Synthèse de réseau de Petri simple à partir de traces d'exécution

Mathieu Poirier
ENS Rennes
Stagiaire
mathieu.poirier@ens-rennes.fr

Adrián Puerto Aubel
Inria Rennes
Encadrant

Éric Badouel
Inria Rennes
Encadrant

***Résumé*—L'opération de *découverte de processus* permet d'obtenir le modèle d'un système à partir de traces de l'exécution de ses processus. Les réseaux de Petri sont un bon choix de modèle pour cette opération, car en plus d'être un bon modèle pour de tels systèmes, la synthèse de réseau de Petri est un exercice bien étudié et permettant d'accomplir cette opération. Dans ce papier nous proposons une amélioration de la synthèse permettant d'obtenir un réseau de Petri plus simple que précédemment grâce à l'exploitation de résultats de la géométrie entière.**

***Mots-clefs*—Découverte de processus, base de Hilbert, cône des régions, réseau de Petri, synthèse, programmation linéaire**

Je remercie l'équipe SUMO pour son accueil chaleureux, l'équipe du séminaire Makushita pour m'avoir donné l'occasion de présenter mes travaux et pour leurs points de vue pratiques, Éric Badouel pour ses conseils avisés, et enfin Adrián Puerto Aubel pour la pertinence son encadrement et la qualité de nos nombreuses discussions.

I. Introduction

La production massive de données ces dernières décennies a contribué à ouvrir de nouveaux horizons pour la recherche en terme de *fouille de données* (en anglais : *data mining*). Nous nous intéressons ici à l'exploitation de données traçant l'exécution d'un processus : des *traces* (en anglais : *logs*).

En effet un certain nombre de domaines (industriels, administratifs, etc) se sont emparés des technologies de l'information et de la communication afin d'automatiser leurs processus. Par la même occasion, ils génèrent un nombre important de données portant sur leur propre déroulement. Cette source de données est particulière car elle ne porte pas sur des entités mais sur des événements. Son exploitation permet de découvrir, de superviser, et d'améliorer les processus, et elle est appelée *fouille de processus* (en anglais : *process mining*).

Parmi les possibilités offertes par cette source, nous nous intéressons tout particulièrement à la *découverte de processus*, i.e. l'élaboration du modèle théorique dont les exécutions correspondent aux processus réels d'un système, et ce à partir d'une spécification formée de leurs traces. Cela constitue une opération d'apprentissage.

Cette synthèse de modèle permet notamment d'identifier les points critiques d'un processus, voire ses failles. Il est également possible de suivre sur le modèle les processus futurs du système, et ainsi détecter des anomalies de fonctionnement. Ainsi l'élaboration du modèle constitue un outil de management précieux.

Le lecteur pourra trouver en [7] davantage d'idées quant à l'exploitation de ces techniques pour le management opérationnel.

Nous choisissons ici d'utiliser les réseaux de Petri comme modèle pour les systèmes. Une grande diversité de réseaux de Petri existent, comme le montrent [3] dans ses premiers chapitres et [4], ainsi ce que nous désignerons ici par réseau de Petri correspond plus précisément aux réseaux de Petri P/T.

La synthèse sous forme de réseau de Petri permet tout particulièrement d'exhiber les actions concurrentes du processus, i.e. les actions qui pourront être réalisées en parallèle. Cette information est tout à fait utile car elle permet alors de répartir des tâches.

Les travaux présentés dans le présent article s'appuient particulièrement sur le livre [1] qui couvre la théorie de la synthèse de réseaux de Petri. Ils prennent de plus la suite des travaux présents en [2], article dans lequel un algorithme incrémental de découverte de processus est présenté. L'approche incrémentale permet une utilisation continue de l'algorithme, puisque le modèle généré peut s'ajuster au fil de l'ajout de traces comme entrées.

Le présent article propose justement une amélioration de l'algorithme proposé en [2] en terme de qualité du résultat. L'accent est mis sur la possibilité d'obtenir un modèle plus concis du système étudié. On s'appuiera pour cela sur la notion géométrique des bases de Hilbert qui permet de redéfinir la notion clé de région minimale.

De plus, cette nouvelle approche sur la simplicité du modèle synthétisé offre de nouveaux horizons pour des variantes dans la découverte de processus, avec par exemple la détection d'anomalies dans les traces proposées en entrée.

Dans la suite de cet article, nous présentons en section II les réseaux de Petri au lecteur non familier avec ce modèle.

La section III présente ensuite les différentes étapes permettant la synthèse d'un réseau de Petri à partir de traces d'exécution d'un processus. Cela ne constitue pas entièrement un travail original mais majoritairement une reformulation qui se veut plus accessible de ce qui est exposé dans l'article [2]. Toutefois, la notion de région minimale présentée ici et utilisant les bases de Hilbert constitue une nouvelle approche du problème.

Enfin nous conclurons en section IV sur les ouvertures que permettent cette nouvelle approche de la simplicité du réseau synthétisé, obtenue grâce à l'utilisation des bases de Hilbert.

II. Réseaux de Petri

Les réseaux de Petri sont un outil de modélisation de systèmes proposé en 1962 par Carl Adam Petri [5].

Ce formalisme nous conviendra pour modéliser des systèmes, où des *transitions* correspondront chacune à une action du système, et des *places* indiqueront par leur *marquage* dynamique quelle transition peut s'activer. Les dépendances entre les différentes étapes du processus seront précisées par des *arcs* reliant dans un sens ou dans l'autre places et transitions.

A. Définitions

On appelle *réseau de Petri P/T* (pour la suite, simplement *réseau de Petri*) un quadruplet $RdP = (P, T, W, M_0)$ où :

- P est un ensemble fini, l'ensemble des *places* ;
- T est un ensemble fini, disjoint de P, l'ensemble des *transitions* ;
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ est la fonction de *pondération* du flot ;
- $M_0 : P \to \mathbb{N}$ est la fonction de *marquage initial*, elle donne le nombre de jetons occupant chaque place initialement.

On appelle *flot* du réseau de Petri l'ensemble :

$$\mathcal{F} = \{(x, y) \in (P \times T) \cup (T \times P) \mid W(x, y) > 0\}$$

dont les éléments sont appelés *arcs* du réseau. Cette relation $\mathcal{F}$ indique les paires places/transitions qui interagissent ensemble.

Pour bien distinguer le sens utilisé, nous utiliserons un opérateur infixe comme alias de W. Pour une place $p \in P$ et une transition $t \in T$, on notera $p^\bullet t = W(p, t)$ pour le poids du flot sortant de p et entrant dans t, et $t^\bullet p = W(t, p)$ pour le poids du flot sortant de t et entrant dans p. Enfin, on qualifiera d'*atomique* un réseau ne possédant qu'une seule place.

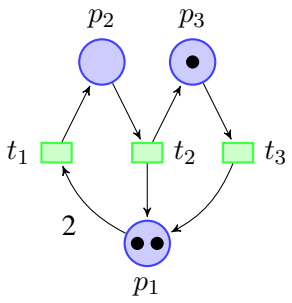

FIGURE 1. Exemple de réseau de Petri.

B. Représentations

La représentation graphique des réseaux de Petri s'appuie sur le fait que le graphe $(P \cup T, \mathcal{F})$ est un graphe biparti. Celle-ci permet de résumer visuellement les éléments d'un réseau de Petri comme en Figure 1. Les places sont représentées par des cercles, les transitions par des rectangles, les arcs par des flèches. Notons que le poids des arcs vaut 1 par défaut s'il n'est pas précisé visuellement. Les autres pondérations sont indiquées, et on peut lire le marquage initial en regardant le nombre de jetons dans chaque place, figurés par des disques noirs.

Numériquement, on représentera un réseau de Petri par une matrice à coefficients entiers positifs ayant $1 + 2n$ lignes et m colonnes, où $n = |T|$ et $m = |P|$ [1].

La matrice se découpe en trois blocs de taille $1 \times m$, $n \times m$ et $n \times m$ respectivement, le premier correspondant directement à M_0, et les suivants formant les matrices PRE et $POST$ [2], telles que pour une transition $t_i \in T$ et une place $p_j \in P$, on a :

$$PRE(i)(j) = W(t_i, p_j) = t_i{}^\bullet p_j,$$
le poids de l'arc liant t_i à p_j,
$$POST(i)(j) = W(p_j, t_i) = p_j{}^\bullet t_i,$$
le poids de l'arc liant p_j à t_i.

Les lignes de PRE et de $POST$ seront respectivement notées $t_i{}^\bullet$ et ${}^\bullet t_i$ pour $i \in [\![1, n]\!]$.

Les colonnes de PRE et de $POST$ seront respectivement notées ${}^\bullet p_j$ et $p_j{}^\bullet$ pour $j \in [\![1, m]\!]$.

En lisant la matrice par la perspective des colonnes, on y observe m blocs de taille $(1+2n) \times 1$ correspondant aux

1. On représente par $|E|$ la cardinalité d'un ensemble E.
2. Ces noms expriment le sens du flot du point de vue des places.

places du réseau. En effet on peut assimiler une place p au vecteur colonne de $\mathbb{N}^{1+2n}$: $p = (p[0], p[1], ..., p[2n])$, où $p[0] = M_0(p)$ et $\forall i \in [\![1, n]\!], p[i] = t_i{}^\bullet p \wedge p[n+i] = p^\bullet t_i$. On peut donc écrire : $p = (M_0(p); {}^\bullet p; p^\bullet)$ [3].

On trouve ainsi dans ce vecteur le marquage initial de la place ($M_0(p)$), puis, pour chaque transition, la pondération du flot entrant dans la place (${}^\bullet p$), et enfin la pondération du flot en sortant ($p^\bullet$).

Ainsi on obtient par exemple la matrice suivante pour le réseau présenté en Figure 1 :

$$\begin{pmatrix} p_1 & p_2 & p_3 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} M_0 \\ t_1{}^\bullet \\ t_2{}^\bullet \\ t_3{}^\bullet \\ {}^\bullet t_1 \\ {}^\bullet t_2 \\ {}^\bullet t_3 \end{pmatrix}$$

C. Dynamique/Sémantique

Le coeur de la dynamique d'un réseau de Petri réside dans l'activation de ses transitions. Informellement, une transition qui s'active consommera des jetons dans les places qui la précèdent et en générera dans celles qui la suivent, et ce selon la pondération des flèches liant ces places à cette transition. Une transition ne pourra donc s'activer que si les places la précédant contiennent un nombre suffisant de jetons pour cette opération, car le marquage des places n'autorise pas un nombre négatif de jetons.

La Figure 2 présente cette dynamique des transitions.

Le marquage du réseau fait office d'état global, et l'activation d'une transition permet d'atteindre un nouveau marquage, cette dynamique permet de construire le graphe des marquages, cette notion est présentée en annexe C car elle n'est pas utile ici.

Formellement, pour un marquage $M \in \mathbb{N}^P$, on dira qu'une transition $t \in T$ est *activable depuis* M, noté $M[t\rangle$, dès lors que :

$$\forall p \in P, M(p) \geq p^\bullet t$$

ou encore vectoriellement :

$$M \geq {}^\bullet t \tag{1}$$

Le marquage (étendu à $\mathbb{Z}$) $M' \in \mathbb{Z}^P$ obtenu après activation d'une transition $t \in T$ quelconque depuis un marquage étendu $M \in \mathbb{Z}^P$ est défini comme $M' = M \bullet t$ tel que :

$$\forall p \in P, M'(p) = M(p) - p^\bullet t + t^\bullet p$$

3. Par la notation $(u; v; w)$ nous désignons le vecteur obtenu par concaténation de vecteurs et/ou de scalaires u, v et w.

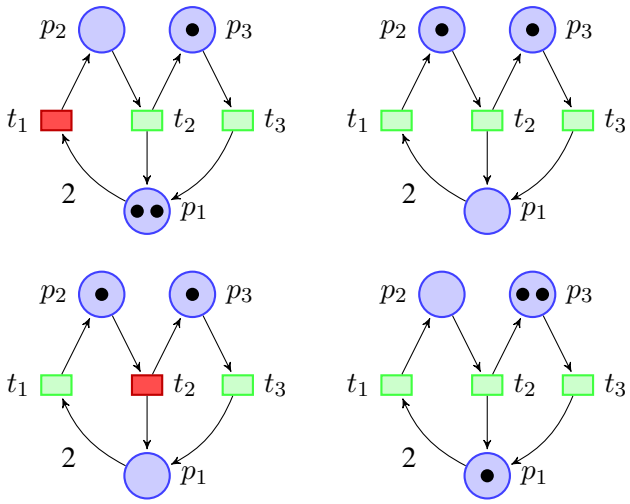

FIGURE 2. Exemples d'activations de transitions. $t1$ est activée puis $t2$. Sens de lecture usuel.

ou encore vectoriellement :

$$M' = M - {}^\bullet t + t^\bullet \tag{2}$$

On adoptera la notation $M[t\rangle M'$ pour signifier $M[t\rangle$ et $M' = M \bullet t$. Auquel cas si M est un marquage valide, i.e. $M \in \mathbb{N}^P$, alors M' est un marquage valide également.

D. Langage d'un réseau P/T

Après l'activation d'une transition, de nouvelles transitions peuvent être activées depuis le marquage obtenu. Par exemple la Figure 2 montre pour le réseau de Petri qu'elle présente qu'après l'activation de t_1, la transition t_2 devient activable. La transition t_3 était activable depuis le début, l'était après t_1 et l'est encore après t_2. Nous allons assez rapidement considérer des *séquences de transitions* appliquées à notre système. Pour cela on utilise le langage T^* comme ensemble des séquences de transitions.

On étend alors naturellement les prédicats $M[\omega\rangle$ et $M[\omega\rangle M'$, ainsi que l'opérateur $M \bullet \omega$ à toute séquence de transitions $\omega \in T^*$. On aura $M[\omega\rangle$ lorsqu'il est possible, depuis le marquage M, d'activer séquentiellement les transitions qui constituent le mot ω, et $M \bullet \omega$ dénotera le marquage ainsi obtenu. L'annexe B détaille formellement les définitions associées.

Le langage accepté par un réseau de Petri donné $RdP = (P, T, W, M_0)$ est alors défini par :

$$\mathscr{L}(RdP) = \{\omega \in T^* \mid M_0[\omega\rangle\}.$$

Remarquons que par la définition de la relation $M[\omega\rangle$, ce langage est clos par préfixe, i.e. on a :

$$\forall \omega \in \mathscr{L}(RdP), Pref(\omega) \subseteq \mathscr{L}(Rdp) \tag{3}$$

où $Pref(\omega)$ désigne l'ensemble des préfixes de ω.

E. Vecteurs de Parikh

Nous définissons ici la notion de vecteur de Parikh d'un mot, car cette notion est au coeur de notre découverte de processus, ainsi que de sa capacité d'apprentissage.

Informellement, il s'agit de compter dans un mot le nombre d'occurrences de chaque lettre, et ce indépendamment de leur position. La motivation derrière l'utilisation de ces objets est la suivante : si deux mots activables ont le même vecteur de Parikh, alors ils auront le même effet sur le réseau de Petri. En effet lorsque plusieurs transitions sont activées séquentiellement, leur effet global est une somme de jetons ajoutés et retirés de chaque place. Ainsi l'ordre dans lequel se présente une séquence de transitions n'a d'incidence que sur son activabilité et pas sur la modification du marquage qu'elle produira.

Formellement, à partir d'un alphabet $T = \{t_1, ..., t_n\}$, on définit coordonnée par coordonnée le vecteur de Parikh d'un mot $\omega \in T^*$, noté $\Psi(\omega)$, par :

$$\forall i \in [\![1, n]\!], \Psi(\omega)[i] = |\omega|_{t_i}$$

où $|\cdot|_{t_i} : T^* \to \mathbb{N}$ désigne la fonction qui compte le nombre d'occurrences de la lettre t_i dans son argument.

Ainsi par exemple sur $T = \{a, e, g, i, m, n, r, t\}$ on a : $\Psi(imaginer) = \Psi(migraine) = (1, 1, 1, 2, 1, 1, 1, 0)$, et $\Psi(gagnante) = (2, 1, 2, 0, 0, 2, 0, 1)$.

Le vecteur de Parikh d'un mot $\omega \in T^*$ permet d'exprimer plus explicitement ce que ce mot, en tant que séquence de transitions, va changer pour le réseau de Petri.

En effet pour $\omega \in T^*$ et $p \in P$, on a :

$$(M_0 \bullet \omega)(p) = M_0(p) + \Psi(\omega) \cdot ({}^\bullet p - p^\bullet). \tag{4}$$

Cette équation montre que le calcul de $M \bullet \omega$ ne dépend du point de vue de ω que de $\Psi(\omega)$. Cela a pour conséquence immédiate que deux mots ω et ω' ayant le même vecteur de Parikh mènent d'un même marquage M au même autre après activation [4] :

$$\Psi(\omega) = \Psi(\omega') \implies M \bullet \omega = M \bullet \omega' \tag{5}$$

Cette dernière propriété a elle-même pour conséquence une propriété sur les langages de réseaux de Petri, que nous appellerons *clôture par Parikh* et qui s'écrit pour un langage L comme suit :

$$\forall \omega, \omega' \in L, (\Psi(\omega) = \Psi(\omega') \implies \forall t \in T, (\omega \cdot t \in L \iff \omega' \cdot t \in L)) \tag{6}$$

Les langages de réseaux de Petri disposent bien de cette propriété car il s'agit dans leur cas d'une conséquence directe de (5) avec $M = M_0$, et le fait que l'activabilité d'une transition ne dépend que du marquage actuel.

4. Pour autant leur activabilité n'est pas liée : dans la figure 1 la séquence $t_1 t_2$ est activable mais pas $t_2 t_1$ bien que ces deux mots aient le même vecteur de Parikh.

On peut d'ailleurs traduire cette propriété autrement pour un réseau de Petri : le réseau n'a pour mémoire pas plus que le vecteur de Parikh de la séquence de transitions activées depuis son initialisation.

III. Synthèse de réseau à partir de traces

Dans cette section, nous commençons par donner la spécification de la synthèse de réseau à partir de traces, puis nous présentons étape par étape la méthode de cette synthèse.

A. Spécification

Commençons tout d'abord par spécifier nos attentes vis-à-vis de la synthèse.

On suppose dès à présent que le système à modéliser fonctionne à partir d'un ensemble $T = \{t_1, ..., t_n\}$ d'actions, qui constituera l'ensemble des transitions des réseaux de synthèse.

Pour la synthèse, nous nous donnons comme entrée des traces d'exécution de processus, c'est-à-dire un ensemble fini $Logs$ de mots de T^*, constituant un échantillon de ce que le système réel est capable d'accepter. On peut imaginer que ces mots correspondent au traitement de différents cas par le système, et que la lecture d'un journal d'événements enregistrant toutes les actions effectuées a permis de produire ces traces.

Notre contrainte principale sera de pouvoir rejouer ces traces avec le système synthétisé. En effet, puisque ce sont des mots enregistrés dans les traces, ce sont des séquences d'actions valides pour notre système. Nous appelons *rejouabilité* ce critère incontournable.

Dans l'idéal le système synthétisé doit faire preuve de généralisation vis-à-vis des traces mais selon une juste mesure. Pour préciser ces termes, présentons les deux extrêmes, dont aucun n'est souhaité :

— si le système synthétisé n'accepte que les traces présentées, la synthèse est trop précise. Nous attendons de la synthèse une généralisation et donc son résultat devrait accepter d'autres exécutions que celles tracées car par construction notre échantillon n'est pas exhaustif, cette généralisation constitue une forme d'apprentissage ;

— si le système synthétisé accepte T^* entièrement, la synthèse est trop générale. Nous attendons de la synthèse de tenir compte des particularités du système réel à modéliser car nous souhaitons par la suite exploiter son modèle.

Un équilibre est donc souhaité entre un sur-apprentissage et un sous-apprentissage du fonctionnement sous-jacent du système réel. Les critères correspondant

sont ceux de la *précision* et de la *généralisation*, dont les tendances sont a priori en contradiction.

Le dernier critère que nous prendrons en compte est celui de la *simplicité*. Il s'agit de préférer, à langage équivalent, un modèle plus simple. Nous plaçons ce critère en dernier car donner davantage de priorité à ce critère conduirait à un procédé de synthèse tout à fait différent. Nous préférons ici donner priorité à la rejouabilité et à la précision.

Cette notion sous-entend qu'il est possible de quantifier la simplicité du modèle généré. Nous choisirons de la mesurer comme la somme des coefficients apparaissant dans le réseau de Petri : poids des arcs et valeurs du marquage initial. Plus cette valeur sera petite, plus le réseau sera considéré simple.

B. Remarques préliminaires

Avant de détailler la synthèse étape par étape, nous avons trois remarques à faire.

Premièrement, puisque nous travaillons avec les réseaux de Petri, qui génèrent des langages clos par préfixe, nous pouvons d'emblée travailler avec $\overline{Logs}$, la clôture par préfixe de $Logs$.

Deuxièmement, nous n'aurons pas à nous soucier du critère de généralisation. Notre synthèse conduira à un réseau dont le langage est une sur-approximation de $Logs$ car nous respecterons la rejouabilité. De plus, encore une fois, travailler avec les réseaux de Petri contraint la classe de langage à laquelle nous avons accès.

Ainsi prendre même le plus petit langage de réseau de Petri contenant nos traces constituera une généralisation car ce langage aura nécessairement des propriétés comme la clôture par préfixe (3) et la clôture par Parikh (6), ce qui agrandit le langage.

Par exemple, sur $T = \{a, b, c, d\}$ et avec les traces $Logs = \{adb, dac\}$, on obtient au moins : $\{\epsilon, a, d, ad, da, adb, adc, dab, dac\}$. Les mots adc et dab illustrent la généralisation induite par la clôture par Parikh.

Troisièmement, à propos des places, remarquons qu'une place constitue une restriction sur le langage produit par le réseau. Il est possible de s'en convaincre en constatant d'une part qu'un réseau où $P = \emptyset$ a pour langage T^* tout entier, et que d'autre part, ajouter une place à un réseau ne permet jamais d'augmenter le langage accepté mais seulement de le réduire.

En effet l'activabilité d'une transition t depuis un marquage M, que nous notons $M[t\rangle$, est vérifiée lorsque la formule en (1) est vérifiée, et celle-ci s'exprime par un quantificateur universel. Ainsi pour un P donné, si cette condition était fausse elle le restera avec un ensemble de places plus grand $P' \supset P$, tandis que si elle était vraie elle peut devenir fausse.

Plus exactement encore, le langage d'un réseau RdP peut se décrire comme l'intersection des langages acceptés par chacun de ses réseaux atomiques, i.e. ne possédant qu'une seule de ses places :

$$\mathscr{L}(RdP) = \bigcap_{p \in P} \mathscr{L}(RdP(p)) \tag{7}$$

où P est l'ensemble des places et $RdP(p)$ est le réseau de Petri atomique n'ayant que la seule place p.

C. Définition des régions

La synthèse du réseau de Petri va consister essentiellement à trouver les places du réseau encodées tel que présenté en section II-B. Nous travaillons donc a priori dans l'espace $\mathbb{N}^{1+2n}$ des places. La contrainte de rejouabilité va nous demander de restreindre notre recherche aux vecteurs entiers d'un cône convexe polyédrique rationnel saillant [5] de l'espace $\mathbb{Q}^{1+2n}$. C'est ce qui définira le cône des régions [6] comme nous allons le voir dans cette section.

Puisque la rejouabilité des traces est le critère principal de notre synthèse, il convient d'après la dernière remarque préliminaire et par (7) que dans le réseau de Petri généré, toute place p inclue le langage $\overline{Logs}$ dans le langage qu'elle produit $\mathscr{L}(RdP(p))$.

Nous définissons alors l'ensemble des régions comme étant :

$$R(Logs) = \{p \in \mathbb{N}^{1+2n} \mid \overline{Logs} \subseteq \mathscr{L}(RdP(p))\}. \tag{8}$$

L'ensemble des régions est l'ensemble de places possibles auxquelles nous nous restreignons afin de satisfaire la contrainte de rejouabilité. La définition ensembliste précédente étant peu pratique, nous allons l'expliciter davantage, et même en tirer une approche géométrique.

Considérons les équivalences suivantes sur un réseau de Petri atomique constitué de la place p :

$$\begin{aligned} \overline{Logs} \subseteq \mathscr{L}(RdP(p)) &\iff \forall \omega \in \overline{Logs}, M_0[\omega\rangle \\ &\iff \forall \omega \cdot t \in \overline{Logs}, (M_0 \bullet \omega)[t\rangle \end{aligned}$$

La dernière équivalence est vérifiée dans notre cas parce que nous exploitons la clôture par préfixe de $\overline{Logs}$. En effet puisque chaque mot dispose de ses préfixes dans le langage, il ne suffit pour lui que de vérifier s'il est possible d'activer sa dernière lettre.

5. Un cône convexe est un ensemble de vecteurs stable par combinaison linéaire positive. On peut le qualifier de polyédrique s'il est généré par un ensemble fini de vecteurs, de rationnel si les vecteurs générateurs sont rationnels, et de saillant s'il ne contient que des demi-droites vectorielles. Ces notions sont issues de [6] et développées plus en détail dans ce livre.

6. Le nom de *région* tient son origine dans la synthèse de réseau de Petri d'un autre type que les réseaux P/T. Il n'y a donc pas d'interprétation à y donner dans notre contexte.

Nous pouvons maintenant traduire algébriquement la dernière formule :

$$\begin{aligned}
&\forall \omega \cdot t \in \overline{Logs}, (M_0 \bullet \omega)[t\rangle \\
&\quad \iff \forall \omega \cdot t \in \overline{Logs}, M_0(p) + \Psi(\omega)({}^\bullet p - p^\bullet) \geq p^\bullet t \\
&\quad \iff \forall \omega \cdot t \in \overline{Logs}, p \cdot \langle \Psi(\omega), t\rangle \geq 0 \qquad (9)
\end{aligned}$$

où $\langle v, t\rangle = (1; v; -v - u_t)$ [7].

La deuxième équivalence est la traduction de l'expression précédente par les formules (1) et (4). La dernière formule équivalente, avec le produit scalaire positif, est celle qui nous intéresse car elle traduit géométriquement la contrainte de rejouabilité d'un mot $\omega \cdot t \in \overline{Logs}$. En effet le vecteur $\langle \Psi(\omega), t\rangle$ définit un hyperplan dont il est un vecteur normal dans l'espace contenant les places $\mathbb{Q}^{1+2n}$. Par ailleurs l'inégalité traduit le fait que pour respecter la rejouabilité du mot $\omega \cdot t$ la place p doit en tant que vecteur se situer au dessus de cet hyperplan, i.e. dans le demi-espace qui contient le vecteur normal.

Ainsi chaque mot $\omega \cdot t \in \overline{Logs}$ va donner un vecteur normal et donc un demi-espace dans lequel devra se trouver toute place p de notre réseau synthétisé afin de valider la contrainte de rejouabilité. On pose ainsi l'ensemble des contraintes :

$$Logs^C = \left\{\langle v, t\rangle \in \mathbb{N}^{1+2n} \mid \exists \omega \cdot t \in \overline{Logs}, \Psi(\omega) = v\right\} \qquad (10)$$

qui nous permet finalement de reformuler la définition de l'ensemble des régions en (8) par :

$$R(Logs) = \left\{p \in \mathbb{N}^{1+2n} \mid \forall c \in Logs^C, c \cdot p \geq 0\right\}. \qquad (11)$$

La définition donnée par l'expression en (11) correspond bien aux vecteurs entiers d'un cône convexe polyédrique rationnel saillant. Nous développons en annexe D la preuve de ces propriétés. Il nous importe ici seulement de les avoir car nous pourrons exploiter les résultats existant sur ces cônes.

D. *Problèmes de séparation et ensemble de régions complet*

Grâce à la restriction au cône des régions comme places possibles, nous garantissons la rejouabilité du résultat de notre synthèse. Le choix des réseaux de Petri comme outil de modélisation nous suffit à la généralisation du modèle. Le critère suivant que nous avons à respecter est donc celui de la précision.

Nous avons déjà fait remarquer qu'une place dans un réseau de Petri constitue une contrainte sur son langage. Nous allons jouer de cette propriété afin d'affiner notre langage au plus près des traces d'exécutions à partir desquelles nous effectuons la synthèse.

7. u_t désigne le vecteur de la base canonique de $\mathbb{N}^n$ ayant un 1 à la composante associée à t. L'ambiguïté sur les coordonnées est levée par l'ordre induit par les indices sur $T = \{t_1, ..., t_n\}$. Pour rappel ';' permet d'obtenir le vecteur concaténé.

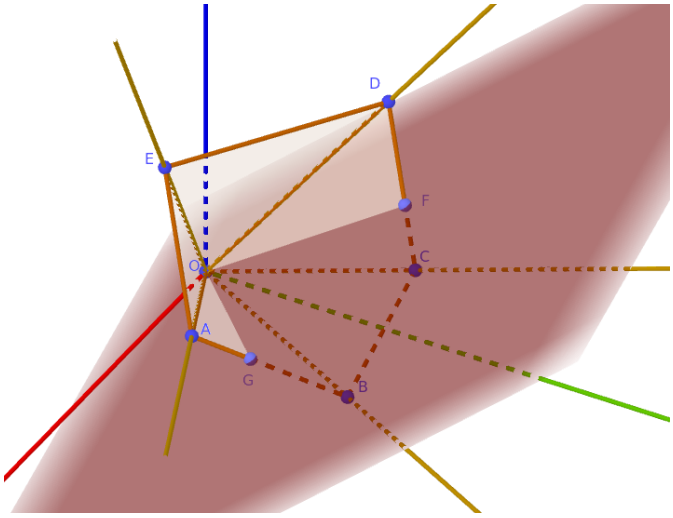

FIGURE 3. Les plans Oll' pour l et l' consécutifs dans $ABCDEA$ définissent le cône des régions. Le plan OGF définit un problème de séparation résoluble : il est possible de se trouver dans le cône et sous ce plan.

Informellement, nous allons tenter d'empêcher notre modèle d'accepter des mots proches des traces mais absents dans ces dernières. Plus exactement, nous allons tenter de vérifier :

$$\forall \omega \in \overline{Logs}, \forall t \in T, (\omega \cdot t \notin \overline{Logs} \implies \omega \cdot t \notin \mathscr{L}(RdP))$$

où RdP est le réseau synthétisé.

C'est ici qu'entre en jeu la notion de problème de séparation. Considérons les équivalences suivantes, analogues à celles qui ont abouties à (9) :

$$\begin{aligned}
\omega \cdot t \notin \mathscr{L}(RdP) &\iff \exists p \in RdP, \omega \cdot t \notin \mathscr{L}(RdP(p)) \\
&\iff \exists p \in RdP, \neg(M_0^{RdP(p)} \bullet \omega)[t\rangle \\
&\iff \exists p \in RdP, p \cdot \langle \Psi(\omega), t\rangle < 0
\end{aligned}$$

On pose ainsi l'ensemble des problèmes de séparation :

$$\begin{aligned}
Logs^{SP} = \{&\langle v, t\rangle \in \mathbb{N}^{1+2n} \mid \\
&\exists \omega \in \overline{Logs}, (\Psi(\omega) = v \wedge \omega \cdot t \notin \overline{Logs})\}
\end{aligned}$$

Encore une fois, ces vecteurs définissent des hyperplans dont ils sont le vecteur normal. Néanmoins cette fois-ci, d'une part il ne faut pas être au-dessus mais strictement en-dessous, et d'autre part nous n'avons pas besoin d'y placer toutes nos places mais seulement au moins une.

On dira d'une place $p \in \mathbb{N}^{1+2n}$ qu'elle résout un problème de séparation $sp \in Logs^{SP}$ lorsque : $p \cdot sp < 0$. Nous pourrons pour une région $r \in R(Logs)$ désigner l'ensemble des problèmes de séparation qu'elle résout par $sp(r) \subseteq Logs^{SP}$. La Figure 3 montre comment visualiser un problème de séparation résoluble.

Toutefois pour un problème de séparation, il peut ne pas exister de région le résolvant. C'est le cas pour l'un d'entre eux si le demi-espace qu'il définit est disjoint de notre cône des régions.

Nous noterons $Logs^{SSP} \subseteq Logs^{SP}$ l'ensemble des problèmes de séparation résolubles [8].

On appelle ensemble *complet* de régions tout ensemble de régions dans lequel pour chaque problème de séparation résoluble, il existe un vecteur résolvant ce problème. C'est-à-dire un ensemble $S \subseteq R(Logs)$ vérifiant :

$$\bigcup_{r \in S} sp(r) = Logs^{SSP}. \tag{12}$$

Notre synthèse sera donc précise dès lors que notre réseau sera constitué de places correspondant à un ensemble de régions complet.

E. Régions minimales

L'ensemble de régions complet que nous avons à trouver se doit d'être fini, car notre modèle doit disposer d'une représentation finie. La notion de région minimale vient assurer qu'il est possible de trouver un ensemble de places fini et complet. De plus, elle assure d'en trouver un de simplicité maximale, remplissant ainsi notre dernier critère.

Puisque nous en venons à la simplicité, rappelons sa mesure. La fonction à minimiser et qui est souvent retenue dans la communauté des réseaux de Petri pour mesurer la simplicité d'un réseau est la fonction $cost$ suivante :

$$cost(RdP) = \sum_{p \in P} \sum_{i=1}^{2n} p[i] \tag{13}$$

où P est l'ensemble des places du réseau RdP mesuré.

Bien que conventionnel, ce choix est particulier. Nous présentons en annexe E les propriétés strictement nécessaires à une fonction $cost$ alternative.

Considérons désormais une région $p = q+r \in R(Logs)$ qui s'exprime comme la somme vectorielle de deux autres régions q et r. Nous allons montrer qu'il est plus intéressant d'ajouter $\{q, r\}$ aux places du réseau plutôt que $\{p\}$.

En terme de simplicité, la propriété de linéarité de $cost$ donne immédiatement que le réseau sera plus simple (au sens large) en y ajoutant l'ensemble $\{q, r\}$ plutôt que $\{p\}$.

Toutefois cette alternative n'est vraiment meilleure que si le premier ensemble est capable d'assurer la précision qu'assurait p. Or c'est le cas car on a :

$$sp(p) \subseteq sp(q) \cup sp(r)$$

comme prouvé en annexe F.

Autrement dit, q et r assurent à eux deux au moins autant de précision que p tout seul. Et ce avec une simplicité égale ou supérieure. On en conclut qu'il n'est pas intéressant de travailler avec des régions pouvant s'exprimer comme la somme de deux autres.

D'après le résultat précédent, à précision égale, les régions les plus simples se trouvent nécessairement dans l'ensemble des régions ne pouvant pas s'exprimer comme la somme de deux autres, c'est donc ainsi que nous définissons l'ensemble des régions minimales $R_{min}(Logs)$:

$$R_{min}(Logs) = \{p \in R(Logs) \mid \nexists q, r \in R(Logs) \setminus \{p\}, p = q + r\}. \tag{14}$$

Nous avons vu dans la partie III-C que l'ensemble $R(Logs)$ est constitué des vecteurs entiers d'un cône convexe polyédrique rationnel saillant de $\mathbb{Q}^{1+2n}$. Or, la définition (14) correspond exactement à celle de la base de Hilbert de ce cône [6, p.233]. Nous bénéficions donc de toutes les propriétés de cette base. En particulier, nous la savons finie, et génératrice des vecteurs entiers du cône (i.e. de $R(Logs)$) par combinaisons linéaires positives entières. Ces résultats sont tous issus de [6].

Cet ensemble génère le cône des régions, nous en déduisons qu'il est complet, comme prouvé en annexe G.

Finalement, l'ensemble $R_{min}(Logs)$ constitue donc un ensemble fini et complet de régions. Il nous assure la précision de notre modèle ainsi que sa finitude, tout en garantissant d'y trouver les places les plus simples pour ce niveau de précision.

F. Extraction d'un réseau de Petri

L'ensemble des régions minimales est un ensemble fini et complet de régions, toutefois il peut présenter de la redondance. Ainsi un sous-ensemble des régions minimales pourrait rester complet tout en formant une solution strictement plus simple.

Il convient donc de rechercher un sous-ensemble complet des régions minimales de simplicité maximale.

En prenant compte de la mesure proposée en partie III-E, nous cherchons donc à résoudre le problème de couverture par ensemble pondéré, i.e. nous cherchons un ensemble solution $S \subseteq R_{min}(Logs)$ tel que :

$$\bigcup_{r \in S} sp(r) = Logs^{SSP}$$
$$\text{avec } \sum_{r \in S} cost(r) \text{ minimal}$$

c'est-à-dire que la solution soit complète, et maximisant la simplicité du modèle synthétisé.

Ce problème bien connu se résout comme un problème d'optimisation linéaire en nombres entiers. La solution S correspond finalement à l'ensemble des places que nous prenons pour constituer notre réseau de synthèse.

8. SSP pour Solvable Separation Problem.

Ainsi le critère de complexité est rempli par la résolution de ce problème d'optimisation. De plus nous avons la souplesse du choix dans la mesure de la complexité du modèle par le choix de la fonction $cost$.

IV. Conclusion

L'approche des régions minimales utilisant la base de Hilbert du cône des régions permet d'assurer l'obtention d'un réseau de simplicité maximale avec les contraintes que nous nous sommes données. Nous obtenons le réseau le plus simple possible pour une mesure linéaire de la taille des places, et avec la rejouabilité des traces assurées et la précision la plus grande possible une fois la rejouabilité assurée.

Comme mentionné dans [2], il est possible de céder en précision pour gagner en simplicité. En effet en s'affranchissant de devoir résoudre tous les problèmes de séparation, il est possible d'obtenir un réseau plus simple encore. Toutefois il n'est pas possible d'optimiser les deux critères en même temps.

De la même manière, il serait possible de céder en rejouabilité, c'est-à-dire de s'affranchir de respecter la totalité des contraintes imposées par l'ensemble des traces. En effet les demi-espaces générés par les traces définissent un premier cône si on choisit de tous les respecter, mais ils en définissent bien d'autres si nous choisissons de n'en respecter qu'une partie. De plus, dans ces nouveaux cônes, l'ensemble des problèmes de séparation résolubles peut se trouver être plus grand, et le réseau le plus simple obtenu dans ce cône peut être plus petit. Ainsi il serait possible de gagner en précision et en simplicité lorsque la contrainte de rejouabilité est assouplie.

En effet la rejouabilité est un critère incontournable tant que les traces sont fiables. Néanmoins si la fiabilité des traces ne peut être assurée, alors il est au contraire très intéressant de savoir remettre en question cette rejouabilité. Plus exactement, il est raisonnable de penser que dans le cas où un réseau bien plus simple et bien plus précis est obtenu en abandonnant une trace, alors cette trace se trouvait être une anomalie pour le système.

De futurs travaux pourraient étudier plus en profondeur les opportunités en terme de détection d'anomalies ouvertes par cette nouvelle approche de la simplicité des réseaux de Petri synthétisés à partir de traces d'exécution.

Références

[1] Eric Badouel, Luca Bernardinello, and Philippe Darondeau. *Petri Net Synthesis*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015.

[2] Eric Badouel and Uli Schlachter. Incremental process discovery using petri net synthesis. *Fundam. Inform.*, 154(1-4) :1–13, 2017.

[3] René David and Hassane Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005.

[4] Tadao Murata. Petri nets : Properties, analysis and application. In *Proceedings of the IEEE*, volume 77, pages 541–580, Apr 1989.

[5] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.

[6] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Series in Discrete Mathematics & Optimization. Wiley, 1998.

[7] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

ANNEXE

A. Déroulement et activités du stage

Ce document constituant avant tout un rapport de stage, quelques mots sur son déroulement suivent ci-après.

Ce stage de fin de L3 s'est déroulé dans les bâtiments de l'Inria/Irisa à Rennes. J'ai été intégré à l'équipe SUMO pour la durée de neuf semaines, encadré par Éric Badouel, et plus quotidiennement par Adrián Puerto Aubel.

Une première partie du stage s'est constituée d'études. Il me fallait notamment me familiariser avec le modèle des réseaux de Petri, que j'ai pu mieux appréhender à l'aide de [4], des premiers chapitres de [3] et de documents de travaux pratiques fournis par Adrián. Cette familiarisation m'a permis d'étudier et de comprendre l'article [2] à la base de ce stage, la lecture parallèle de [1] m'a aidé à la compréhension de l'article.

Par la suite, il m'a été demandé de chercher de possibles extensions aux travaux détaillés dans l'article. Après diverses avancées sur la notion de synthèse de réseau à partir de langage flou, je me suis focalisé sur la notion de simplicité du réseau, pour finalement produire le résultat énoncé dans ce document.

J'ai pu au cours de ce stage prendre part aux activités de l'équipe, notamment les divers séminaires d'équipe ou à thème. L'occasion me fut donnée de présenter mon travail aux membres de l'équipe au cours d'un séminaire dédié aux stagiaires. J'ai aussi pu faire une présentation dans le cadre du séminaire Makushita, séminaire des non-permanents des équipes SUMO et LogicA. L'ensemble de ces activités m'ont assuré une bonne intégration dans l'équipe.

B. Extension des prédicats et opérateur sur les transitions

Comme indiqué en section II-D, les prédicats $M[t\rangle$, $M[t\rangle M'$ et l'opérateur $M \bullet t$ pour des marquages M et M' et une transition t s'étendent à toute séquence de transitions $\omega \in T^*$.

Pour cela on définit pour tous marquages $M, M' \in \mathbb{N}^P$, toute transition $t \in T$ et tout mot $\omega \in T^*$:

- $M \bullet \epsilon = M$;
- $M \bullet (\omega \cdot t) = (M \bullet \omega) \bullet t$;
- $M[\epsilon\rangle$ comme toujours valide ;
- $M[\omega \cdot t\rangle$, ssi $M[\omega\rangle \wedge (M \bullet \omega)[t\rangle$;
- $M[\omega\rangle M'$, ssi $M[\omega\rangle \wedge M' = M \bullet \omega$

où ϵ désigne le mot vide dans T^*.

C. Graphe des marquages

Cette annexe reprend les notions présentées en section II-C et II-D afin de développer celle du graphe des marquages.

Un *état global* d'un réseau de Petri $RdP = (P, T, W, M_0)$ est un marquage de ses places $M \in \mathbb{N}^P$. L'activation d'une transition permet de plus de passer d'un état à un autre. Ainsi on peut finalement décrire le *graphe des marquages* du réseau RdP par un automate initialisé (éventuellement infini) $G(RdP) = (S, T, \Delta, M_0)$, où :

- $S \subseteq \mathbb{N}^P$ est l'ensemble des états accessibles depuis M_0, i.e. on a $M \in S \iff \exists \omega \in T^*, M_0[\omega\rangle M$;
- T est l'ensemble des étiquettes des liaisons ;
- $\Delta : S \times T \to S$ est l'ensemble des liaisons étiquetées entre états, $\Delta(M, t)$ a une image ssi $M[t\rangle$ et alors $\Delta(M, t) = M \bullet t$;
- $M_0 \in S$ est l'état initial ;
- la notion d'état final n'a pas de sens ici car tous les états sont acceptant, ainsi on ne précise pas d'ensemble d'états finaux.

Remarquons que cet automate est un automate fini si et seulement si S est fini, pour lequel S tout entier constitue l'ensemble des états finaux.

Remarquons de plus que par construction, le langage accepté par cet automate coïncide avec celui du réseau de Petri dont il est issu : $\mathscr{L}(G(RdP)) = \mathscr{L}(RdP)$.

D. Propriétés du cône des régions

Comme indiqué en section III-C, le cône des régions correspond aux vecteurs entiers d'un cône convexe polyédrique rationnel saillant de $\mathbb{Q}^{1+2n}$. Nous détaillons ici ces propriétés.

Le cône correspondant est celui ayant la définition (11) mais prenant ses éléments p dans $\mathbb{Q}_+^{1+2n}$. C'est-à-dire le cône $\mathcal{C}$:

$$\mathcal{C} = \left\{ p \in \mathbb{Q}_+^{1+2n} \mid \forall c \in Logs^C, c \cdot p \geq 0 \right\}.$$

En effet cela définit bien un cône convexe car toute combinaison linéaire positive de vecteurs de cet ensemble maintiendra la contrainte de positivité du produit scalaire avec chaque élément de $Logs^C$, et sera donc elle-même dans l'ensemble.

Il est bien saillant car contenu dans $\mathbb{Q}_+^{1+2n}$, ce qui l'empêche de contenir la moindre droite vectorielle.

Enfin, il est bien polyédrique et rationnel car comme expliqué dans [6], il est équivalent de définir un cône polyédrique rationnel par un ensemble fini de vecteurs rationnels générateurs ou bien par l'intersection de demi-espaces définis par un ensemble fini d'hyperplans dont les vecteurs normaux sont rationnels.

Nous avons ici bien affaire à cette deuxième version car $Logs^C$ définit l'ensemble des vecteurs dont sont issus les hyperplans et donc les demi-espaces. De plus c'est un ensemble de vecteurs rationnels car entiers, et enfin cet ensemble est fini car nous partons d'un ensemble de traces $Logs$ qui est lui-même fini.

E. Sur la mesure de simplicité

La mesure retenue pour la simplicité d'un réseau est la fonction $cost$ suivante :

$$cost(RdP) = \sum_{p \in P} \sum_{i=1}^{2n} p[i]$$

où P est l'ensemble des places du réseau RdP mesuré.

Elle a été retenue car c'est une mesure usuelle pour la communauté des réseaux de Petri. Toutefois elle est particulière, et nous n'utilisons que certaines de ces propriétés.

Ainsi nous pouvons abstraire de cette mesure les propriétés souhaitées pour une fonction de coût alternative. Nous souhaitons une fonction $cost$ vérifiant les propriétés suivantes pour un réseau de Petri RdP constitué des places dans P, pour deux places p et p' quelconques, et pour $k \in \mathbb{N}$:

$$cost(RdP) = \sum_{p \in P} cost(p) \tag{15}$$

$$cost(p + p') \geq cost(p) + cost(p') \tag{16}$$

$$cost(k \cdot p) = k \times cost(p) \tag{17}$$

La contrainte (15) vient assurer qu'il est possible de calculer la simplicité du réseau place par place, et que le moins sera le mieux. Les équations (16) et (17) viennent assurer que les places plus petites seront préférées, quitte à en avoir quelques unes en plus.

F. Précision d'une région somme de deux autres

Montrons ici que pour une région $p = q + r \in R(Logs)$ qui s'exprime comme la somme vectorielle de deux autres régions q et r, on a :

$$sp(p) \subseteq sp(q) \cup sp(r).$$

Soit $sp \in sp(p)$ un problème de séparation que la région p résout. Nous avons par déductions successives :

$$\begin{aligned} p \cdot sp &< 0 \\ (q + r) \cdot sp &< 0 \\ q \cdot sp + r \cdot sp &< 0 \\ \exists v \in \{q, r\}, v \cdot r &< 0 \\ \exists v \in \{q, r\}, sp &\in sp(v) \\ sp &\in sp(q) \cup sp(r) \end{aligned}$$

Il en suit qu'on a bien :

$$sp(p) \subseteq sp(q) \cup sp(r).$$

G. Complétude des régions minimales

Montrons ici que l'ensemble des régions minimales se trouve être complet.

Nous savons que cet ensemble est générateur du cône des régions $R(Logs)$ par sa définition, ses propriétés, et les résultats de [6]. Ainsi toute région $p \in R(Logs)$ s'écrit comme :

$$p = \sum_{i=1}^{K} a_i p_i$$

avec les $a_i \in \mathbb{N}$ et où $R_{min}(Logs) = \{p_1, ..., p_K\}$.

Ainsi pour un problème de séparation résoluble $sp \in Logs^{SSP}$, il existe une région p la résolvant, or celle-ci se décompose dans $R_{min}(Logs)$, nous avons donc par déductions successives :

$$\begin{aligned} p \cdot sp &< 0 \\ \left(\sum_{i=1}^{K} a_i p_i \right) \cdot sp &< 0 \\ \sum_{i=1}^{K} a_i (p_i \cdot sp) &< 0 \\ \exists i \in [\![1, K]\!], p_i \cdot sp &< 0 \end{aligned}$$

Ainsi tout problème de séparation résoluble est résolu par au moins une des régions minimales, il en suit que l'ensemble des régions minimales est complet.