochastique pour faire face à de telles situations.

L'implémentation de GreatSPN est fondée sur une structure modulaire qui suit la démarche d'analyse des réseaux de Petri (définition, analyse qualitative, évaluation de performance). Chacune de ces étapes est réalisée par un code écrit en langage C, dont l'exécutable s'invoque également en ligne de commande. Ceci autorise une utilisation programmée (sous forme de scripts) des différents algorithmes implantés dans l'outil. Le lecteur pourra consulter le site `www.unito.it/~GreatSPN` pour des informations détaillées.

8.4. Vérification probabiliste de chaînes de Markov

La présentation détaillée portera sur les CTMC.

8.4.1. *Limites des indices de performance standard*

Les indices de performance définis lors de la section précédente apportent des informations précieuses au concepteur du système. Cependant ils ne répondent pas à tous les besoins en terme d'évaluation. Illustrons ce point à l'aide de l'exemple de la disponibilité d'un service. Voici quelques propriétés relatives à ce concept :

– Garantie de disponibilité *instantanée* en régime transitoire. Il s'agit de la probabilité à un instant τ de la disponibilité du service.

– Garantie de disponibilité instantanée en régime stationnaire. Il s'agit de la probabilité à un instant donné de la disponibilité du service en régime stationnaire.

– Garantie de disponibilité *dans la durée* en régime transitoire. Il s'agit de la probabilité que le service soit constamment disponible entre deux instants τ et τ'.

– Garantie de disponibilité dans la durée en régime stationnaire. Il s'agit de la probabilité que le service soit constamment disponible entre deux instants en régime stationnaire. Puisque le processus est en régime stationnaire. Cette mesure ne dépend que la durée de l'intervalle constitué des deux instants.

– Garantie de disponibilité et de temps de réponse en régime stationnaire. Il s'agit la probabilité qu'après une requête, le service soit fonctionnel jusqu'à la réponse et que le temps de réponse n'excède pas une borne donnée.

Si les deux premières propriétés se déduisent facilement des distributions stationnaires et transitoires, il n'en est pas de même des autres. On pourrait imaginer un algorithme ad hoc pour chacune de celles-ci. Mais, il est plus judicieux d'introduire une logique afin d'exprimer des indices de performance complexes et de concevoir un algorithme général d'évaluation de formules de cette logique.

8.4.2. *Une logique temporelle pour les chaînes de Markov*

La logique temporelle CSL («Continuous Stochastic Logic») que nous allons détailler est une adaptation de la logique CTL («Computation Tree Logic» [EME 80]) aux chaînes de Markov à temps continu. Elle exprime des formules *qui s'évaluent sur les états* et dont la syntaxe est la suivante. Ici, nous suivrons principalement l'approche de [BAI 03a].

Définition 2. *Une formule de CSL est définie inductivement par :*

– Si $\phi \in \mathcal{P}$ alors ϕ est une formule de CSL ;

– Si ϕ et ψ sont des formules de CSL alors $\neg\phi$ et $\phi \wedge \psi$ sont des formules de CSL ;

– Si ϕ est une formule de CSL, $a \in [0,1]$ est un réel, $\bowtie \in \{<, \leq, >, \geq\}$ alors $S_{\bowtie a}\phi$ est une formule de CSL ;

– Si ϕ et ψ sont des formules de CSL, $a \in [0,1]$ est un réel, $\bowtie \in \{<, \leq, >, \geq\}$ et I est un intervalle de $\mathbb{R}_{\geq 0}$ alors $P_{\bowtie a}\mathcal{X}^I\phi$ et $P_{\bowtie a}\phi\mathcal{U}^I\psi$ sont des formules de CSL.

Seule l'interprétation des deux derniers points nécessite quelques explications. La formule $S_{\bowtie a}\phi$ est satisfaites par s un état de la chaîne si, pour le processus démarré en s, la probabilité stationnaire cumulée (disons p) des états qui satisfont ϕ vérifie $p \bowtie a$. L'évaluation de cette formule est bien définie car pour une CTMC finie, une distribution stationnaire existe toujours. Notons que l'évaluation de cette formule est indépendante de l'état considéré si la chaîne est ergodique.

Une réalisation du processus stochastique satisfait $\mathcal{X}^I\phi$ si le premier changement d'état a lieu dans l'intervalle I et l'état atteint vérifie ϕ. Un état s satisfait $P_{\bowtie a}\mathcal{X}^I\phi$ si la probabilité (disons p) qu'une réalisation du processus démarré en s satisfasse la contrainte énoncée vérifie $p \bowtie a$.

Une réalisation du processus stochastique satisfait $\phi\mathcal{U}^I\psi$ s'il existe un instant $\tau \in I$ tel que ψ soit satisfait et qu'à tous les instants précédents ϕ soit satisfait. Un état s satisfait $P_{\bowtie a}\phi\mathcal{U}^I\psi$ si la probabilité (disons p) qu'une réalisation du processus démarré en s satisfasse la contrainte énoncée vérifie $p \bowtie a$.

A titre d'exemple, nous formalisons maintenant les propriétés de disponibilité énoncées plus haut.

– Garantie de disponibilité instantanée en régime transitoire de 99% :

$$P_{\geq 0.99} true \mathcal{U}^{[\tau,\tau]} disp$$

où $disp$ est une proposition atomique indiquant si le service est disponible.

– Garantie de disponibilité instantanée en régime stationnaire de 99% :

$$S_{\geq 0.99} disp$$

– Garantie de disponibilité dans la durée en régime transitoire de 99% :

$$P_{<0.01} true \mathcal{U}^{[\tau,\tau']} \neg disp$$

– Garantie de disponibilité dans la durée en régime stationnaire de 99% :

$$S_{<0.01} true \mathcal{U}^{[\tau,\tau']} \neg disp$$

– Garantie de disponibilité et de temps de réponse (3 unités de temps) en régime stationnaire de 99% :

$$S_{\geq 0.99}(req \Rightarrow P_{\geq 0.99}(disp \mathcal{U}^{[0,3]} acq))$$

où req est une proposition atomique indiquant une réception de requête et acq est une proposition atomique indiquant une réponse à une requête. On notera qu'en réalité les deux occurrences de 99% n'ont pas la même signification. Celle correspondant à l'opérateur interne est une exigence sur le comportement du processus démarré en un état particulier tandis que la deuxième occurrence est une exigence globale sur les états pondérée par une distribution stationnaire. *A priori*, des valeurs différentes d'exigence auraient pu être spécifiées.

8.4.3. *Algorithme de vérification*

Etant données une CTMC et une formule ϕ de CSL, l'algorithme de vérification procède par évaluation successive des sous-formules de ϕ en «remontant» l'arbre syntaxique de la formule ϕ des feuilles à la racine et en étiquetant chaque état avec les sous-formules qu'il vérifie. Ainsi chaque étape de l'algorithme évalue une formule en interprétant les opérandes de l'opérateur le plus externe comme des propositions atomiques.

Nous sommes donc conduits à étudier chaque opérateur.

$\boxed{\phi = \neg\psi}$ L'algorithme étiquette avec ϕ chaque état non étiqueté avec ψ.

$\boxed{\phi = \psi \wedge \chi}$ L'algorithme étiquette avec ϕ chaque état étiqueté avec ψ et χ.

$\boxed{\phi = S_{\bowtie a}\psi}$ L'algorithme calcule la distribution stationnaire du processus démarré en s (ainsi qu'indiqué à la section 8.2.3). Puis il cumule les probabilités des états étiquetés par ψ et étiquette s avec ϕ si la quantité obtenue (disons p) vérifie $p \bowtie a$. Notons que pour les états d'une c.f.c. puits, un seul calcul est nécessaire pour tous les états de la c.f.c. De même, si la CTMC admet une unique distribution stationnaire alors la formule a une valeur de vérité indépendante de l'état.

$\boxed{\phi = P_{\bowtie a}\mathcal{X}^I\psi}$ Soit un état s, l'occurrence de la prochaine transition dans l'intervalle I et la satisfaction de ψ par l'état atteint sont deux événements indépendants. La probabilité recherchée est donc le produit de la probabilité de chacun de ces événements. Notons $I = [\tau, \tau']$; nous supposons ici sans perte de généralité que les intervalles sont fermés. En effet, en raison de la continuité des distributions, le fait que les bornes supérieures et inférieures de l'intervalle en fassent partie n'a pas d'incidence sur l'évaluation de la formule. Soit $\mathbf{Q}$ le générateur infinitésimal de la chaîne et $\mathbf{P}$ la matrice de transition de la chaîne incluse, alors la probabilité du premier événement est donnée par $e^{\tau\mathbf{Q}[s,s]} - e^{\tau'\mathbf{Q}[s,s]}$ tandis que celle du second événement est donnée par $\sum_{s' \vDash \psi} \mathbf{P}[s, s']$.

$\boxed{\phi = P_{\bowtie a}\psi\mathcal{U}^I\chi}$ L'évaluation de cette formule consiste essentiellement à effectuer une analyse transitoire de chaînes obtenues par des transformations élémentaires à partir de la chaîne originelle. Ainsi soit X une chaîne, alors X^ϕ est la chaîne obtenue en rendant absorbants les états qui vérifient ϕ. Afin de simplifier la présentation, nous étudions les différents types d'intervalle.

– $\phi = P_{\bowtie a}\psi\mathcal{U}^{[0,\infty[}\chi$. Dans ce cas, la réalisation du processus doit rester dans des états qui vérifient ψ jusqu'à ce qu'un état vérifiant χ soit atteint et ceci sans contrainte de temps. Autrement dit, on suit le comportement de la chaîne jusqu'à ce qu'on rencontre un état qui vérifie $\neg\psi \vee \chi$. Etudions la chaîne $X^{\neg\psi\vee\chi}$. Si une c.f.c. puits de cette chaîne contient un état qui vérifie χ alors la probabilité recherchée est 1 pour tous les états de cette c.f.c. (car tous les états d'une c.f.c. puits sont récurrents) sinon cette probabilité est nulle. Appelons une c.f.c. associée à une probabilité 1, une "bonne" c.f.c. Par conséquent, la probabibilité recherchée pour les états restants est égale à la probabilité d'atteindre un état d'une bonne c.f.c. Cette probabilité ne dépend que de la chaîne incluse de $X^{\neg\psi\vee\chi}$ et son calcul a déjà été décrit à la section 8.2.2.

– $\phi = P_{\bowtie a}\psi\mathcal{U}^{[0,\tau]}\chi$. Dans ce cas, la réalisation du processus doit rester dans des états qui vérifient ψ jusqu'à ce qu'un état vérifiant χ soit atteint et ceci au plus tard à l'instant τ. Autrement dit on suit le comportement de la chaîne jusqu'à ce qu'on rencontre un état qui vérifie $\neg\psi \vee \chi$. La probabilité à calculer est donc égale à $\Pr(X^{\neg\psi\vee\chi}(\tau) \vDash \chi \mid X^{\neg\psi\vee\chi}(0) = s)$.

– $\phi = P_{\bowtie a}\psi\mathcal{U}^{[\tau,\tau]}\chi$. Dans ce cas, la réalisation du processus doit rester dans des états qui vérifient ψ durant l'intervalle $[0, \tau]$ et de plus vérifier χ à l'instant τ. On néglige la possibilité d'un changement d'état à l'instant τ car sa probabilité est nulle.

Par conséquent, la probabilité à calculer est égale à $\Pr(X^{\neg\psi}(\tau) \vDash \psi \wedge \chi \mid X^{\neg\psi}(0) = s)$.

– $\phi = P_{\bowtie a}\psi\mathcal{U}^{[\tau,\infty[}\chi$. Dans ce cas, la réalisation du processus doit rester dans des états qui vérifient ψ durant l'intervalle $[0,\tau]$ puis à partir de l'état atteint s à l'instant τ verifier la formule $\psi\mathcal{U}^{[0,\infty[}\chi$. La probabilité recherchée est donc $\sum_{s'\vDash\psi} \Pr(X^{\neg\psi}(\tau) = s' \mid X^{\neg\psi}(0) = s) \cdot \boldsymbol{\pi}(s')$ où $\boldsymbol{\pi}(s')$ est calculée suivant la procédure du premier cas.

– $\phi = P_{\bowtie a}\psi\mathcal{U}^{[\tau,\tau']}\chi$. Un raisonnement similaire au cas précédent conduit la formule suivante pour la probabilité recherchée : $\sum_{s'\vDash\psi} \Pr(X^{\neg\psi}(\tau) = s' \mid X^{\neg\psi}(0) = s) \cdot \Pr(X^{\neg\psi\vee\chi}(\tau'-\tau) \vDash \chi \mid X^{\neg\psi\vee\chi}(0) = s')$.

8.4.4. *Panorama de la vérification probabiliste de chaînes de Markov*

Historiquement, la vérification des chaînes à temps discret a précédé celle des chaînes à temps continu. La première approche de vérification de formules LTL sur des DTMC (proposée dans [VAR 85]) est conceptuellement simple : traduire la formule en un automate de Büchi, puis déterminiser cet automate en un automate de Rabin, effectuer le produit synchronisé de cet automate avec la DTMC ce qui conduit à une nouvelle DTMC sur laquelle une variante de l'analyse vue à la section 8.2 fournit la probabilité recherchée. Cependant la complexité de cet algorithme est doublement exponentielle relativement à la taille de la formule. Dans [COU 95], les auteurs construisent aussi une nouvelle DTMC en raffinant itérativement la DTMC initiale par une analyse des opérateurs de la formule. Ceci conduit à une procédure simplement exponentielle. Ils démontrent de plus qu'il s'agit là de la complexité optimale. Un troisième algorithme [COU 03] traduit aussi la formule en un automate de Büchi. Cependant le choix de l'algorithme de traduction permet d'évaluer la probabilité associée à la formule directement à partir du produit synchronisé de l'automate et de la DTMC. Cette méthode a aussi une complexité théorique optimale et se comporte mieux dans les cas pratiques que la précédente.

Une technique classique d'analyse des modèles de performance consiste à associer des «récompenses» aux états et/ou aux transitions d'une chaîne et de calculer des indices de performance relatifs à ces récompenses. Afin d'étendre la portée de la vérification probabiliste à ce type de modèle, une nouvelle logique PRCTL est introduite dans [AND 03] accompagnée d'un algorithme d'évaluation de formules.

Les premiers travaux significatifs relatifs aux CTMC ont été établis dans [AZI 96, AZI 00]. Il y est démontré que la vérification de formules CSL sur les CTMC est décidable. Cependant l'algorithme correspondant est extrêmement complexe car il «s'interdit» les approximations que nous avons implicitement faites lors des calculs du paragraphe précédent.

De fait, même avec la méthode décrite plus haut, le calcul peut s'avérer impraticable pour des CTMC de grande taille. Une approche efficace pour faire face à ce problème consiste à tirer profit de la modularité de la spécification. On cherche alors à remplacer un module par un module plus petit mais équivalent vis à vis de la formule à vérifier. On procède ensuite à la vérifcation du modèle composé des modules réduits. Cette démarche initiée par [BAI 03a] a été généralisée dans [BAI 03b] où de nombreuses formes d'équivalence sont étudiées.

Une approche radicalement différente pour réduire la complexité de la vérification est proposée dans [YOU 05]. Supposons que nous devions vérifier la formule $P_{\leq a}\phi$, nous pouvons générer des exécutions aléatoires et calculer le ratio des exécutions qui vérifient ϕ ; en vertu de résultats classiques de probabilité, cette valeur tend vers la probabilité recherchée. Lorsque l'évaluation de la formule ϕ ne requiert qu'une exécution temporellement bornée, cette méthode est particulièrement efficace.

8.4.5. *L'outil ETMCC*

ETMCC (Erlangen-Twente Markov Chain mdel Checker) est le premier outil (2000) de vérification de formules CSL sur les modèles CMTC. Il est développé par plusieurs universités allemandes, Erlangen à l'origine, et l'université néerlandaise de Twente. Il est disponible en version limitée après inscription auprès des auteurs. ETMCC permet d'analyser des CMTC avec des propriétés en logique CSL étendue sur plusieurs points pratiques du point de vue du modélisateur dont les récompenses. L'utilisation est classique : spécification du modèle dans un formalisme spécifique, définition des propriétés en logique temporelle probabilisée puis calcul et analyse des résultats. Même si ETMCC analyse aussi des modèles en temps discret, nous présentons ici les fonctionnalités pour les systèmes à temps continu qui constituent sa «spécialité». Le cas discret a d'ailleurs fait l'objet de travaux croisés avec l'équipe de l'outil PRISM (voir plus loin).

• Langage des modèles de systèmes - ETMCC a choisi une solution très simple : les modèles sont des CTMC pour lesquelles l'utilisateur fournit une version textuelle de la matrice des vitesses sous forme creuse (éléments non nuls). Notons qu'un tel choix permet d'utiliser ETMCC à partir de divers modèles de plus haut niveau (algèbres de processus, réseaux de Petri, ...) du moment que l'on sait générer leurs espaces d'états (finis) : il suffit ce convertir ces descriptions au format textuel d'entrée de ETMCC. La liste des propriétés atomiques satisfaites en chaque état doit également être fournie dans un fichier texte.

• Langage des propriétés - ETMCC permet de définir les propriétés à vérifier dans la logique CSL détaillée ci-dessus. Ces propriétés sont saisies dans l'interface utilisateur graphique. CSL est étendue au calcul de récompenses moyennes en régime transitoire ou à l'équilibre, à partir des fonctions de récompense définies pour les états et les

événements ; ceci permet d'obtenir des indicateurs utiles au praticien comme le coût moyen d'une transaction sur une durée donnée.

• Résultats calculés - Le logiciel analyse le modèle conformément aux propriétés demandées et présente les résultats, valeurs de vérité des formules de CSL et indicateurs de récompense, dans l'interface.

L'implémentation de ETMCC est réalisée entièrement en Java. La partie la plus délicate concerne l'évaluation des opérateurs à temps borné (Next et Until). Elle met en jeu des solveurs par intégration numérique ou par uniformisation de CTMC. L'ensemble du logiciel utilise également l'algorithmique des graphes, en particulier pour la détermination des composantes fortement connexes. Le graphe des états accessibles est représenté sous forme creuse. Cette représentation est également exploitée par tous les calculs numériques. Le lecteur se reportera au site `http://www7.informatik.uni-erlangen.de/etmcc` pour une description détaillée de l'outil.

8.5. Les processus de décision markoviens

8.5.1. *Présentation des processus de décision markoviens*

Supposons que nous devions analyser l'exécution d'un ensemble de transactions telles que chacune d'entre elles puisse être modélisée par une DTMC. Dans la recherche d'une représentation du système complet, nous sommes alors confrontés au fait que l'ordonnanceur du système transactionnel nous est inconnu. Nous pourrions représenter les choix de l'ordonnanceur par des actions probabilistes et se ramener ainsi à une DTMC globale. Mais cette solution limite considérablement la portée des mesures ainsi obtenues. En effet, les résultats s'interpréteraient comme des indices de performance valables pour un ordonnanceur « moyen ». Or dans la pratique, les indices que l'on recherche sont des indices extrêmes tels que la probabilité maximale d'un abandon de transaction en considérant l'ensemble (infini) des ordonnanceurs possibles.

Il convient donc d'adopter un formalisme plus expressif que celui des DTMC. Plus précisement, ce formalisme doit permettre d'exprimer à la fois des choix probabilistes et des choix non déterministes. Ceci nous conduit naturellement aux *processus de décision markoviens* (PDM).

Définition 3. *Un processus de décision markovien* $\Sigma = (S, \mathcal{P}, V, next)$ *est défini par :*

– S*, l'ensemble (fini) des états ;*

– $\mathcal{P}$*, l'ensemble (fini) des propositions atomiques ;*

– V*, la fonction caractéristique des états qui associe à chaque état* s*, le sous-ensemble* $V(s)$ *de* $\mathcal{P}$ *des propositions qui sont vraies en* s *;*

– $next$ la fonction qui associe à chaque état s, l'ensemble $next(s) = \{\boldsymbol{\pi}_1^s, \ldots, \boldsymbol{\pi}_{k_s}^s\}$ de distributions dont le support est S.

L'ingrédient fondamental de cette définition est la fonction $next$ qui contrôle l'évolution du processus. Dans un état s, le processus choisit de manière non déterministe une distribution $\boldsymbol{\pi}_i^s \in next(s)$, puis effectue un tirage probabiliste selon cette distribution ce qui détermine l'état suivant. En accord avec cette interprétation, nous introduisons une relation de succession (immédiate) ρ définie par $\rho(s, s') \Leftrightarrow \exists \boldsymbol{\pi}_i^s \ \boldsymbol{\pi}_i^s(s') > 0$. Un chemin d'exécution est alors une suite d'états tels que tout couple de deux états consécutifs satisfasse cette relation.

Nous désirons placer ce système dans un cadre probabiliste «pur». A cette fin, nous définissons la notion de stratégie. Une stratégie St est une fonction qui associe à un chemin d'exécution $s_0, s_1, \ldots, s_n$, $St(s_0, s_1, \ldots, s_n)$ une distribution appartenant à $next(s_n)$. Pour une stratégie fixée et un état initial donné, le processus de décision markovien se comporte comme un processus stochastique à temps discret et par conséquent, la probabilité d'un événement Ev de ce processus est bien définie et sera notée $\Pr^{St}(Ev)$.

8.5.2. *Une logique temporelle pour les processus de décision markoviens*

La logique temporelle pCTL («probabilistic Computation Tree Logic») que nous allons détailler est une adaptation de CTL aux processus de décision markoviens. Ici, nous suivons principalement l'approche de [BIA 95]. Les formules de cette logique s'évaluent sur les états et leur syntaxe est la suivante.

Définition 4. *Une formule de pCTL est définie inductivement par :*

– Si $\phi \in \mathcal{P}$ alors ϕ est une formule de pCTL ;

– Si ϕ et ψ sont des formules de pCTL alors $\neg\phi$ et $\phi \wedge \psi$ sont des formules de pCTL ;

– Si ϕ et ψ sont des formules de pCTL, $a \in [0, 1]$ est un réel et $\bowtie \in \{<, \leq, >, \geq\}$ alors $A\phi\mathcal{U}\psi$, $E\phi\mathcal{U}\psi$ et $P_{\bowtie a}\phi\mathcal{U}\psi$ sont des formules de pCTL.

Seule l'interprétation du dernier point nécessite quelques explications. Les deux premiers opérateurs ne font pas intervenir les valeurs numériques des distributions. $A\phi\mathcal{U}\psi$ (resp. $E\phi\mathcal{U}\psi$) est vrai en un état s ssi tout (resp. au moins un) chemin d'exécution à partir de s comprend un préfixe constitué d'états qui satisfont ϕ suivi d'un état qui satisfait ψ. Le dernier opérateur fait intervenir les stratégies de la manière suivante : $P_{\bowtie a}\phi\mathcal{U}\psi$ est vrai en s si *pour toute stratégie*, la probabilité (disons p) pour le processus stochastique associé qu'un chemin d'exécution issu de s comprenne un préfixe constitué d'états qui satisfont ϕ suivi d'un état qui satisfait ψ vérifie $p \bowtie a$.

8.5.3. *Algorithme de vérification*

Etant donné un processus de décision markovien et une formule ϕ de pCTL, l'algorithme de vérification procède par évaluation successive des sous-formules de ϕ en «remontant» l'arbre syntaxique de la formule ϕ des feuilles à la racine et en étiquetant chaque état avec les sous-formules qu'il vérifie. Ainsi chaque étape de l'algorithme évalue une formule en interprétant les opérandes de l'opérateur le plus externe comme des propositions atomiques.

Nous sommes donc conduits à étudier chaque opérateur.

$\boxed{\phi = \neg\psi}$ L'algorithme étiquette avec ϕ chaque état non étiqueté avec ψ.

$\boxed{\phi = \psi \wedge \chi}$ L'algorithme étiquette avec ϕ chaque état étiqueté avec ψ et χ.

$\boxed{\phi = E\psi\mathcal{U}\chi}$ Dans un premier temps, l'algorithme étiquette les états étiquetés avec χ. Puis en remontant à partir de ces états à l'aide de la relation de prédécesseur (ρ^{-1}) il étiquette les états étiquetés avec ψ. Il itère cette étape à partir des états nouvellement étiquetés jusqu'à saturation.

$\boxed{\phi = A\psi\mathcal{U}\chi}$ L'algorithme emploie une fonction récursive *en marquant* les états visités. Lorsqu'il évalue un état étiqueté par χ, il l'étiquette avec ϕ et renvoie vrai. Lorsqu'il évalue un état non étiqueté par χ et par ψ, il renvoie faux. Lorsqu'il évalue un état non étiqueté par χ mais étiqueté par ψ et *non encore visité*, il appelle la fonction pour chacun des successeurs de l'état et l'étiquette par ϕ si tous les appels renvoient vrai. Lorsqu'il évalue un état *déjà visité* et non étiqueté, il renvoie faux. Le lecteur vérifiera que cette procédure renvoie faux s'il existe un chemin issu de l'état qui comprend un état non étiqueté par ϕ ou χ avant tout état étiqueté par χ ou un chemin (infini) constitué d'états étiquetés par ϕ mais pas par χ. Ce dernier cas est détecté par l'existence d'un circuit à l'aide du marquage des états.

$\boxed{\phi = P_{\geq a}\psi\mathcal{U}\chi}$ Nous traiterons uniquement ce cas d'opérateur probabiliste, les autres cas étant similaires. L'algorithme calcule simultanément pour tous les états la probabilité minimale (disons $\boldsymbol{\pi}_{min}(s) = Inf(\{\Pr^{St}(s \vDash \psi\mathcal{U}\chi)\})$) des «bons» chemins puis la compare avec a afin de déterminer les états à étiqueter.

Si un état vérifie χ, alors quelque soit la stratégie St, $\Pr^{St}(s \vDash \psi\mathcal{U}\chi) = 1$ et par conséquent $\boldsymbol{\pi}_{min}(s) = 1$. Appelons ce sous-ensemble d'états S_{good}. A partir de S_{good}, on obtient l'ensemble des états pour lesquels $\boldsymbol{\pi}_{min}(s) > 0$. Cet ensemble, noté $S_{>0}$, est obtenu en l'initialisant à S_{good} puis en l'élargissant (itérativement) avec les états qui, quelque soit la stratégie adoptée, ont une probabilité non nulle de l'atteindre en un pas : $S_{>0} \leftarrow S_{>0} \cup \{s \mid \forall \boldsymbol{\pi}^i_s, \exists s' \in S_{>0}, \boldsymbol{\pi}^i_s(s') > 0\}$. Cette procédure se termine nécessairement. Appelons $S_{bad} = S \setminus S_{>0}$. On vérifie aisément que $\forall s \in S_{bad}, \boldsymbol{\pi}_{min}(s) = 0$. Il nous reste à évaluer $S_{int} = S_{>0} \setminus S_{good}$. Cette évaluation est

au coeur de la méthode et de ses extensions, aussi nous allons maintenant la détailler en démontrant sa correction.

Première observation Le vecteur $\boldsymbol{\pi}_{min}$ est solution de l'équation 8.6 où le vecteur $\mathbf{x}$ est l'inconnue.

$$\forall s \in S_{int}, \mathbf{x}(s) = Inf(\{ \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s')\mathbf{x}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')\}_{1 \leq i \leq k_s}) \quad (8.6)$$

Démonstration. Nous établissons l'égalité en démontrant l'inégalité dans les deux sens.
$(\geq)$ Soit St une stratégie pour le processus démarrant en s, alors
$\Pr^{St}(s \vDash \psi \mathcal{U} \chi) = \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St(s)}(s') \Pr^{St_{s'}}(s' \vDash \psi \mathcal{U} \chi) + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St(s)}(s')$
avec $St_{s'}$ la stratégie définie par $St_{s'}(s', \ldots, s_n) = St(s, s', \ldots, s_n)$.
Par conséquent,
$\Pr^{St}(s \vDash \psi \mathcal{U} \chi) \geq \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St(s)}(s') \boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St(s)}(s')$
$\geq Inf(\{\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St(s)}(s') \boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St(s)}(s')\}_{1 \leq i \leq k_s})$
Cette dernière inégalité étant vraie pour tout St, on obtient :
$\forall s \in S_{int}, \boldsymbol{\pi}_{min}(s) \geq Inf(\{\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') \boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')\}_{1 \leq i \leq k_s})$

$(\leq)$ Soit maintenant $\epsilon > 0$ alors, par définition de $\boldsymbol{\pi}_{min}$, pour tout s' il existe une stratégie $St_{s'}$ telle que $\Pr^{St_{s'}}(s' \vDash \psi \mathcal{U} \chi) \leq \boldsymbol{\pi}_{min}(s') + \epsilon$. Etant donné un état s, nous construisons une stratégie St pour le processus démarrant en s de la façon suivante. St choisit la distribution $\boldsymbol{\pi}_s^i$ qui minimise la quantité $\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') \Pr^{St_{s'}}(s \vDash \psi \mathcal{U} \chi) + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')$ puis applique au prochain état s' atteint la stratégie $St_{s'}$.
Par construction,
$\forall i, \Pr^{St}(s \vDash \psi \mathcal{U} \chi) \leq \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') \Pr^{St_{s'}}(s' \vDash \psi \mathcal{U} \chi) + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')$
$\leq \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') (\boldsymbol{\pi}_{min}(s') + \epsilon) + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')$
$\leq \epsilon + \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') \boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')$
Par conséquent, $\boldsymbol{\pi}_{min}(s) \leq \Pr^{St}(s \vDash \psi \mathcal{U} \chi)$
$\leq \epsilon + Inf(\{\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s') \boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')\}_{1 \leq i \leq k_s})$

Cette dernière inégalité étant vraie pour ϵ arbitrairement petit établit la deuxième inégalité et conclut la preuve. □

Deuxième observation Le vecteur $\boldsymbol{\pi}_{min}$ est *l'unique* solution de l'équation 8.6.

Démonstration. Afin d'établir l'unicité, nous étudions les stratégies *markoviennes*, c'est à dire les stratégies pour lesquelles la distribution choisie ne dépend que du dernier état du chemin d'exécution. Notons $St(s)$ la distribution choisie lorsque cet

état est s. Remarquons que le comportement du processus est alors celui d'une DTMC. L'équation vérifiée par une stratégie markovienne est :

$$\forall s \in S_{int}, \Pr^{St}(s \vDash \psi \mathcal{U} \chi) = \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St(s)}(s') \Pr^{St}(s' \vDash \psi \mathcal{U} \chi) + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St(s)}(s') \tag{8.7}$$

Afin de simplifier les notations, $\mathbf{x}(s)$ désignera $\Pr^{St}(s \vDash \psi \mathcal{U} \chi)$, $\mathbf{A}[s, s']$ désignera $\boldsymbol{\pi}_s^{St(s)}(s')$ et $\mathbf{b}(s)$ désignera $\sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St(s)}(s')$. L'équation 8.7 se réécrit alors sous la forme vectorielle $\mathbf{x} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b}$. La quantité $\mathbf{A}^n[s, s']$ est la probabilité d'être en s' partant de s après n pas sans quitter S_{int}. D'après la définition de S_{int}, pour toute stratégie la probabilité de rester indéfiniment dans S_{int} est nulle ce qui se traduit en termes de DTMC par la convergence de la série $\sum_{n \geq 0} \mathbf{A}^n$ (voir la section 8.2.2). En remplaçant (n fois) dans le terme droit de l'égalité, $\mathbf{x}$ par son expression on obtient $\mathbf{x} = \sum_{i \leq n} \mathbf{A}^i \mathbf{b} + \mathbf{A}^{n+1} \mathbf{x}$. Par passage à limite, $\mathbf{x} = \sum_{n \geq 0} \mathbf{A}^n \mathbf{b}$, ce qui signifie que l'équation 8.7 admet une solution unique.

Soit maintenant $\mathbf{x}$ une solution de l'équation 8.6. Notons St la stratégie markovienne qui consiste à choisir pour un état s, la distribution $\boldsymbol{\pi}_s^i$ pour laquelle le minimum est atteint avec la solution $\mathbf{x}$. Alors $\mathbf{x}$ vérifie l'équation 8.7 correspondant à cette stratégie. Par conséquent, $\mathbf{x}(s) = \Pr^{St}(s \vDash \psi \mathcal{U} \chi)$. Nous en déduisons que $\forall s, \mathbf{x}(s) \geq \boldsymbol{\pi}_{min}(s)$. Soit maintenant St' la stratégie markovienne conduisant à $\boldsymbol{\pi}_{min}$. Nous remarquons que :

$\forall s \in S_{int}$,
$\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St'(s)}(s')(\mathbf{x}(s') - \boldsymbol{\pi}_{min}(s')) =$
$(\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St'(s)}(s')\mathbf{x}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St'(s)}(s'))$
$-(\sum_{s' \in S_{int}} \boldsymbol{\pi}_s^{St'(s)}(s')\boldsymbol{\pi}_{min}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^{St'(s)}(s'))$
$\geq \mathbf{x}(s) - \boldsymbol{\pi}_{min}(s)$

Réécrit avec les notations vectorielles précédentes, ceci s'exprime par $\mathbf{A}(\mathbf{x} - \boldsymbol{\pi}_{min}) \geq \mathbf{x} - \boldsymbol{\pi}_{min} \geq \mathbf{0}$. En itérant, on obtient $\mathbf{A}^n(\mathbf{x} - \boldsymbol{\pi}_{min}) \geq \mathbf{x} - \boldsymbol{\pi}_{min} \geq \mathbf{0}$. Et finalement par passage à la limite, $\mathbf{x} - \boldsymbol{\pi}_{min} = \mathbf{0}$ ce qui établit cette observation. □

Troisième observation Le vecteur $\boldsymbol{\pi}_{min}$ est l'unique solution du programme linéaire :

Maximiser $\sum_{s \in S_{int}} \mathbf{x}(s)$ soumis aux contraintes

$$\forall s \in S_{int}, \forall 1 \leq i \leq k_s, \mathbf{x}(s) \leq \sum_{s' \in S_{int}} \boldsymbol{\pi}_s^i(s')\mathbf{x}(s') + \sum_{s' \in S_{good}} \boldsymbol{\pi}_s^i(s')$$

Démonstration. Soit $\mathbf{x}$ satisfaisant les contraintes de ce programme, alors $\mathbf{x}$ satisfait la version de l'équation 8.6 où les égalités sont remplacées par des inégalités larges. Supposons de plus que $\mathbf{x}$ soit une solution optimale et que l'une des inégalités de l'équation 8.6 soit stricte (disons pour $\mathbf{x}(s)$). Alors on peut remplacer $\mathbf{x}(s)$ par le terme

droit de l'inégalité en question et obtenir une meilleure solution. Par conséquent, une solution optimale vérifie l'équation 8.6 et en vertu de la deuxième observation $\boldsymbol{\pi}_{min}$ est l'unique solution de ce programme linéaire. □

L'évaluation consiste donc à résoudre ce programme linéaire.

Complexité Par construction, l'algorithme a une complexité linéaire en fonction de la taille de la formule. La complexité, fonction de la taille du processus, dépend des opérateurs. Pour les opérateurs non probabilistes, la description faite ci-dessus devrait convaincre le lecteur qu'elle est à nouveau linéaire. Pour les opérateurs probabilistes, l'étape la plus coûteuse est la résolution d'un programme linéaire qui se fait en temps polynomial à l'aide des méthodes intérieures [ROO 97].

8.5.4. *Panorama de la vérification des processus de décision markoviens*

Les processus de decision markoviens ont été introduits essentiellement pour modéliser et résoudre des problèmes d'optimisation [PUT 94]. L'un des travaux fondateurs ([COU 95]) relatifs à la vérification des PDM porte sur la satisfaction probabiliste du point de vue des logiques temporelles linéaires et établit des bornes de complexité exactes pour des formules exprimées soit en LTL (propositionnel) soit par un automate de Büchi.

L'approche présentée au paragraphe précédent a été étendue à différentes logiques arborescentes : pCTL* [BIA 95], une variante de CTL*, puis pTL et pTL* obtenus en introduisant des horloges à des fins d'observation [ALF 97]. Une autre extension concerne la sémantique des logiques. Ainsi, en reprenant notre exemple introductif, le modélisateur ne souhaite pas prendre en compte tous les ordonnanceurs. En effet, un ordonnanceur «réaliste» vérifie des propriétés d'équité (même faibles) vis à vis du choix de la transaction à exécuter. Or la méthode usuelle qui consiste à transformer la formule à vérifier pour prendre ces hypothèses n'est pas applicable ici. Il faut donc introduire dans les logiques des opérateurs équitables et concevoir des algorithmes appropriés [BAI 98, ALF 00]. Enfin, afin de prendre en compte l'imprécision des paramètres numériques du PDM, d'autres logiques ont été introduites et analysées [ALF 04].

Les méthodes d'évaluation de formules de logique temporelle pour les PDM peuvent aussi se combiner avec d'autres méthodes pour traiter des modèles plus expressifs. Ainsi, les automates temporisés probabilistes spécifient des systèmes à temps continu dont le caractère non déterministe recouvre à la fois le choix du prochain événement et l'instant de son occurrence. Lorsque ces automates ne sont pas probabilisés, la méthode usuelle consiste à construire une abstraction finie de leur système de transition temporisé (généralement infini) suffisante pour vérifier des formules de certaines logiques temporelles (comme par exemple TCTL). Cette abstraction prend différentes

formes : graphe des régions, graphe des zones, etc. Dans le cas probabiliste, l'abstraction est aussi possible mais le résultat est alors un PDM qui s'analyse par les méthodes vues précédemment [KWI 01, KWI 02b, KWI 02a]. D'autres modèles plus expressifs ne s'analysent que par des techniques d'approximation [KWI 00].

8.5.5. *L'outil PRISM*

PRISM (Probabilistic Symbolic Model Checker) est l'outil de référence en vérification de systèmes probabilisés développé à l'université de Birmingham en Grande Bretagne depuis le début des années 2000, disponible sous licence GPL. Il permet d'analyser des systèmes en temps discret (chaînes de Markov et processus de décision markoviens, logique PCTL) et en temps continu (chaînes de Markov, logique CSL). L'utilisation de PRISM suit la démarche classique : définition du modèle dans un formalisme spécifique, définition des propriétés en logique temporelle probabilisée puis calcul et analyse des résultats. Nous nous concentrons ici sur la présentation des fonctionnalités pour les systèmes à temps discret, celles pour le temps continu étant voisines des possibilités offertes par l'outil ETMCC vu plus haut.

• Langage des modèles de systèmes - PRISM utilise une description des systèmes sous forme de modules réactifs proche du modèle d'Alur et Henzinger [ALU 99] employé dans l'outil SMV de vérification CTL. Les modules correspondent aux entités actives du système et un état est défini par les valeurs des variables (globales) déclarées et manipulées par les modules. La synchronisation est réalisée par des gardes sur les valeurs des variables. Les paramètres stochastiques (probabilités de transitions) sont définies au sein des modules et l'utilisateur précise dans son modèle s'il s'agit d'une DTMC ou d'un PDM. Par extension au modèle de base, PRISM permet la définition de fonctions de récompense associées aux états et aux événements ; il sera ainsi possible d'obtenir des indices de performance moyens (gains ou pertes cumulés) à temps fini ou à l'équilibre.

• Langage des propriétés - Pour les DTMC et les PDM, PRISM permet de définir les propriétés à vérifier dans la logique PCTL présentée ci-dessus. Il calcule également avec quelle probabilité telle expression de PCTL est satisfaite par l'ensemble des états d'une DTMC sans nécessité de fixer un seuil de probabilité.

• Résultats calculés - Le logiciel analyse le modèle conformément aux propriétés demandées et renvoie les résultats qui sont présentés au sein de l'interface utilisateur sous forme d'ensemble de graphiques offrant une vue synthétique des propriétés du système. De plus, on obtient les paramètres moyens définis par les fonctions de récompense du modèle.

Du point de vue interne, l'implémentation est réalisée en langage C++ pour les parties numériques et en Java pour tout ce qui concerne l'interface utilisateur et la

gestion globale du logiciel. Une caractéristique importante de PRISM est la gestion des ensembles d'états du modèle à l'aide de structures de données dites symboliques, fondées sur les diagrammes de décision binaires multi-valués (MTBDD, [CLA 93]) ce qui lui permet de traiter des systèmes de taille importante (plus de 10^{10} états dans les cas favorables). Le vérificateur non probabilisé manipule systématiquement les structures BDD. Pour la résolution numérique, PRISM emploie trois «moteurs» fondés sur les MTBDD, les représentations creuses de matrices et un modèle mélangeant les deux premiers. Les expériences et nombreuses études de cas réalisées avec l'outil montrent en effet que la représentation MTBDD a tendance à ralentir fortement la résolution numérique par rapport aux méthodes (itératives), maintenant finement optimisées, à base de matrices creuses. Le lecteur intéressé par une étude détaillée est invité à consulter le site `www.cs.bham.ac.uk/~dxp/prism`.

8.6. Bibliographie

[Ajm 87] AJMONE MARSAN M., CHIOLA G., « On Petri nets with deterministic and exponentially distributed firing times », ROZENBERG G., Ed., *Advances in Petri Nets 1987*, n° 266LNCS, p. 132–145, Springer–Verlag, 1987.

[Ajm 95] AJMONE MARSAN M., BALBO G., CONTE G., DONATELLI S., FRANCESCHINIS G., *Modelling with Generalized Stochastic Petri Nets*, Wiley series in parallel computing, John Wiley & Sons, England, 1995.

[ALF 97] DE ALFARO L., « Model Checking of Probabilistic and Nondeterministic Systems », *STACS'97*, n° 1200LNCS, Springer-Verlag, p. 165-176, 1997.

[ALF 00] DE ALFARO L., « From Fairness to Chance », *Electronic Notes on Theoretical Computer Science*, vol. 22, 2000.

[ALF 04] DE ALFARO L., FAELLA M., HENZINGER T., MAJUMDAR R., STOELINGA M., « Model Checking Discounted Temporal Properties », *TACAS 2004 : 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, n° 2988LNCS, Springer Verlag, p. 77-92, 2004.

[ALU 99] ALUR R., HENZINGER T. A., « Reactive modules », *Formal methods in system design*, vol. 15, n°1, p. 7–48, 1999.

[AND 03] ANDOVA S., HERMANNS H., KATOEN J.-P., « Discrete-time rewards model-checked », *Formal Modelling and Analysis of Timed Systems (FORMATS 2003)*, n° 2791LNCS, Marseille, France, Springer Verlag, p. 88 - 103, 2003.

[AZI 96] AZIZ A., SANWAL K., V.SINGHAL, BRAYTON R., « Verifying continuous-time Markov chains », *8th Int. Conf. on Computer Aided Verification (CAV'96)*, n° 1102LNCS, New Brunswick, NJ, USA, Springer Verlag, p. 269–276, 1996.

[AZI 00] AZIZ A., SANWAL K., V.SINGHAL, BRAYTON R., « Model Chcking continuous-time Markov chains », *ACM Transactions on Computational Logic*, vol. 1, n°1, p. 162–170, 2000.

[BAI 98] BAIER C., KWIATKOWSKA M., « Model Checking for a Probabilistic Branching Time Logic with Fairness », *Distributed Computing*, vol. 11(3), p. 125-155, 1998.

[BAI 03a] BAIER C., HAVERKORT B., HERMANNS H., KATOEN J.-P., « Model-Checking Algorithms for Continuous Time Markov Chains », *IEEE Transactions on Software Engineering*, vol. 29, n°7, p. 524–541, juillet 2003.

[BAI 03b] BAIER C., HERMANNS H., KATOEN J.-P., WOLF V., « Comparative branching-time semantics for Markov chains », *Concurrency Theory (CONCUR 2003)*, n° 2761LNCS, Marseille, France, Springer Verlag, p. 492 - 507, 2003.

[BIA 95] BIANCO A., DE ALFARO L., « Model Checking of Probabilistic and Nondeterministic Systems », *FST TCS 95 : Foundations of Software Technology and Theoretical Computer Science*, n° 1026LNCS, Bangalore, India, Springer-Verlag, p. 499-513, 1995.

[CAM 94] CAMPOS J., COLOM J., JUNGNITZ H., SILVA M., « Approximate throughput computation of stochastic marked graphs », *IEEE Transactions on Software Engineering*, vol. 20, n°7, p. 526–535, juillet 1994.

[CHI 93a] CHIOLA G., ANGLANO C., CAMPOS J., COLOM J., SILVA M., « Operationnal analysis of timed Petri nets and application to computation of performance bounds », *Proc. of the 5th International Workshop on Petri Nets and Performance Models*, Toulouse, France, IEEE Computer Society Press, p. 128–137, octobre 19–22 1993.

[CHI 93b] CHIOLA G., DUTHEILLET C., FRANCESCHINIS G., HADDAD S., « Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications », *IEEE Transactions on Computers*, vol. 42, n°11, p. 1343–1360, novembre 1993.

[CLA 93] CLARKE E., FUJITA M., MCGEER P., MCMILLAN K., YANG J., ZHAO X., « Multi-terminal binary decision diagrams : An efficient data structure for matrix representation », *Proc. Int. Workshop on Logics Synthesis (IWLS'93)*, p. 1–15, 1993, also available in Formal Methods in System Design, 10(2/3) :149-169, 1997.

[COU 95] COURCOUBETIS C., YANNAKAKIS M., « The complexity of probabilistic verification », *Journal of the ACM*, vol. 42(4), p. 857–907, july 1995.

[COU 03] COUVREUR J.-M., SAHEB N., SUTRE G., « An optimal automata approach to LTL model checking of probabilistic systems », *In Proc. 10th Int. Conf. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2003)*, n° 2850LNAI, Almaty, Kazakhstan, Springer Verlag, p. 361-375, september 2003.

[DIA 01] DIAZ M., Ed., *Les réseaux de Petri : Modèles Fondamentaux*, Traité IC2 - Série Informatique et systèmes d'information, Edition Hermes, 2001.

[DON 98] DONATELLI S., HADDAD S., MOREAUX P., « Structured characterization of the Markov chains of phase-type SPN », *Proc. of the 10th International Conference on Computer Performance Evaluation. Modelling Techniques and Tools (TOOLS'98)*, n° 1469LNCS, Palma de Mallorca, Spain, Springer–Verlag, p. 243–254, septembre 14–18 1998.

[EME 80] EMERSON E. A., CLARKE E. M., « Characterizing Correctness Properties of Parallel Programs Using Fixpoints », *7^{th} International Colloquium on Automata, Languages and Programming, (ICALP)*, Noordweijkerhout, The Netherland, p. 169-181, 1980.

[FEL 68] FELLER W., *An introduction to probability theory and its applications. Volume I*, John Wiley & Sons, 1968, (third edition).

[FEL 71] FELLER W., *An introduction to probability theory and its applications. Volume II*, John Wiley & Sons, 1971, (second edition).

[FLO 85] FLORIN G., NATKIN S., « Les réseaux de Petri stochastiques », *TSI*, vol. 4, n°1, p. 143–160, 1985.

[FOA 98] FOATA D., FUCHS A., *Calcul des probabilités*, Dunod, 1998, Seconde édition.

[FOA 02] FOATA D., FUCHS A., *Processus stochastiques. Processus de Poisson, chaînes de Markov et martingales*, Dunod, 2002.

[GER 99] GERMAN R., TELEK M., « Formal relation of Markov renewal theory and supplementary variables in the analysis of stochastic Petri nets », BUCHHOLZ P., SILVA M., Eds., *Proc. of the 8th International Workshop on Petri Nets and Performance Models*, Zaragoza, Spain, IEEE Computer Society Press, p. 64–73, septembre 8–10 1999.

[HAD 05] HADDAD S., MOREAUX P., SERENO M., M.SILVA, « Product-form and stochastic Petri nets : a structural approach », *Performance Evaluation*, vol. 59/4, p. 313-336, 2005.

[HAV 95] HAVERKORT B., « Matrix-Geometric Solution of Infinite Stochastic Petri Nets », *Proc. of the International Performance and Dependability Symposium*, IEEE Computer Society Press, p. 72–81, 1995.

[HEN 90] HENDERSON W., PEARCE C., TAYLOR P., VAN DIJK N., « Closed queueing networks with batch services », *Queueing Systems*, , n°6, p. 59–70, 1990.

[HIL 96] HILLSTON J., *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.

[JEN 53] JENSEN A., « Markov chains as an aid in the study of Markov processes », *Skand. Aktuarietidskrift*, vol. 3, p. 87–91, 1953.

[KLE 75] KLEINROCK L., *Queueing systems. Volume I : Theory*, Wiley-Interscience, New-York, 1975.

[KLE 76] KLEINROCK L., *Queueing systems. Volume II :*, Wiley-Interscience, New-York, 1976.

[KWI 00] KWIATKOWSKA M., NORMAN G., SEGALA R., SPROSTON J., « Verifying Quantitative Properties of Continuous Probabilistic Timed Automata », *CONCUR 2000 - Concurrency Theory*, n° 1877LNCS, Springer Verlag, p. 123-137, August 2000.

[KWI 01] KWIATKOWSKA M., NORMAN G., SPROSTON J., « Symbolic computation of maximal probabilistic reachability », *Proc. 13th International Conference on Concurrency Theory (CONCUR'01)*, n° 2154LNCS, Springer Verlag, p. 169–183, August 2001.

[KWI 02a] KWIATKOWSKA M., NORMAN G., SEGALA R., SPROSTON J., « Automatic Verification of Real-time Systems with Discrete Probability Distributions », *Theoretical Computer Science*, vol. 282, p. 101-150, June 2002.

[KWI 02b] KWIATKOWSKA M., NORMAN G., SPROSTON J., « Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol », *Proc. PAPM/PROBMIV'02*, n° 2399LNCS, Springer Verlag, p. 169-187, July 2002.

[LAP 95] LAPRIE J., Ed., *Guide de la sûreté de fonctionnement*, Cépaduès - Éditions, Toulouse, France, 1995.

[LIN 98] LINDEMANN C., *Performance Modelling with Deterministic and Stochastic Petri Nets*, John Wiley & Sons, 1998.

[LIN 99] LINDEMANN C., REUYS A., THÜMMLER A., « DSPNexpress 2.000 Performance and dependability modeling environment », *Proc. of the* 29^{th} *Int. Symp. on Fault Tolerant Computing*, Madison, Wisconsin, juin 1999.

[MEY 80] MEYER J., « On evaluating the performability of degradable computing systems », *IEEE Transactions on Computers*, vol. 29, n°8, p. 720–731, août 1980.

[MOL 81] MOLLOY M. K., On the integration of delay and throughput in distributed processing models, PhD Dissertation, University of California, Los Angeles, CA, USA, septembre 1981.

[PUT 94] PUTERMAN M., *Markov decision processes : Discrete Stochastic Dynamic Programming*, John Wiley & Sons inc., 1994.

[ROO 97] ROOS C., TERLAKY T., VIAL J.-P., *Theory and Algorithms for Linear Optimization. An Interior Point Approach*, Wiley-Interscience, John Wiley & Sons Ltd, West Sussex, England, 1997.

[STE 94] STEWART W. J., *Introduction to the numerical solution of Markov chains*, Princeton University Press, USA, 1994.

[TRI 82] TRIVEDI K. S., *Probability & statistics with reliability, queueing, and computer science applications*, Prentice Hall, Englewood Cliffs, NJ, USA, 1982.

[TRI 92] TRIVEDI K. S., MUPPALA J. K., WOOLE S. P., HAVERKORT B. R., « Composite performance and dependability analysis », *Performance Evaluation*, vol. 14, n°3–4, p. 197–215, février 1992.

[VAR 85] VARDI M., « Automatic Verification of Probabilistic Concurrent Finite-State Programs », *FOCS 1985*, p. 327-338, 1985.

[YOU 05] YOUNES H., KWIATKOWSKA M., NORMAN G., PARKER D., « Numerical vs. Statistical Probabilistic Model Checking », *Int. Journal on Software Tools for Technology Transfer (STTT)*, 2005, To appear.