Contrôle Des Systèmes Non-déterministes

Rapport de DEA 2004

Jean-Baptiste Raclet encadré par Sophie Pinchinat
IRISA, Projet S4

Table des matières

Remerciements

Je tiens d'abord à remercier mon encadrante de stage, Sophie Pinchinat, pour son investissement, sa gentillesse, et surtout ses conseils méthodologiques.

Je remercie également Stéphane Riedweg pour m'avoir prêté ses transparents de soutenance de thèse et avoir effectué une relecture de ce mémoire.

Je remercie toute l'équipe S4 pour son accueil.

Enfin je remercie Céline, Camille, Maëlle, Jean-François, Jildaz et Serge pour les cafés salvateurs offerts à la cafet et les moments agréables passés à Rennes grâce à eux.

Chapitre 1

Introduction

Les systèmes réactifs désignent l'ensemble des programmes qui ont pour objectif de maintenir une interaction permanente avec leur environnement en réagissant spontanément à des événements physiques extérieurs. La notion de comportement joue donc un rôle central pour leur conception et leur analyse. Ce type de programme est présent dans un large domaine d'applications comme par exemple, les systèmes d'exploitation, les protocoles de communication ou encore les logiciels embarqués pour le transport (avion, train, voiture, navette, ..). Les systèmes réactifs sont souvent des systèmes critiques au sens où l'occurrence d'erreurs au cours de leur fonctionnement peut avoir des conséquences graves du point de vue économique ou de la sécurité des personnes. Leur correction est primordiale. Les problèmes de contrôle s'inscrivent dans cette optique en visant à empêcher les comportements fautifs d'un système.

De manière plus précise, un problème de contrôle possède deux paramètres : d'une part, le système à contrôler et d'autre part, une propriété attendue sur le comportement de ce système. La question est alors de savoir s'il existe une politique de contrôle pour ce système, par exemple en le composant avec un autre système appelé le contrôleur, de sorte que la propriété comportementale soit vérifiée. Le cas échéant, on souhaite synthétiser cette solution.

Les premiers travaux ayant attrait à cette classe de problème sont dûs à Ramadge et Wonham (cf. [RW87a], [RW87b] et [RW89]) qui ont développé une théorie du contrôle basée sur la théorie des langages formels. Ils ont de plus défini un ensemble de problèmes de contrôle parmi lesquels on peut citer :

- le respect des événements incontrôlables : ce problème de contrôle consiste à dire que, parmi l'ensemble des événements possibles du système, certains sont incontrôlables, c'est-à-dire qu'il n'existe pas de moyen de les empêcher. Par exemple, on ne peut pas interdire à un utilisateur d'accéder à un fichier d'un système d'exploitation s'il possède les droits d'accès adéquats sur celui-ci.

- le problème de l'observation partielle : un sous-ensemble des événements du système forme l'ensemble des événements inobservables pour lesquels le contrôleur

ignore si ces événements ont lieu ainsi que leur fréquence.

– le contrôle décentralisé : le but est de synthétiser plusieurs sous-contrôleurs et non un seul afin de répartir la charge de contrôle parmi ces sous-contrôleurs.

Ramadge et Wonham énoncent, pour chacun de ces problèmes, une condition nécessaire et suffisante pour l'existence d'un contrôleur. De plus, lorsqu'il n'existe pas de contrôleur pour un problème donné, ils fournissent un algorithme pour calculer le langage contrôlable maximal. Ces travaux ont été étendus par Thistle et Wonham afin de pouvoir traiter l'aspect infinitaire du comportement des systèmes à contrôler (cf. [TW94a] et [TW94b]).

Une autre approche pour traiter les problèmes de contrôle utilise les logiques temporelles. Elle a été initiée dans [KMTV00] où figurent des bornes de complexité pour des objectifs de contrôle formulés en logique CTL et CTL^*.

Dans [AVW03], la logique du *Mu-calcul* est étendue par l'ajout de formules atomiques indiquant que, par un événement, on puisse boucler sur un état du système. Cette nouvelle logique appelée *Loop Mu-calcul* est particulièrement adaptée aux problèmes de contrôle sous observation partielle.

Enfin dans [Rie03] et [RP03a], la logique du *Mu-calcul quantifié* définie, qui est aussi une extension du *Mu-calcul*, permet notamment d'exprimer des critères d'optimalité sur les contrôleurs.

Des algorithmes de synthèse de contrôleurs existent pour ces logiques. Ils reposent sur la manipulation d'automates d'arbres équivalents aux énoncés logiques correspondant aux propriétés comportementales souhaitées et sur la théorie des jeux de parité associés au système à contrôler et à l'automate d'arbre précédent.

Il faut cependant noter que ces résultats sont énoncés sous l'hypothèse implicite que les systèmes sont déterministes au sens de la théorie des langages. Or cette hypothèse n'est pas réaliste. Par exemple, [Mil89] indique que les systèmes concurrents sont par nature non-déterministes : ils peuvent présenter plusieurs comportements différents en raison des vitesses variables de ses composantes. De même, l'étape d'abstraction d'un système par fusion d'états (obtenu, par exemple, par oubli d'une variable) peut conduire à des systèmes qui ne seront pas déterministes. Il existe de nombreux autres exemples.

Dans [Tho97] est introduit un codage des non-déterministes dans les arbres utilisant une nouvelle action τ :

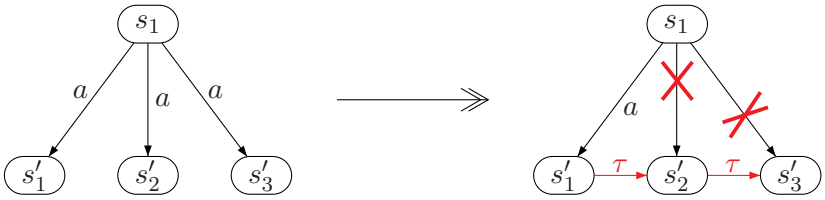


FIG. 1.1 – *Codage du non-déterminisme dans un arbre.*

L'objectif de ce rapport est d'adapter ce codage dans les arbres aux systèmes non-déterministes et de s'appuyer sur les dernières avancées effectuées en terme d'expressivité des logiques afin de réduire un problème de contrôle d'un système non-déterministe en un problème de contrôle d'un système déterministe.

Dans un premier temps, nous définirons les systèmes à contrôler et nous proposerons un codage d'un système non-déterministe en une structure déterministe appelée *processus* (cf. section 2). Ensuite, nous présenterons différentes logiques temporelles permettant d'énoncer des problèmes de contrôle et nous proposerons une transformation de ces derniers afin qu'ils correspondent aux structures déterministes codées (cf. section 3). Cette partie s'achèvera par un théorème énonçant des cas où les problèmes de contrôle d'un système non-déterministe sont décidables. Enfin, nous examinerons une alternative pour aborder cette réduction (cf. section 4) utilisant un autre codage des systèmes non-déterministes en processus ainsi qu'une extension des logiques temporelles permettant d'indiquer que deux événements sont indistinguables pour un contrôleur (ie. le contrôleur doit se comporter de la même façon pour ces deux événements, cf. [ABW03]).

Chapitre 2

Les systèmes à contrôler

La théorie des problèmes de contrôle a été développée autour des processus qui correspondent à des machines à états-transitions déterministes. Un état du système réactif modélisé est représenté par un état du processus auquel on assigne un ensemble de propositions atomiques qui codent les propriétés vraies lorsque le système est dans cet état. Une transition dans le processus correspond à un changement d´état du système suite à l'événement noté par l'étiquette de la transition. Examinons de manière plus formelle cette notion de processus :

2.1 Les processus

On pose $\Sigma = \{a, b, ...\}$ un ensemble fini d'événements, $AP = \{p, q, ...\}$ un ensemble fini de propositions atomiques et $Var = \{X, Y, ...\}$ un ensemble infini de variables. La définition d'un processus est la suivante :

Définition 1 (Processus)
Un processus sur Γ *est un tuple* $\mathcal{S} =< \Gamma, S, s^0, t, L >$ *avec :*

- $\Gamma \subseteq AP$ *un ensemble fini de propositions atomiques ;*
- S *un ensemble d'états ;*
- $s^0 \in S$ *l'état initial ;*
- $t : S \times \Sigma \to S$ *la fonction partielle de transition du processus ;*
- $L : S \to \mathcal{P}(\Gamma)$ *associe à chaque état* $s \in S$ *du processus un ensemble* $L(s) \subseteq \Gamma$ *de propositions atomiques.*

De plus, on dira qu'un processus $\mathcal{S}$ est *fini* si S est fini et qu'il est *complet* si pour tout couple $(\sigma, s) \in \Sigma \times S$, $t(s, \sigma)$ est défini.

Les processus sont combinés par *produit synchrone*. Par exemple, on obtient le système contrôlé par le produit synchrone du système et de son contrôleur :

Définition 2 (Produit synchrone)
Soient deux processus $\mathcal{S}_1 =< \Gamma_1, S_1, s_1^0, t_1, L_1 >$ *et* $\mathcal{S}_2 =< \Gamma_2, S_2, s_2^0, t_2, L_2 >$ *tels*

que Γ_1 *et* Γ_2 *soient disjoints, leur produit synchrone est le processus suivant noté* $\mathcal{S}_1 \times \mathcal{S}_2 =< \Gamma, S_1 \times S_2, (s_1^0, s_2^0), t, L >$ *avec* $\Gamma = \Gamma_1 \cup \Gamma_2$ *tel que :*

- $t((s_1, s_2), \sigma) = (s_1', s_2')$ *si* $t_1(s_1, \sigma) = s_1'$ *et si* $t_2(s_2, \sigma) = s_2'$,
- $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$.

Définissons maintenant la notion centrale dans notre étude : les *systèmes non-déterministes*.

2.2 Les systèmes non-déterministes

Définition 3 (Système non-déterministe)
Un système non-déterministe sur Γ *est un tuple* $\mathcal{S} =< \Gamma, S, s^0, t, L >$ *similaire à un processus mais dont les transitions sont modélisées non plus par une fonction mais par une relation* $t \subseteq S \times \Sigma \times S$.

Par commodité, on notera $t(s, a)$ l'ensemble suivant des états du processus :

$$\{s' \in S : (s, a, s') \in t\}.$$

La théorie du contrôle a été développée autour des processus qui sont, par définition, déterministes. Nous proposons un codage des systèmes non-déterministes en processus afin de pouvoir réduire un problème de contrôle d'un système non-déterministe en un problème de contrôle d'un processus que l'on saura donc traiter :

2.3 Codage des systèmes non-déterministes en processus

Dans un premier temps, il faut remarquer que l'on ne peut pas effectuer une déterminisation comme celle que l'on fait habituellement sur les automates de mots. En effet, considérons l'exemple suivant composé à gauche d'une modélisation non-déterministe d'une machine à café et à droite de sa version déterministe :

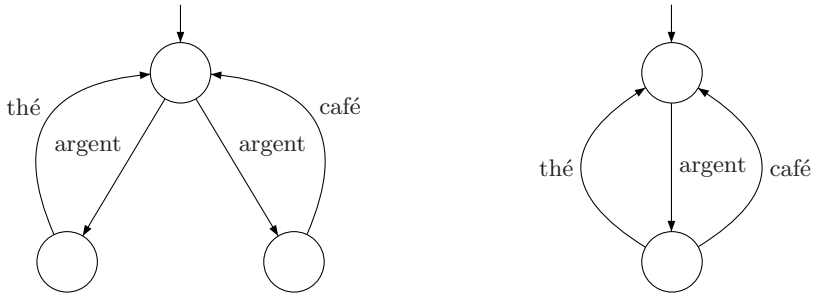


FIG. 2.1 – *Modélisation d'une machine à café.*

Après avoir mis de l'argent dans la machine, on peut, pour la machine de gauche, soit prendre du café soit prendre du thé alors qu'à droite c'est la machine qui choisit entre le thé et le café. Ces systèmes ne se comportent donc pas de la même façon. Pour

déterminer l'équivalence de systèmes dans le cadre des problèmes de contrôle, il faut considérer l'équivalence comportementale appelée *bisimulation* :

Définition 4 (Bisimulation)
Soient $\mathcal{S}_1 =< \Gamma, S_1, s_1^0, t_1, L_1 >$ *et* $\mathcal{S}_2 =< \Gamma, S_2, s_2^0, t_2, L_2 >$ *deux systèmes. Une relation binaire* $\mathcal{R}$ *est une bisimulation sur* $\mathcal{S}_1$ *et* $\mathcal{S}_2$ *si elle est totale et si pour tout* $s_1 \in S_1$ *et pour tout* $s_2 \in S_2$, $\mathcal{R}(s_1, s_2)$ *implique :*

- $L_1(s_1) = L_2(s_2)$;
- *pour toute transition* $s_1' \in t(s_1, a)$, *il existe* $s_2' \in S_2$ *tel que* $s_2' \in t(s_2, a)$ *et* $\mathcal{R}(s_1', s_2')$;
- *réciproquement :* $s_2' \in t(s_2, a)$, *il existe* $s_1' \in S_1$ *tel que* $s_1' \in t(s_1, a)$ *et* $\mathcal{R}(s_1', s_2')$.

Une bisimulation entre $\mathcal{S}_1$ *et* $\mathcal{S}_2$, *notée* $\mathcal{R} : \mathcal{S}_1 \leftrightarrows \mathcal{S}_2$, *est une bisimulation sur* $\mathcal{S}_1$ *et* $\mathcal{S}_2$ *qui relie les états initiaux.*

On note $\mathcal{S}_1 \leftrightarrows \mathcal{S}_2$ *s'il existe une bisimulation entre* $\mathcal{S}_1$ *et* $\mathcal{S}_2$.

Par exemple, les deux systèmes modélisant une machine à café dans la figure 2.1 ne sont pas bisimilaires.

Le codage que nous proposons repose sur le codage des non-déterministes dans les arbres défini dans [Tho97] et représenté dans la figure 2.2 ci-dessous :

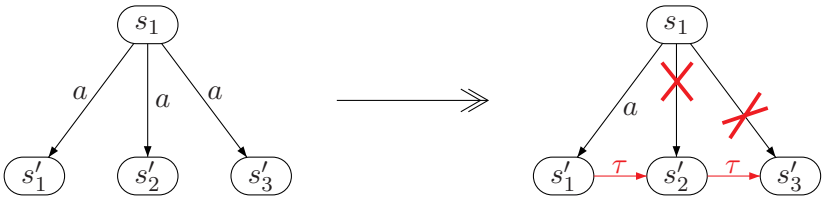


FIG. 2.2 – *Codage du non-déterminisme dans un arbre.*

Lorsqu'on a un non-déterminisme en a, on conserve une a-transition et on parcourt l'ensemble des autres a-successeurs grâce à une nouvelle action τ.

Ce principe s'applique aux arbres mais ne peut pas être directement généralisé aux systèmes non-déterministes comme le montre l'exemple de la figure 2.3 où un cas de

non-déterminisme en τ est généré.

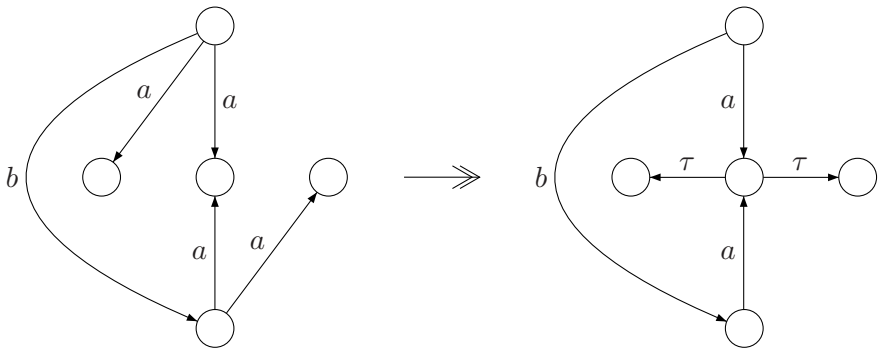


FIG. 2.3 – *Exemple où le codage de [Tho97] génère un cas de non-déterminisme.*

La démarche intuitive que nous proposons est la suivante : étant donné un processus $\mathcal{S}$ non-déterministe, effectuer son dépliage en arbre $T_{\mathcal{S},<}$ (avec $<$ un ordre total sur les sommets de l'arbre). Ajouter les τ-transitions pour obtenir l'arbre $T^{\tau}_{\mathcal{S},<}$ et replier "convenablement" cet arbre pour obtenir un processus $\mathcal{S}^{\tau}$ déterministe.

Nous proposons un codage utilisant deux transformations $\mathcal{T}_1$ et $\mathcal{T}_{2,<}$ tel que le processus $\mathcal{S}^{\tau}$ produit à partir de $\mathcal{S}$ soit déterministe et bisimilaire à $T^{\tau}_{\mathcal{S},<}$.

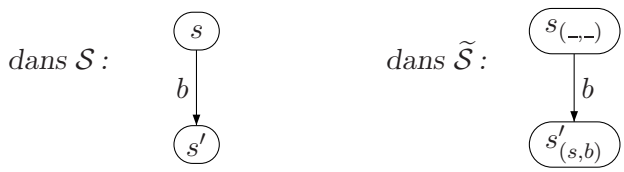
2.3.1 Transformation $\mathcal{T}_1$: sauvegarde du passé immédiat

La transformation $\mathcal{T}_1$ consiste à stocker dans les états du système le passé immédiat. Ceci correspond à effectuer un dépliage sur un niveau du processus considéré.

De manière plus précise, étant donné un processus $\mathcal{S}$, on sauvegarde, dans chaque état s', l'état source s et l'événement a de la transition qui mène de s à s' dans $\mathcal{S}$. On adapte ensuite la fonction de transition à ce nouvel ensemble d'états :

Définition 5 (Transformation $\mathcal{T}_1$)
Soit un système non-déterministe $\mathcal{S} =< \Gamma, S, s^0, t, L >$, *on construit par* $\mathcal{T}_1$ *le système* $\widetilde{\mathcal{S}} =< \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L} >$ *de la façon suivante :*

- $\widetilde{S} = \{s^0\} \cup \{s'_{(s,a)} : s' \in t(s,a)\}$;
- $\widetilde{s^0} = s^0$;
- *si* $s' \in t(s,b)$ *alors* $s'_{(s,b)} \in \widetilde{t}(s_{(r,a)}, b) \quad \forall r \in S, \forall a \in \Sigma$ *tels que* $s_{(r,a)} \in \widetilde{S}$ *:*

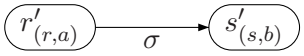


si de plus $s = \widetilde{s^0}$ *alors* $s'_{(\widetilde{s^0},b)} \in \widetilde{t}(\widetilde{s^0}, b)$;

– $\widetilde{L}(\widetilde{s^0}) = L(s^0)$ *et*
$\widetilde{L}(s'_{(s,a)}) = L(s')$.

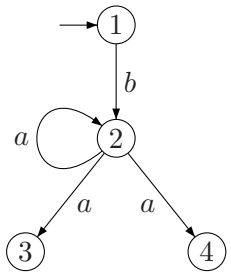
L'alphabet des événements possibles pour $\widetilde{\mathcal{S}}$ *est identique à celui de* $\mathcal{S}$.

Remarque : pour chaque transition de $\widetilde{\mathcal{S}}$ de la forme :

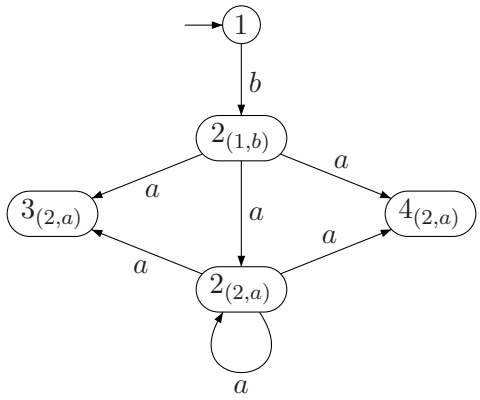


on peut déduire, par construction de $\widetilde{t}$, que $s = r'$ d'une part et que $\sigma = b$ d'autre part.

Exemple : on suppose que l'ensemble de propositions atomiques assigné à chaque état du système $\mathcal{S}$ ci-dessous est l'ensemble vide.



Système $\mathcal{S}$



Système $\mathcal{T}_1(\mathcal{S})$

En considérant la relation binaire $\mathcal{R}$ suivante (qui relie les états initiaux de $\mathcal{S}$ et $\mathcal{T}_1(\mathcal{S})$) :

$$\mathcal{R}(2, 2_{(1,b)})$$
$$\mathcal{R}(2, 2_{(2,a)})$$
$$\mathcal{R}(3, 3_{(2,a)})$$
$$\mathcal{R}(4, 4_{(2,a)})$$

on remarque que, sur cet exemple, les systèmes $\mathcal{S}$ et $\mathcal{T}_1(\mathcal{S})$ sont bisimilaires. Ce résultat se généralise à l'ensemble des systèmes :

Propriété 1
Le système non-déterministe $\mathcal{S}$ *et le système non-déterministe* $\widetilde{\mathcal{S}}$ *obtenu par la transformation* $\mathcal{T}_1$ *sont bisimilaires.*

Démonstration
Soit la relation binaire $\mathcal{R} \subseteq S \times \widetilde{S}$ *suivante :*

$$\begin{cases} \mathcal{R}(s^0, \widetilde{s^0}) \\ \mathcal{R}(s', s'_{(s,a)}) \quad \forall s \in S, \forall a \in \Sigma \text{ tels que } s_{(s,a)} \in \widetilde{S}. \end{cases}$$

Vérifions que cette relation définit bien une relation de bisimilarité entre $\mathcal{S}$ *et* $\mathcal{T}_1(\mathcal{S}) = \widetilde{S}$ *:*

- *Supposons que* $r' \in t(s', b)$ *et que* $\mathcal{R}(s', s'_{(s,a)})$*, par définition de* $\widetilde{t}$*, on a :* $r'_{(s',b)} \in \widetilde{t}(s'_{(s,a)}, b)$ *et on a bien* $\mathcal{R}(r', r'_{(s',b)})$ *par définition de* $\mathcal{R}$*.*
- *Supposons que* $r'_{(s',b)} \in \widetilde{t}(s'_{(s,a)}, b)$ *et que* $\mathcal{R}(s', s'_{(s,a)})$*, par définition de* $r'_{(s',b)}$*, on a :* $r' \in t(s', b)$ *et on a bien* $\mathcal{R}(r', r'_{(s',b)})$ *par définition de* $\mathcal{R}$*.*

Le cas de l'état initial $\mathcal{R}(s^0, \widetilde{s^0})$ *est similaire et* $\widetilde{L}(s'_{(s,a)}) = L(s')$ *est vrai par définition de* $\mathcal{T}_1$*.*

$\diamondsuit$

Complexité : Dans chaque sommet de $\widetilde{S} = \mathcal{T}_1(\mathcal{S})$, on sauvegarde l'état source et l'évènement étiquette de la transition, la taille du système non-déterministe $\widetilde{S}$ obtenu est donc quadratique par rapport à la taille du système non-déterministe initial $\mathcal{S}$:

$$|\widetilde{\mathcal{S}}| = O(|\mathcal{S}| \times |\mathcal{S}| \times |\Sigma|).$$

2.3.2 Transformation $\mathcal{T}_{2,<}$: introduction d'événements internes

Le principe de la transformation $\mathcal{T}_{2,<}$ est schématisé dans la figure 2.2 présentée précédemment. On ajoute dans le système une action interne τ et on munit chaque processus d'un ordre total $<$ sur l'ensemble de ses états. On note par $Succ(s, a)$ l'ensemble des successeurs du sommet s par l'événement a, ordonné par $<$:

$$Succ(s, a) := \{t(s, a), <\}$$

On écrit $s_1 <_i s_2$ lorsque s_1 et s_2 appartiennent à $Succ(s, a)$, que $s_1 < s_2$ et qu'il n'existe pas de $s_3 \in Succ(s, a)$ tel $s_1 < s_3 < s_2$.

Définition 6
Étant donné le système $\mathcal{S} =< \Gamma, S, s^0, t, L >$*, par la transformation* $\mathcal{T}_{2,<}$ *on construit ainsi la fonction de transition* t^τ *du système* $\mathcal{S}^\tau =< \Gamma, S, s^0, t^\tau, L >$ *:*

- *si* $|Succ(s, a)| \leq 1$ *alors* $t^\tau(s, a) = t(s, a)$ *;*
- *sinon, soit* $Succ(s, a) = \{s_1, s_2, ..., s_n\}$ *avec* $s_1 <_i s_2 <_i ... <_i s_n$ *:*

$$\begin{array}{lcll} t^\tau(s, a) & = & s_1 & \text{avec } s_1 = min(Succ(s, a)) \text{ et} \\ t^\tau(s_i, \tau) & = & s_{i+1}, & \forall i < |Succ(s, a)| \end{array}$$

L'ensemble des événements possibles de $\mathcal{S}^\tau$ *est* $\Sigma^\tau = \Sigma \cup \{\tau\}$*.*

Remarque : il faut noter qu'il existe une τ-transition entre s_1 et s_2 si et seulement si $s_1 <_i s_2$.

Par la suite, on dira que $s_1, s_2, ..., s_n$ *appartiennent à la même chaîne de* τ (celle issue de l'état s et de l'événement a) s'il existe un couple (s, a) tel que $Succ(s, a) = \{s_1, s_2, ..., s_n\}$.

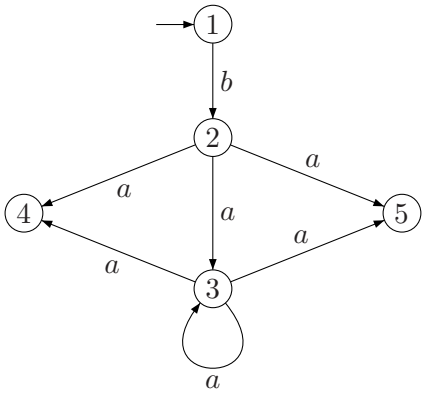
Exemple : Soit le système $\mathcal{S}$ ci-dessous, muni de l'ordre total sur ses états suivant :

$$5 < 2 < 3 < 4 < 1.$$

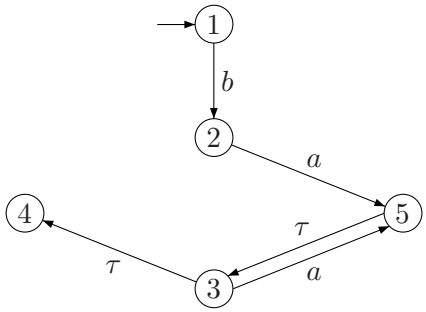
On a deux cas de non-déterminisme à résoudre :

- $Succ(2, a) = \{3, 4, 5\}$ avec, d'après l'ordre total $<$: $\quad 5 <_i 3 <_i 4$;
- $Succ(3, a) = \{3, 4, 5\}$ avec, d'après l'ordre total $<$: $\quad 5 <_i 3 <_i 4$.

Par définition de $\mathcal{T}_{2,<}$ on obtient le système $\mathcal{S}^\tau$ ci-dessous :



Système $\mathcal{S}$



Système $\mathcal{S}^\tau$

2.3.3 Composition des transformations $\mathcal{T}_1$ et $\mathcal{T}_{2,<}$

Dans cette section, nous allons prouver qu'en composant les transformations $\mathcal{T}_1$ et $\mathcal{T}_{2,<}$, on obtient, à partir d'un système non-déterministe, un processus.

Notations : par la suite, on notera par $cod_<$ la transformation $(\mathcal{T}_{2,<} \circ \mathcal{T}_1)$ et $\mathcal{S}^\tau$ le processus correspondant au système non-déterministe $\mathcal{S}$. Ainsi :

$$cod_<(\mathcal{S}) = (\mathcal{T}_{2,<} \circ \mathcal{T}_1)(\mathcal{S}) = \mathcal{T}_{2,<}(\mathcal{T}_1(\mathcal{S})) = \mathcal{S}^\tau.$$

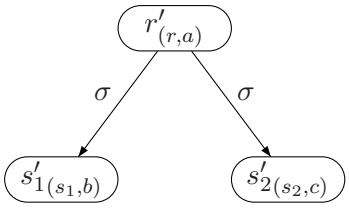
Propriété 2
Soit $\mathcal{S}$ un système non-déterministe, $cod_<(\mathcal{S})$ est déterministe (ie. est un processus).

Démonstration
On pose :

- $\mathcal{S} =< \Gamma, S, s^0, t, L >$
- $\widetilde{\mathcal{S}} = \mathcal{T}_1(\mathcal{S}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L} >$
- $\mathcal{S}^\tau = \mathcal{T}_{2,<}(\widetilde{\mathcal{S}}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, t^\tau, \widetilde{L} >$

Supposons que $\mathcal{S}^\tau$ ne soit pas déterministe. Deux cas de figures peuvent se présenter :

1er cas : *On est dans la situation suivante :*



– *si $\sigma \neq \tau$:*
Nécessairement, $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{1(s_1,b)}$ et $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{2(s_2,c)}$ car $\sigma \neq \tau$.
Par construction de t^τ, on a :

$$\begin{array}{rcl} s'_{1(s_1,b)} & = & min(Succ(r'_{(r,a)}, b)) \quad et \\ s'_{2(s_2,c)} & = & min(Succ(r'_{(r,a)}, b)) \end{array}$$

car $\sigma \neq \tau$. D'où : $s'_{1(s_1,b)} = s'_{2(s_2,c)}$.

– *si $\sigma = \tau$:*
Supposons que $s'_{2(s_2,c)}$, $r'_{(r,a)}$ et $s'_{1(s_1,b)}$ fassent partie de la même chaîne de τ. Alors il existe un état source $q'_{(q,e)}$ tel que :

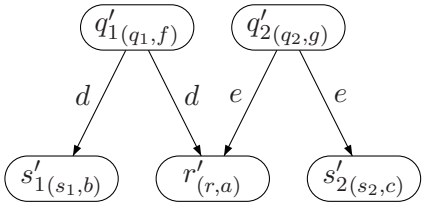
$$\{s'_{2(s_2,c)}, r'_{(r,a)}, s'_{1(s_1,b)}\} \in Succ(q'_{(q,e)}, d).$$

Mais compte tenu de l'existence des τ-transitions, on a alors :

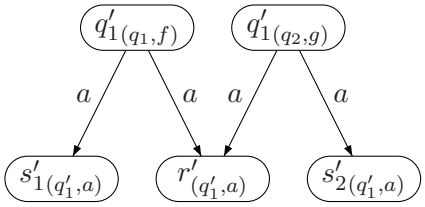
$$r'_{(r,a)} <_i s'_{1(s_1,b)} \text{ et } r'_{(r,a)} <_i s'_{2(s_2,c)}$$

Les τ-transitions sont donc issues de deux chaînes de τ distinctes.

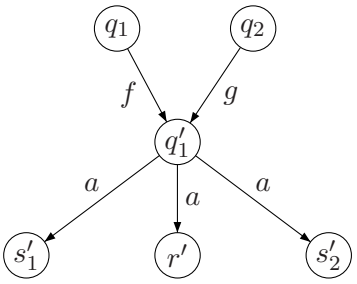
Le système $\widetilde{\mathcal{S}}$ est donc de la forme :



Par construction de $\widetilde{t}$, on a d'une part $r = q'_1 = s_1$ et $b = d = a$ et d'autre part, $r = q'_2 = s_2$ et $a = e = c$.
Le système $\widetilde{\mathcal{S}}$ est donc de la forme :



On peut donc en déduire, d'après les informations sur le passé immédiat contenues dans les états, que le système $\mathcal{S}$ est de la forme :

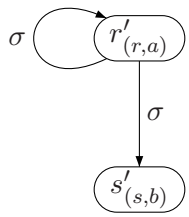


Mais alors, dans $\widetilde{\mathcal{S}}$, on devrait avoir :

$$Succ(q'_{1(q_1,f)}) = \{s'_{1(q'_1,a)}, r'_{(q'_1,a)}, s'_{2(q'_1,a)}\}$$

c'est à dire, $s'_{1(q'_1,a)}$, $r'_{(q'_1,a)}$ et $s'_{2(q'_1,a)}$ appartiennent à la même chaîne de τ. Contradiction !
Ce cas de figure est donc impossible.

2nd cas : *on est dans la situation suivante :*



– *si $\sigma \neq \tau$:*
Par construction de t^τ, on a nécessairement $\widetilde{t}(r'_{(r,a)}, \sigma) = r'_{(r,a)}$ et $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{(s,b)}$ car $\sigma \neq \tau$.
Et, de plus :

$$\begin{array}{rcl} r'_{(r,a)} & = & min(Succ(r'_{(r,a)}, a)) \quad et \\ s'_{(r,a)} & = & min(Succ(r'_{(r,a)}, a)) \end{array}$$

On a donc : $r'_{(r,a)} = s'_{(r,a)}$.

– *si $\sigma = \tau$:*
Ce cas de figure est impossible, car par définition de t^τ on ne crée pas de boucle de τ : $t^\tau(r'_{(r,a)}, \tau) = r'_{(r,a)}$ équivaut à $r'_{(r,a)} <_i r'_{(r,a)}$ ce qui est impossible.

$\Diamond$

Soient $\mathcal{S}$ un système non-déterministe, $T_\mathcal{S}$ le dépliage en arbre de ce système et $T^\tau_{\mathcal{S},<}$ l'arbre correspondant à l'application de la transformation décrite dans la figure 2.2 sur

l'arbre $T_{\mathcal{S}}$, alors on démontre la correction de la construction $cod_<$ grâce à la propriété 3 suivante :

Propriété 3 (Correction de $cod_<$)
Soit un système $\mathcal{S}$ non-déterministe, le processus $cod_<(\mathcal{S})$ est bisimilaire à l'arbre $T^{\tau}_{\mathcal{S},<}$.

Démonstration
Les états de $T^{\tau}_{\mathcal{S},<}$ sont les mots v de $(\Sigma \cup \{\tau\})^*$ correspondant aux traces des exécutions successives du système $\mathcal{S}$.

Par la fonction stateseq on associe la séquence d'états de S^* correspondant aux états successifs visités lors de l'exécution d'un $v \in (\Sigma \cup \{\tau\})^*$. En cas de non-déterminisme, on peut désigner l'état suivant grâce au nombre de τ contenu dans v et à l'ordre total sur les sommets. La fonction stateseq est définie de manière récursive de la manière suivante :

$$\begin{cases} stateseq(\epsilon) = s^0. \\ stateseq(va\tau^i) = stateseq(v).s_i & \text{avec } v \in (\Sigma \cup \{\tau\})^* \text{ et } s_i \text{ l'élément} \\ & \text{d'ordre } i \text{ dans } Succ(last(stateseq(v), a)). \end{cases}$$

La fonction $last : S^* \to S$ retourne le dernier état de la séquence d'états en entrée. Nous proposons la relation de bisimilarité suivante entre les états de $T^{\tau}_{\mathcal{S},<}$ et ceux de $\mathcal{S}^{\tau}$:

$$\mathcal{R} \subseteq (\Sigma \cup \{\tau\})^* \times ((S \times \Sigma \times S) \cup \{\widetilde{s^0}\})$$

$$\begin{cases} \mathcal{R}(\epsilon, \widetilde{s^0}) \\ \mathcal{R}(u, s'_{(s,a)}) \quad \text{avec } s' = last(stateseq(u)). \end{cases}$$

Vérifions que $\mathcal{R}$ définit bien une relation de bisimilarité (on considère uniquement le cas où $u \neq \epsilon$) : supposons que $\mathcal{R}(u, s'_{(s,a)})$ avec $s' = last(stateseq(u))$. Deux cas sont à étudier :

- ***l'événement suivant est un** $a \in \Sigma$:*
  Par définition de stateseq, on a :
  $$stateseq(ua) = stateseq(u).s''_0$$
  avec s''_0 l'élément d'ordre 0 dans $Succ(last(stateseq(u)), a)$ c'est à dire dans l'ensemble $Succ(s', a)$.
  De plus :
  $\widetilde{t}(s'_{(s,a)}, a) = s''_{1(s',a)}$ avec $s''_1 = t(s', a)$ dans $\mathcal{S}$ c'est à dire s''_1 d'ordre 0 dans $Succ(s', a)$.
  D'où, $\mathcal{R}(ua, s''_{(s',a)})$ avec $s'' = s''_0 = s''_1$.

– ***l'événement suivant est un*** τ *:*
Nécessairement par définition du codage, u *est de la forme* $va\tau^i$ *:*

$$\begin{aligned} stateseq(u\tau) &= stateseq(va\tau^i.\tau) \\ &= stateseq(v).s''_{i+1} \end{aligned}$$

avec s''_{i+1} *l'élément d'ordre* $i+1$ *dans* $Succ(last(stateseq(v)), a)$.
De plus :
$\widetilde{t}(s'_{(s,a)}, \tau) = s''_{(s,a)}$ *avec* $s'_{(s,a)} <_i s''_{(s,a)}$.
Or par hypothèse, s' *est égal à* $last(stateseq(va\tau^i))$ *et est donc d'ordre* i *dans* $Succ(last(stateseq(v)), a)$ *(que l'on peut aussi noter* $Succ(s, a)$ *d'après le passé immédiat stocké dans* $s'_{(s,a)}$*) donc* s'' *est d'ordre* $i+1$ *dans* $Succ(s, a)$ *c'est à dire dans* $Succ(last(stateseq(v)), a)$.
D'où $s''_{i+1} = s''$ *et* $\mathcal{R}(u\tau, s''_{(s,a)})$ *avec* $s'' = last(stateseq(u\tau))$.

♢

2.3.4 Bilan de complexité

Pour évaluer la complexité de $cod_<$, examinons chaque étape de cette construction :

– *Transformation* $\mathcal{T}_1$ *:* comme indiqué précédemment, on a $|\widetilde{\mathcal{S}}| = O(|\mathcal{S}| \times |\mathcal{S}| \times |\Sigma|)$;
– *Transformation* $\mathcal{T}_{2,<}$ *:* seule la fonction de transition du système de départ est modifiée; on ajoute au plus autant d'arcs que l'on en supprime.

La construction $cod_<$ permettant de construire un processus à partir d'un système non-déterministe est donc quadratique en la taille du système non-déterministe initial.

Chapitre 3

Spécifications logiques des problèmes de contrôle

[Rie03], [RP03b] et [RP03a] proposent une approche pour traiter les problèmes de contrôle qui est intuitivement la suivante : de manière simple, contrôler un système c'est interdire à certains événements d'avoir lieu ou encore, si on regarde le système à contrôler comme une machine à états-transitions, c'est interdire certaines transitions. Une façon d'indiquer qu'une transition d'un processus est autorisée ou pas est d'inclure une proposition atomique x_a dans $L(s)$ si la transition de l'état s vers l'état $t(s, a)$ est autorisée.

Maintenant, si on prend un ensemble $X = \{x_a, x_b, ...\}$ de propositions atomiques avec une proposition atomique pour chaque événement possible de $\Sigma = \{a, b, ...\}$, un problème de contrôle se ramène à un problème d'étiquetage du système à contrôler. Plus concrètement, on va chercher à synthétiser un processus $\mathcal{E}$ qui va, par produit synchrone, étiqueter le processus $\mathcal{S}$ à contrôler de telle façon que l'ensemble X indiquera les transitions autorisées pour le processus étiqueté $\mathcal{S} \times \mathcal{E}$.

Cette approche se concrétise dans une logique appelée *Mu-calcul quantifié*. Cette logique est une extension du *Mu-calcul* comprenant un opérateur de quantification aux propositions atomiques du Mu-calcul. Soit $\mathcal{S}$ le système à contrôler et ϕ un objectif de contrôle exprimé en Mu-calcul quantifié, un problème de contrôle est alors décrit par une équation de la forme : $\mathcal{S} \models \exists X.\phi$ dont la solution est un processus étiquetant $\mathcal{E}$ sur X tel que le système étiqueté $\mathcal{S} \times \mathcal{E}$ satisfasse la formule ϕ. Un problème de contrôle (ou d'étiquetage) se ramène donc à un problème de *model-checking* d'une formule de Mu-calcul quantifié. Dans [Rie03], le *model-checking* de cette logique est montré décidable et un algorithme de synthèse du processus étiquetant $\mathcal{E}$ à partir de $\mathcal{S}$ et de ϕ est fourni.

Cette logique est très expressive, elle permet, par exemple, de traiter les classiques problèmes de contrôle avec respect des événements incontrôlables ou de contrôle décentralisé mais aussi d'exprimer des critères d'optimalité d'un contrôleur (cf. [Rie03]). Nous allons utiliser cette logique afin de réduire un énoncé de problème de contrôle d'un système non-déterministe en un problème de contrôle d'un processus correspond à un codage du système non-déterministe.

3.1 La logique du Mu-calcul quantifié

Avant d'étudier le Mu-calcul quantifié, présentons brièvement le Mu-calcul. On pose $AP = \{p, q, ...\}$ un ensemble fini de propositions atomiques et $Var = \{X, Y, ...\}$ un ensemble infini de variables.

3.1.1 Le Mu-calcul

La logique du Mu-calcul a été introduite par [Koz83]. Sa syntaxe est la suivante :

Syntaxe du Mu-calcul

Définition 7 (Syntaxe du Mu-calcul)
L'ensemble des formules du Mu-calcul sur $\Gamma \subseteq AP$*, noté* $L_\mu(\Gamma)$ *est défini par la grammaire :*

$$\top \mid p \mid X \mid \langle a \rangle \beta_1 \mid \neg\beta_1 \mid \beta_1 \vee \beta_2 \mid \mu X.\beta_1(X)$$

avec β_1, $\beta_2 \in L_\mu(\Gamma)$, $p \in \Gamma$, $a \in \Sigma$ *et* $X \in Var$.

Pour assurer l'existence de points fixes, on requiert que la variable X *soit sous la portée d'un nombre pair de symboles de négations* $\neg$ *dans* $\beta_1(X)$.

On appelle formule *une formule de Mu-calcul sans variable libre.*

On facilite l'écriture de formule de Mu-calcul par l'utilisation de notations répertoriées dans le tableau ci dessous :

Notation	Équivalent en Mu-calcul
$\bot$	$\neg\top$
$[a]\beta_1$	$\neg \langle a \rangle (\neg\beta_1)$
$\beta_1 \wedge \beta_2$	$\neg(\neg\beta_1 \vee \neg\beta_2)$
$\beta_1 \Rightarrow \beta_2$	$\neg\beta_1 \vee \beta_2$
$\rightarrow^a$	$\langle a \rangle \top$
$\not\rightarrow^a$	$[a]\bot$
$\nu X.\beta_1(X)$	$\neg\mu X.\beta_1(\neg X)$
$AG(\beta_1)$	$\nu X. \bigvee_{a\in\Sigma}[a]X \wedge \beta_1$

FIG. 3.1 – *Macros pour les formules de Mu-calcul*

La notation $AG(\beta)$ est empreintée à la logique temporelle CTL et signifie : "β est toujours vraie" ou "β est un invariant".

Sémantique du Mu-calcul sur des systèmes non-déterministes

La sémantique d'une formule de Mu-calcul sur un système non-déterministe désigne le sous-ensemble des états de ce système qui satisfont la formule.

Soit $val : Var \to \mathcal{P}(S)$, la fonction qui instancie les variables libres d'une formule de Mu-calcul, la sémantique du Mu-calcul est inductivement la suivante :

Définition 8 (Sémantique du Mu-calcul)
L'interprétation $[\![\alpha]\!]_{\mathcal{S}}^{[val]}$ *d'une formule* α *de Mu-calcul est donnée de manière inductive, par rapport à un système non-déterministe* $\mathcal{S} =< \Gamma, S, s^0, t, L >$ *et une fonction de valuation* $val : Var \to \mathcal{P}(S)$ *:*

$$
\begin{aligned}
&[\![\top]\!]_{\mathcal{S}}^{[val]} = S \\
&[\![p]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid p \in L(s)\} \\
&[\![X]\!]_{\mathcal{S}}^{[val]} = val(X) \\
&[\![\neg\alpha]\!]_{\mathcal{S}}^{[val]} = S \setminus [\![\alpha]\!]_{\mathcal{S}}^{[val]} \\
&[\![\beta_1 \vee \beta_2]\!]_{\mathcal{S}}^{[val]} = [\![\beta_1]\!]_{\mathcal{S}}^{[val]} \cup [\![\beta_2]\!]_{\mathcal{S}}^{[val]} \\
&[\![\langle a \rangle\, \beta]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists s' : s' \in t(s,a) \text{ et } s' \in [\![\beta]\!]_{\mathcal{S}}^{[val]}\} \\
&[\![\mu X.\beta(X)]\!]_{\mathcal{S}}^{[val]} = \cap\{V \subseteq S \mid [\![\beta]\!]_{\mathcal{S}}^{[val(V/X]} \subseteq V\}
\end{aligned}
$$

avec $val(V/X) : Var \to \mathcal{P}(S)$ *la valuation définie par :*

$$
val(V/X)(Y) = \begin{cases} val(Y) & \text{si } Y \neq X \\ V & \text{sinon .} \end{cases}
$$

On dit que le système $\mathcal{S}$ satisfait la formule α si l'état initial s^0 de $\mathcal{S}$ appartient à $[\![\alpha]\!]_{\mathcal{S}}$ et on note $\mathcal{S} \models \alpha$. $\mathcal{S}$ est alors appelé un "modèle" de α.

Maintenant, nous allons étudier le Mu-calcul quantifié qui généralise le Mu-calcul par des quantifications $\exists\Lambda$ sur un ensemble Λ de propositions atomiques des formules du Mu-calcul :

3.1.2 Syntaxe du Mu-calcul quantifié

Définition 9 (Syntaxe du Mu-calcul quantifié)
L'ensemble des formules du Mu-calcul quantifié sur $\Gamma \subseteq AP$*, noté* $QL_\mu(\Gamma)$ *est défini par la grammaire :*

$$\exists\Lambda.\alpha \mid \neg\alpha_1 \mid \alpha_1 \vee \alpha_2 \mid \beta$$

avec $\Lambda \subseteq AP$ *disjoint de* Γ*,* α *est une formule de* $QL_\mu(\Gamma \cup \Lambda)$*,* α_1 *et* α_2 *formules de* $QL_\mu(\Gamma)$ *et* β *est une formule du Mu-calcul sur* Γ*.*

On écrit $\forall\Lambda.\alpha$ la formule $\neg(\exists\Lambda.\neg\alpha)$. De même, on écrira abusivement $\exists x.\alpha$ pour la formule $\exists\{x\}.\alpha$.

Remarque : La syntaxe du Mu-calcul quantifié ne permet pas de quantifications à l'intérieur des points fixes des formules de Mu-calcul. En général, les opérateurs de points fixes et les quantifications ne commutent pas.

3.1.3 Sémantique du Mu-calcul quantifié

Jusqu'à présent, dans l'ensemble des travaux relatifs au Mu-calcul quantifié, celui-ci était interprété sur les processus donc sur une structure déterministe. Nous proposons la sémantique suivante sur les systèmes non-déterministes généralisant ainsi l'interprétation des formules de Mu-calcul quantifié :

Définition 10 (Processus étiquetant)
Un processus étiquetant $\mathcal{E}$ sur $\Lambda \subseteq AP$ est un processus complet sur Λ. Pour tout système $\mathcal{S} =< \Gamma, S, s^0, t, L >$ sur Γ disjoint de Λ, le produit synchrone $\mathcal{S} \times \mathcal{E}$ est appelé étiquetage de $\mathcal{S}$ par $\mathcal{E}$ sur Λ. L'ensemble des processus étiquetant sur Λ est noté Lab_Λ.

Définition 11 (Sémantique du Mu-calcul quantifié)
La sémantique des quantifications $\exists\Lambda$ interprétées par rapport à un système non-déterministe $\mathcal{S} =< \Gamma, S, s^0, t, L >$ et à la fonction de valuation $val : Var \to \mathcal{P}(S)$ est :

$$[\![\exists\Lambda.\alpha]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists \mathcal{E} =< \Lambda, E, \epsilon^0, t', L' > \in Lab_\Lambda,\ (s, \epsilon^0) \in [\![\alpha]\!]_{\mathcal{S}\times\mathcal{E}}^{[val\times E]}\}$$

avec $(val \times E)(X) = val(X) \times E$ pour tout $X \in Var$.

La sémantique des formules de $\mathrm{Q}L_\mu$ sur les systèmes non-déterministes que nous proposons est la même que celle sur les processus. Il faut bien noter que, dans le cas des systèmes non-déterministes, l'étiquetage reste assuré par un processus et donc par une structure **déterministe**. Nous verrons dans le paragraphe 3.2.1 l'importance de ce point.

3.1.4 Observation partielle : ajout de $\circlearrowleft$ et Loop Mu-calcul quantifié

Dans le cadre d'un problème de contrôle avec observation partielle, on effectue une partition de l'ensemble des événements Σ en un ensemble d'événements observables Σ_o et un ensemble d'événements inobservables Σ_{uo}. La particularité de ces derniers est que l'on ignore s'ils ont lieu ainsi que leur fréquence le cas échéant.

Le processus étiquetant sous observation partielle le plus simple pour résoudre cette contrainte *boucle* sur les événements inobservables. Ainsi, si on est dans dans un état p pour le système à contrôler et que l'état courant pour le processus étiquetant est q, tous les états atteignables par un événement inobservable à partir de p seront étiquetés par produit synchrone grâce aux propositions atomiques de l'état q pour lequel on boucle sur les événements inobservables.

Cependant on ne peut pas exprimer en Mu-calcul quantifié le fait que l'on boucle sur un état du système. On contourne cette difficulté par l'ajout, dans la logique, de formules atomiques $\circlearrowleft^a$ $(a \in \Sigma)$ dont la sémantique sur les processus est la suivante :

$$[\![\circlearrowleft^a]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid t(s, a) = s\}$$

Remarque : On note par $\emptyset^a$ la formule $\neg \circlearrowleft^a$.

On retrouve cette extension dans [AVW03] où est définie la logique du Loop Mu-calcul et dans [Rie03] et [RP03a] avec le Loop Mu-calcul quantifié dont la syntaxe est la suivante :

Définition 12 (Syntaxe du Loop Mu-calcul quantifié)
L'ensemble des formules du Loop Mu-calcul quantifié sur $\Gamma \subseteq AP$, *noté* $\mathrm{Q}^{\circlearrowleft} L_\mu(\Gamma)$ *est défini par la grammaire :*

$$\exists[\Lambda \in \beta^{\circlearrowleft}].\alpha \mid \neg\alpha_1 \mid \alpha_1 \vee \alpha_2 \mid \beta$$

avec $\Lambda \subseteq AP$ *disjoint de* Γ, $\beta^{\circlearrowleft}$ *est une formule de Loop Mu-calcul sur* Λ, α *est une formule de* $QL_\mu(\Gamma \cup \Lambda)$, α_1 *et* α_2 *formules de* $QL_\mu(\Gamma)$ *et* β *est une formule du Mu-calcul sur* Γ.

On remarque que l'on ne peut avoir des boucles que dans la formule $\beta^{\circlearrowleft}$ qui caractérise le processus étiquetant.

Définition 13 (Sémantique du Loop Mu-calcul quantifié)
La sémantique des quantifications $\exists[\Lambda \in \beta^{\circlearrowleft}]$ *interprétées par rapport à un système non-déterministe* $\mathcal{S} =< \Gamma, S, s^0, t, L >$ *et à la fonction de valuation* $val : Var \to \mathcal{P}(S)$ *est :*

$$[\![\exists[\Lambda \in \beta^{\circlearrowleft}].\alpha]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists \mathcal{E} =< \Lambda, E, \epsilon^0, t', L' >\in Lab_\Lambda,$$
$$(s, \epsilon^0) \in [\![\alpha]\!]_{\mathcal{S}\times\mathcal{E}}^{[val\times E]} \wedge \epsilon^0 \in [\![\beta^{\circlearrowleft}]\!]_{\mathcal{E}}^{[val']}\}$$

avec $(val \times E)(X) = val(X) \times E$ *et* $val' : Var \to \mathcal{P}(E)$ *pour tout* $X \in Var$.

Comme indiqué précédemment, un problème de contrôle est ramené à du *model-checking* de formules de Mu-calcul quantifié. Nous définissons donc une fonction de traduction Tr d'une formule de Mu-calcul quantifié exprimée sur un système non-déterministe $\mathcal{S}$ en une formule sur le processus $\mathcal{S}^\tau$ correspondant au système non-déterministe codé grâce à la fonction de codage $cod_<$ définie dans le paragraphe 2.3.:

3.2 Traduction d'un problème de contrôle d'un système non-déterministe

3.2.1 Politique du "tout ou rien"

Supposons que le système à contrôler soit dans un état p et que, depuis cet état, il y ait plusieurs transitions sortantes étiquetées par l'événement a. Nous avons décidé que l'on devait adopter, vis-à-vis de ces cas de non-déterminisme, une politique du "tout ou rien", à savoir, soit on interdit toutes les transitions étiquetées a depuis s, soit on les autorise toutes. Cette politique semble conforme à l'idée de problème de contrôle où l'on souhaite avoir un oracle qui définit les événements autorisés à partir de la seule connaissance de l´état du système.

C'est pour respecter ce principe que l'on impose à l'étiquetage d'un système non-déterministe d´être déterministe et donc d'être effectué par un processus.

3.2.2 Conditions supplémentaires dues au codage des systèmes non-déterministes

Le codage des systèmes non-déterministes en processus proposé dans le paragraphe 2.3 et récapitulé dans la figure 3.2 s'accompagne de l'ajout d'une action interne τ. Le status de cet événement est le suivant :

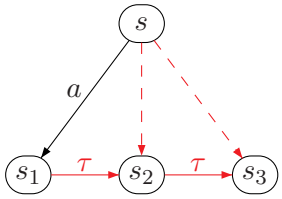


FIG. 3.2 – *Codage des cas de non-déterminisme.*

- τ est incontrôlable : on ne peut pas couper les transitions étiquetées par un τ afin de respecter la politique du "tout ou rien" définie dans le paragraphe 3.2.1. Si on souhaite empêcher l'événement a d'avoir lieu, on synthétise un processus étiquetant tel qu'on n'ajoutera pas la proposition atomique x_a au sommet s interdisant ainsi l'accès à tous les a-successeurs de s dans le système non-déterministe initial (les états s_1, s_2 et s_3) ;
- τ est inobservable : on parcourt l'ensemble des a-successeurs de manière transparente pour le processus étiquetant.

Ainsi, un problème de contrôle d'un système non-déterministe se réduit en un problème de contrôle d'un processus *avec observation partielle* car l'action interne τ introduite par le codage est inobservable.

3.2.3 Traduction des énoncés logiques des problèmes de contrôle

La seule modification, induite par le codage des systèmes non-déterministes, concerne les modalités de la forme $\langle a \rangle \varphi$ qui sont vraies pour les états qui ont un successeur par l'événement a qui vérifie φ. Compte tenu de la façon dont on traite les cas de non-déterminisme, ces modalités deviennent vraies pour les états qui ont un successeur par l'événement a suivi d'une chaîne finie de τ qui vérifie φ.

Cas des formules de Mu-calcul

Définition 14

Soit φ une formule de L_μ, Tr *est définie de manière inductive sur la structure de φ :*

$$\mathrm{Tr}(\top) = \top \qquad \mathrm{Tr}(p) = p$$
$$\mathrm{Tr}(X) = X \qquad \mathrm{Tr}(\langle a \rangle \varphi_1) = \langle a \rangle (\mu X. \langle \tau \rangle X \vee \mathrm{Tr}(\varphi_1))$$
$$\mathrm{Tr}(\neg \varphi_1) = \neg \mathrm{Tr}(\varphi_1) \qquad \mathrm{Tr}(\varphi_1 \vee \varphi_2) = \mathrm{Tr}(\varphi_1) \vee \mathrm{Tr}(\varphi_2)$$
$$\mathrm{Tr}(\mu X.\varphi_1(X)) = \mu X.\mathrm{Tr}(\varphi_1(X))$$

Notation : La formule $\langle a \rangle (\mu X. \langle \tau \rangle X \vee \varphi)$ sera par la suite notée $\langle a\tau^* \rangle \varphi$.

Ainsi, on a : $\mathrm{Tr}(\langle a \rangle \varphi) = \langle a\tau^* \rangle \mathrm{Tr}(\varphi)$. Sémantiquement, cette notation est correcte en tenant compte du lemme suivant :

Lemme 1
Soit $\mathcal{S}$ un processus quelconque, on a :

$$[\![\langle a \rangle (\mu X. \langle \tau \rangle X \vee \varphi)]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists n \in \mathbb{N}, \exists s' : t(s, a\tau^n) = s' \text{ et } s' \in [\![\varphi]\!]_{\mathcal{S}}^{[val]}\}.$$

Démonstration
Par définition, on a :

$$[\![\langle a\tau^* \rangle \varphi]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists n \in \mathbb{N}, \exists s' : t(s, a\tau^n) = s' \text{ et } s' \in [\![\varphi]\!]_{\mathcal{S}}^{[val]}\}.$$

Ainsi un état s satisfait $\langle a\tau^* \rangle \varphi$ s'il existe un chemin fini du processus partant de s (commençant par un a), étant suivi d'une séquence finie de τ et atteignant un état satisfaisant la formule φ. En logique CTL, ceci s'écrit : EF φ. *Or, d'après l'étude de l'expressivité du Mu-calcul qui figure dans [Rie03], on a :*

$$EFp = \mu X. \bigvee_{a \in \Sigma} \langle a \rangle X \vee p$$

Appliqué à notre contexte, ceci donne :

$$[\![\langle a\tau^* \rangle \varphi]\!]_{\mathcal{S}}^{[val]} = [\![\langle a \rangle (\mu X. \langle \tau \rangle X \vee \varphi)]\!]_{\mathcal{S}}^{[val]}$$

D'où le résultat énoncé :

$$[\![\langle a \rangle (\mu X. \langle \tau \rangle X \vee \varphi)]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists n \in \mathbb{N}, \exists s' : t(s, a\tau^n) = s' \text{ et } s' \in [\![\varphi]\!]_{\mathcal{S}}^{[val]}\}.$$

♢

Lemme 2
Pour toute formule $\varphi \in L_\mu$, on a :

$$[\![\varphi]\!]_{\widetilde{\mathcal{S}}}^{[val]} = [\![\mathrm{Tr}(\varphi)]\!]_{\mathcal{S}^\tau}^{[val]}$$

Démonstration
On prend $\widetilde{\mathcal{S}} = \mathcal{T}_1(\mathcal{S}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L} >$ et $\mathcal{S}^\tau = \mathcal{T}_{2,<}(\widetilde{\mathcal{S}}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, t^\tau, \widetilde{L} >$. La preuve procède par induction sur φ.
Étant donné que les processus $\widetilde{\mathcal{S}}$ et $\mathcal{S}^\tau$ ne différent que par leur fonction de transition, on a directement :

$$\begin{array}{lllll}
[\![\mathrm{Tr}(\top)]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![\top]\!]_{\mathcal{S}^\tau}^{[val]} & = & \widetilde{S} & = & [\![\top]\!]_{\widetilde{\mathcal{S}}}^{[val]} \\
[\![\mathrm{Tr}(X)]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![X]\!]_{\mathcal{S}^\tau}^{[val]} & = & val(X) & = & [\![X]\!]_{\widetilde{\mathcal{S}}}^{[val]} \\
[\![\mathrm{Tr}(p)]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![p]\!]_{\mathcal{S}^\tau}^{[val]} & = & \{s \in S \mid p \in \widetilde{L}(s)\} & = & [\![p]\!]_{\widetilde{\mathcal{S}}}^{[val]}
\end{array}$$

De plus, avec l'hypothèse d'induction $[\![\mathrm{Tr}(\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} = [\![\varphi_1]\!]_{\widetilde{\mathcal{S}}}^{[val]}$, on a :

$$
\begin{array}{lcl}
[\![\mathrm{Tr}(\neg\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![\neg\mathrm{Tr}(\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} \\
& = & \widetilde{S} \setminus [\![\mathrm{Tr}(\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} \\
& = & \widetilde{S} \setminus [\![\varphi_1]\!]_{\widetilde{S}}^{[val]} \\
& = & [\![\neg\varphi_1]\!]_{\widetilde{S}}^{[val]}
\end{array}
$$

$$
\begin{array}{lcl}
[\![\mathrm{Tr}(\mu X.\varphi_1(X))]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![\mu X.\mathrm{Tr}(\varphi_1(X))]\!]_{\mathcal{S}^\tau}^{[val]} \\
& = & \cap\{V \subseteq \widetilde{S} \mid [\![\mathrm{Tr}(\varphi_1(X))]\!]_{\mathcal{S}^\tau}^{[val(V/X]} \subseteq V\} \\
& = & \cap\{V \subseteq \widetilde{S} \mid [\![\varphi_1(X)]\!]_{\widetilde{S}}^{[val(V/X]} \subseteq V\} \\
& = & [\![\mu X.\varphi_1(X)]\!]_{\widetilde{S}}^{[val]}
\end{array}
$$

$$
\begin{array}{lcl}
[\![\mathrm{Tr}(\langle a\rangle\, \varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} & = & [\![\langle a\tau^*\rangle\, \mathrm{Tr}(\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} \\
& = & \{s \in \widetilde{S} \mid \exists s' : t^\tau(s, a\tau^*) = s' \text{ et } s' \in [\![\mathrm{Tr}(\varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]}\} \\
& = & \{s \in \widetilde{S} \mid \exists s' : t^\tau(s, a\tau^*) = s' \text{ et } s' \in [\![\varphi_1]\!]_{\widetilde{S}}^{[val]}\}
\end{array}
$$

On distingue deux cas en fonction du nombre de $a-$successeurs d'un sommet s dans le processus $\mathcal{S}$ (quelques soient l'événement a et le sommet s) :

– *cas $|Succ(s,a)| \leq 1$:*
par construction, $\widetilde{t}(s,a) = t^\tau(s,a)$, et :

$$
\begin{array}{lcll}
t^\tau(s, a\tau^*) & = & t^\tau(t^\tau(s,a), \tau^*) & \\
& = & t^\tau(t^\tau(s,a), \epsilon) & \text{car il n'y a pas de } \tau-\text{transition} \\
& = & t^\tau(s,a) & \\
& = & \widetilde{t}(s,a) &
\end{array}
$$

– *cas $|Succ(s,a)| > 1$:*

$$
\begin{array}{lcl}
s' = t^\tau(s, a\tau^*) & \Leftrightarrow & \exists\, i : s' = t^\tau(s, a\tau^i) \\
& \Leftrightarrow & s' \in Succ(s,a) \text{ par construction de } t^\tau \text{ et} \\
& & \text{étant donné que } \mathcal{S}^\tau \text{ est déterministe} \\
& \Leftrightarrow & s' = \widetilde{t}(s,a)
\end{array}
$$

d'où : $t^\tau(s, a\tau^*) = \widetilde{t}(s,a)$

On en déduit finalement :

$$
\begin{array}{lcl}
[\![\mathrm{Tr}(\langle a\rangle\, \varphi_1)]\!]_{\mathcal{S}^\tau}^{[val]} & = & \{s \in \widetilde{S} \mid \exists s' : t^\tau(s, a\tau^*) = s' \text{ et } s' \in [\![\varphi_1]\!]_{\widetilde{S}}^{[val]}\} \\
& = & \{s \in \widetilde{S} \mid \exists s' : \widetilde{t}(s,a) = s' \text{ et } s' \in [\![\varphi_1]\!]_{\widetilde{S}}^{[val]}\} \\
& = & [\![\langle a\rangle\, \varphi_1]\!]_{\widetilde{S}}^{[val]}
\end{array}
$$

Enfin, avec l'hypothèse d'induction supplémentaire $[\![\mathrm{Tr}(\varphi_2)]\!]_{\mathcal{S}^\tau}^{[val]} = [\![\varphi_2]\!]_{\widetilde{S}}^{[val]}$, on a :

$$\begin{aligned}
[\![\mathrm{Tr}(\varphi_1 \vee \varphi_2)]\!]^{[val]}_{\mathcal{S}^\tau} &= [\![\mathrm{Tr}(\varphi_1) \vee \mathrm{Tr}(\varphi_2)]\!]^{[val]}_{\mathcal{S}^\tau} \\
&= [\![\mathrm{Tr}(\varphi_1)]\!]^{[val]}_{\mathcal{S}^\tau} \cup [\![\mathrm{Tr}(\varphi_2)]\!]^{[val]}_{\mathcal{S}^\tau} \\
&= [\![\varphi_1]\!]^{[val]}_{\widetilde{\mathcal{S}}} \cup [\![\varphi_2]\!]^{[val]}_{\widetilde{\mathcal{S}}} \\
&= [\![\varphi_1 \vee \varphi_2]\!]^{[val]}_{\widetilde{\mathcal{S}}}
\end{aligned}$$

♢

Théorème 1
Soit $\mathcal{S}$ un système non-déterministe, pour toute formule $\varphi \in L_\mu$, on a :

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<\mathcal{S})}$$

Démonstration
D'après la propriété 1, les systèmes $\mathcal{S}$ et $\widetilde{\mathcal{S}} = \mathcal{T}_1(\mathcal{S})$ sont bisimilaires. Le Mu-calcul étant invariant par bisimilarité, on a :

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\varphi]\!]^{[val]}_{\widetilde{\mathcal{S}}}$$

Or, d'après le lemme précédent, on a :

$$[\![\varphi]\!]^{[val]}_{\widetilde{\mathcal{S}}} = [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{\mathcal{S}^\tau}$$

D'où :

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{\mathcal{S}^\tau}$$

♢

Cas des formules de Mu-calcul quantifié

Le status de l'action τ induit une traduction non triviale de la quantification aux propositions atomiques contenues dans le Mu-calcul quantifié. Étant donné que τ est inobservable, on impose au processus étiquetant spécifié de boucler sur l'événement τ. De plus, comme τ est aussi incontrôlable, on ajoute x_τ à tous les sommets afin de ne jamais interdire cet événement.

Définition 15
Soit φ une formule de $\mathrm{Q}L_\mu$, Tr *est définie de manière inductive sur la structure de φ :*

$\mathrm{Tr}(\exists X.\alpha) = \exists X \cup \{x_\tau\} \in \mathit{Loop}(\tau).\mathrm{Tr}(\alpha)$
$\mathrm{Tr}(\neg\varphi_1) = \neg\mathrm{Tr}(\varphi_1)$
$\mathrm{Tr}(\varphi_1 \vee \varphi_2) = \mathrm{Tr}(\varphi_1) \vee \mathrm{Tr}(\varphi_2)$
avec $\mathit{Loop}(\tau) = AG(x_\tau \wedge \circlearrowleft^\tau)$.

Si φ est une formule de Mu-calcul, on se réfère à la fonction de traduction définie pour les formules de Mu-calcul.

Complexité : La taille de la formule n'est pas modifiée par la transformation Tr :

$$|\mathrm{Tr}(\varphi)| = O(|\varphi|).$$

Pour ce qui est dans la classe $Loop(\tau)$ du processus étiquetant, la formule qui la caractérise ne dépend ni de $\mathcal{S}$, ni de φ et est constante, sa taille est donc en $O(1)$.

Il faut noter que l'on demande à tous les états du processus étiquetant de boucler sur τ et d'avoir la proposition atomique x_τ. Ainsi les transitions étiquetées par τ sont toujours autorisées dans le processus étiqueté. Considérons le cas suivant :

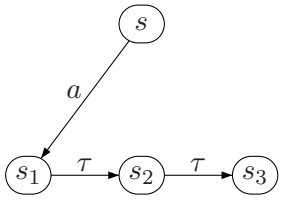


FIG. 3.3 – *Processus codant un système non-déterministe.*

Par produit synchrone, la proposition atomique x_τ appartiendra à $L(s_1)$ et $L(s_2)$ autorisant ainsi la τ-transition sortante des états s_1 et s_2 mais l'accès à ces a-successeurs de s dans le système non-déterministe initial sera conditionné par la présence ou non de x_a en amont, dans $L(s)$.

Avant de prouver la correction de la fonction de traduction définie en 15, nous allons étudier la correspondance entre le processus étiquetant solution d'un problème de contrôle sur un système non-déterministe et le processus étiquetant solution du problème de contrôle d'un processus correspondant au système déterministe codé :

Définition 16
Soit $\mathcal{E}^{\circlearrowleft}$ un processus étiquetant appartenant à $Loop(\tau)$, par application de la fonction bijective l, on obtient de la façon suivante le processus étiquetant $\mathcal{E} = l(\mathcal{E}^{\circlearrowleft})$:

- *pour tout état s, si $t_{\mathcal{E}^{\circlearrowleft}}(s,a) = s'$ et $a \neq \tau$ alors $t_{\mathcal{E}}(s,a) = s'$;*
- *pour tout état s, $L_{\mathcal{E}}(s) = L_{\mathcal{E}^{\circlearrowleft}}(s) \setminus \{x_\tau\}$*

Remarque : Si $\mathcal{E}^{\circlearrowleft}$ est un processus étiquetant sur $X \cup \{x_\tau\}$ alors $\mathcal{E}$ est un processus étiquetant sur X. De manière condensée, la signature de l est donc :

$$l : Lab_{X\cup\{x_\tau\}} \in Loop(\tau) \rightarrow Lab_X$$

Propriété 4
La fonction l vérifie la propriété suivante :

$$l(\mathcal{E}^{\circlearrowleft} \times \mathcal{E}'^{\circlearrowleft}) = l(\mathcal{E}^{\circlearrowleft}) \times l(\mathcal{E}'^{\circlearrowleft}) = \mathcal{E} \times \mathcal{E}'.$$

Démonstration
Soient $\mathcal{E}^{\circlearrowleft} =< X \cup \{x_\tau\}, E, \epsilon^0, t_{\mathcal{E}^{\circlearrowleft}}, L_{\mathcal{E}^{\circlearrowleft}} >$ et $\mathcal{E}'^{\circlearrowleft} =< X \cup \{x_\tau\}, E', \epsilon'^0, t_{\mathcal{E}'^{\circlearrowleft}}, L_{\mathcal{E}'^{\circlearrowleft}} >$, on pose $\mathcal{S}_1$ et$\mathcal{S}_2$ respectivement les processus $l(\mathcal{E}^{\circlearrowleft} \times \mathcal{E}'^{\circlearrowleft})$ et $l(\mathcal{E}^{\circlearrowleft}) \times l(\mathcal{E}'^{\circlearrowleft})$. Prouvons que $\mathcal{S}_1$ et$\mathcal{S}_2$ sont identiques :

D'abord, $\mathcal{S}_1$ et$\mathcal{S}_2$ ont le même ensemble d'états inclus dans $E \times E'$ et le même état initial $(\epsilon^0, \epsilon'^0)$.

De même, par produit synchrone, pour tout état (ϵ, ϵ') des processus $\mathcal{S}_1$ et $\mathcal{S}_2$, l'ensemble des propositions atomiques associé est : $(L_{\mathcal{E}^\circlearrowleft}(\epsilon) \cup L_{\mathcal{E}'^\circlearrowleft}(\epsilon')) \setminus \{x_\tau\}$.

Enfin pour la fonction de transition, on a (avec $a \neq \tau$) :

$$\begin{array}{rcl} t_{\mathcal{S}_2}((\epsilon_1, \epsilon'_1), a) = (\epsilon_2, \epsilon'_2) & \Leftrightarrow & t_{l(\mathcal{E}^\circlearrowleft)}(\epsilon_1, a) = \epsilon_2 \text{ et } t_{l(\mathcal{E}'^\circlearrowleft)}(\epsilon'_1, a) = \epsilon'_2 \\ & \Leftrightarrow & t_{\mathcal{E}^\circlearrowleft}(\epsilon_1, a) = \epsilon_2 \text{ et } t_{\mathcal{E}^\circlearrowleft}(\epsilon_1, \tau) = \epsilon_1 \\ & & \text{et } t_{\mathcal{E}'^\circlearrowleft}(\epsilon'_1, a) = \epsilon'_2 \text{ et } t_{\mathcal{E}'^\circlearrowleft}(\epsilon'_1, \tau) = \epsilon'_1 \\ & \Leftrightarrow & t_{\mathcal{E}^\circlearrowleft \times \mathcal{E}'^\circlearrowleft}((\epsilon_1, \epsilon'_1), a) = (\epsilon_2, \epsilon'_2) \\ & & \text{et } t_{\mathcal{E}^\circlearrowleft \times \mathcal{E}'^\circlearrowleft}((\epsilon_1, \epsilon'_1), \tau) = (\epsilon_1, \epsilon'_1) \\ & \Leftrightarrow & t_{\mathcal{S}_1}((\epsilon_1, \epsilon'_1), a) = (\epsilon_2, \epsilon'_2) \end{array}$$

Donc $\mathcal{S}_1$ et $\mathcal{S}_2$ ont la même fonction de transition et les processus $l(\mathcal{E}^\circlearrowleft \times \mathcal{E}'^\circlearrowleft)$ et $l(\mathcal{E}^\circlearrowleft) \times l(\mathcal{E}'^\circlearrowleft)$ sont identiques.

♢

Pour prouver la correction de la traduction proposée dans la définition 15, on utilise la propriété suivante :

Propriété 5 (Équivalence des solutions)
$\mathcal{E}^\circlearrowleft \in Loop(\tau) \cap Lab_{X \cup \{x_\tau\}}$ est un processus étiquetant pour le processus $\mathcal{S}^\tau = cod_<(\mathcal{S})$ et pour la sentence $\mathrm{Tr}(\alpha) \in \mathrm{Q}^\circlearrowleft L_\mu$ si et seulement si $\mathcal{E} = l(\mathcal{E}^\circlearrowleft)$ est un processus étiquetant sur X pour le système non-déterministe $\mathcal{S}$ et la sentence $\alpha \in \mathrm{Q}L_\mu$.

Pour prouver la proposition précédente, nous avons besoin du lemme suivant :

Lemme 3
Les processus $cod_<(\mathcal{S}) \times \mathcal{E}^\circlearrowleft$ et $cod_<(\mathcal{S} \times \mathcal{E})$ sont bisimilaires.

Démonstration
D'après la proposition 3, on a :

- *d'une part :* $cod_<(\mathcal{S} \times \mathcal{E}) \leftrightarrows T^\tau_{\mathcal{S} \times \mathcal{E}, <}$
- *et d'autre part :* $cod_<(\mathcal{S}) \leftrightarrows T^\tau_{\mathcal{S}, <}$
  ce qui est équivaut à : $cod_<(\mathcal{S}) \times \mathcal{E}^\circlearrowleft \leftrightarrows T^\tau_{\mathcal{S}, <} \times \mathcal{E}^\circlearrowleft$.

Montrons que $T^\tau_{\mathcal{S} \times \mathcal{E}, <} \leftrightarrows T^\tau_{\mathcal{S}, <} \times \mathcal{E}^\circlearrowleft$ (en fait ces arbres sont isomorphes) :

Soit $last(stateseq(w, \mathcal{T}))$ le dernier état de $\mathcal{T}$ atteint après l'exécution de la séquence $w \in (\Sigma \cup \{\tau\})^*$. On a :

$$last(stateseq(w, T^\tau_{\mathcal{S} \times \mathcal{E}, <})) = (s', t_{\mathcal{E}^\circlearrowleft}(\epsilon^0, w))$$
$$\text{avec } s' \text{ l'état tel que } s'_{(s,a)} = t^\tau(s^0, w)$$

On vérifie ce résultat par récurrence sur la longueur $|w|$ de w :

- ***cas*** $|w| = 0$ ***:***
  D'une part : $last(stateseq(\epsilon, T^\tau_{\mathcal{S} \times \mathcal{E}, <})) = (s^0, \epsilon^0)$
  D'autre part : $t^\tau(s^0, \epsilon) = \widetilde{s^0} = s^0$ *et* $t_{\mathcal{E}^\circlearrowleft}(\epsilon^0, \epsilon)) = \epsilon^0$

– ***cas*** $|w| > 0$ ***:***
supposons que le résultat soit vrai pour tout $|w| \leq k$*. On considère les cas où* $w' = wb$ *(avec* $b \neq \tau$*) et* $w' = w\tau$ *:*

– ***cas*** $w' = wb$ ***(avec*** $b \neq \tau$***)***
D'une part : $last(stateseq(wb, T^{\tau}_{\mathcal{S}\times\mathcal{E},<})) = (t(s', b), t_{\mathcal{E}^{\circlearrowleft}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, w), b)$
D'autre part :

$$\begin{array}{lcll} t^{\tau}(s'_{(s,a)}, b) & = & t(s', b)_{(s',b)} & \text{par définition de } t^{\tau} \text{ et } \widetilde{t} \\ t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, wb) & = & t_{\mathcal{E}^{\circlearrowleft}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, w), b) & \\ & = & t_{\mathcal{E}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, w), b) & \text{car } b \neq \tau \text{ et que } \mathcal{E} \text{ et } \mathcal{E}^{\circlearrowleft} \\ & & & \text{ne différent que par des boucles de } \tau \end{array}$$

– ***cas*** $w' = w\tau$
Nécessairement, par codage, $w' = w\tau = vb\tau^i.\tau$*. Comme* $|v| < |w'|$*, on a :*

$$last(stateseq(v, T^{\tau}_{\mathcal{S}\times\mathcal{E},<})) = (s', t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, v))$$
$$\text{avec } s'_{(s,a)} = t^{\tau}(s^0, v)$$

Dans $\mathcal{S}$*, depuis* s'*, on tire un* b*; par codage et compte tenu de la séquence de* τ*, on arrive dans l'état, successeur d'ordre* i *dans* $Succ(s', b)$*.*
Dans $\mathcal{E}$*, on ne voit la séquence de* τ*, on arrive dans l'état* $t_{\mathcal{E}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, v), b)$*.*
De plus :

$$\begin{array}{lcl} t^{\tau}(s^0, vb\tau^i.\tau) & = & t^{\tau}(t^{\tau}(s^0, v), b\tau^i.\tau) \\ & = & t^{\tau}(s'_{(s,a)}, b\tau^i.\tau) \\ & = & s_{i(s',b)} \end{array}$$

avec s_i *le successeur d'ordre* i *dans* $Succ(s', b)$*.*
Enfin :

$$\begin{array}{lcll} t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, vb\tau^i.\tau) & = & t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, vb) & \text{car } \mathcal{E}^{\circlearrowleft} \text{ boucle sur les } \tau \\ & = & t_{\mathcal{E}^{\circlearrowleft}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, v), b) & \\ & = & t_{\mathcal{E}}(t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, v), b) & \text{car } b \neq \tau \end{array}$$

Ainsi, on a :

$$\begin{array}{lcl} last(stateseq(w, T^{\tau}_{\mathcal{S}\times\mathcal{E},<})) & = & (s', t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, w)) \\ & & \text{avec } s' \text{ l'état tel que } s'_{(s,a)} = t^{\tau}(s^0, w) \\ & = & (last(stateseq(w, T^{\tau}_{\mathcal{S},<})), t_{\mathcal{E}^{\circlearrowleft}}(\epsilon^0, w)) \\ & & \text{d'après la démonstration de la prop. 3} \end{array}$$

par définition du produit synchrone, ces couples sont le résultat du produit cartésien des états de $T_{\mathcal{S},<}$ *et* $\mathcal{E}^{\circlearrowleft}$ *d'où :* $T^{\tau}_{\mathcal{S}\times\mathcal{E},<} = T^{\tau}_{\mathcal{S},<} \times \mathcal{E}^{\circlearrowleft}$ *et donc :* $cod_<(\mathcal{S}\times\mathcal{E}) \leftrightarrows cod_<(\mathcal{S}) \times \mathcal{E}^{\circlearrowleft}$*.*

♢

Nous pouvons maintenant démontrer la proposition 5.

Démonstration
On souhaite démontrer que $\mathcal{S}^{\tau} \times \mathcal{E}^{\circlearrowleft} \models \mathrm{Tr}(\alpha)$ *si et seulement si* $\mathcal{S} \times \mathcal{E} \models \alpha$*. On procède par induction sur la structure de* $\alpha \in \mathrm{Q}L_{\mu}$*.*

– ***si*** α ***est une formule du Mu-calcul pur :***

$$\begin{array}{lcll} [\![\varphi]\!]^{[val]}_{\mathcal{S}\times\mathcal{E}} & = & [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<(\mathcal{S}\times\mathcal{E})} & \textit{d'après le th. 1} \\ [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<(\mathcal{S}\times\mathcal{E})} & = & [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<(\mathcal{S})\times\mathcal{E}^{\circlearrowleft}} & \textit{car } cod_<(\mathcal{S}\times\mathcal{E}) \leftrightarrows cod_<(\mathcal{S})\times\mathcal{E}^{\circlearrowleft} \\ & & & \textit{d'après le lemme 3} \end{array}$$

d'où $[\![\varphi]\!]^{[val]}_{\mathcal{S}\times\mathcal{E}} = [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<(\mathcal{S})\times\mathcal{E}^{\circlearrowleft}}$.

– ***cas de la quantification*** $\alpha = \exists X'.\alpha'$ **:**

$$\begin{array}{lcl} \mathcal{S}^{\tau}\times\mathcal{E}^{\circlearrowleft} \models \mathrm{Tr}(\exists X'.\alpha') & \Leftrightarrow & \mathcal{S}^{\tau}\times\mathcal{E}^{\circlearrowleft} \models \exists X'\cup\{x_\tau\}\in \mathit{Loop}(\tau).\mathrm{Tr}(\alpha') \\ & \Leftrightarrow & \exists\mathcal{E}'^{\circlearrowleft}\in \mathit{Lab}_{X'\cup\{x_\tau\}} : \\ & & (\mathcal{S}^{\tau}\times\mathcal{E}^{\circlearrowleft})\times\mathcal{E}'^{\circlearrowleft} \models \mathrm{Tr}(\alpha')\wedge\mathcal{E}'^{\circlearrowleft}\models \mathit{Loop}(\tau) \\ & \Leftrightarrow & \mathcal{S}^{\tau}\times(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}) \models \mathrm{Tr}(\alpha') \\ & & \wedge\ \mathcal{E}'^{\circlearrowleft}\models \mathit{Loop}(\tau)\ \textit{et,} \\ & & \textit{par hypothèse, } \mathcal{E}^{\circlearrowleft}\models \mathit{Loop}(\tau) \\ & \Leftrightarrow & \mathcal{S}^{\tau}\times(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}) \models \mathrm{Tr}(\alpha')\wedge\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}\models \mathit{Loop}(\tau) \\ & & \textit{par définition du produit synchrone} \\ & \Leftrightarrow & \mathcal{S}\times l(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft})\models\alpha' \\ & & \textit{d'après l'hypothèse d'induction} \\ & \Leftrightarrow & \mathcal{S}\times l(\mathcal{E}^{\circlearrowleft})\times l(\mathcal{E}'^{\circlearrowleft})\models\alpha'\ \textit{d'après la prop. 4} \\ & \Leftrightarrow & \mathcal{S}\times\mathcal{E}\times\mathcal{E}'\models\alpha' \\ & \Leftrightarrow & \exists\mathcal{E}'\in \mathit{Lab}_{X'} : (\mathcal{S}\times\mathcal{E})\times\mathcal{E}'\models\alpha' \\ & \Leftrightarrow & \mathcal{S}\times\mathcal{E}\models\exists X'.\alpha' \end{array}$$

– *Les cas de la négation* $\alpha = \neg\alpha_1$ *et de la disjonction* $\alpha = \alpha_1\vee\alpha_2$ *sont évidents.*

♢

Compte tenu de la propriété 5, nous pouvons démontrer la correction de la traduction des formules de Mu-calcul quantifié proposée précédemment :

Théorème 2
Soit $\mathcal{S}$ *un système non-déterministe, pour toute formule* $\varphi\in \mathrm{Q}L_\mu$*, on a :*

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\mathrm{Tr}(\varphi)]\!]^{[val]}_{cod_<(\mathcal{S})}$$

Démonstration
On procède par induction sur la structure de $\varphi\in \mathrm{Q}L_\mu$*. Examinons le cas de la quantification :*

$$\begin{array}{lcl} [\![\exists X.\alpha]\!]^{[val]}_{\mathcal{S}} & = & \{s\in S : \exists\mathcal{E}\in \mathit{Lab}_X,\ (s,\epsilon^0)\in [\![\alpha]\!]^{[val\times E]}_{\mathcal{S}\times\mathcal{E}}\} \\ & = & \{s_{(r,a)}\in\widetilde{S} : \exists\mathcal{E}^{\circlearrowleft}\in \mathit{Lab}_{X\cup\{x_\tau\}},\ (s_{(r,a)},\epsilon^0)\in [\![\mathrm{Tr}(\alpha)]\!]^{[val\times E]}_{\mathcal{S}^{\tau}\times\mathcal{E}^{\circlearrowleft}} \\ & & \textit{avec } \mathcal{E}^{\circlearrowleft}\models \mathit{Loop}(\tau)\} \\ & = & [\![\exists X\cup\{x_\tau\}\in \mathit{Loop}(\tau).\mathrm{Tr}(\alpha)]\!]^{[val]}_{\mathcal{S}^{\tau}} \end{array}$$

Les autres cas sont évidents, ils se déduisent directement en revenant à la définition de la sémantique de la logique et en utilisant l'hypothèse d'induction.

♢

3.2.4 Décidabilité d'un problème de contrôle d'un système non-déterministe

A partir des travaux de [Rie03] sur le model-checking des formules de Loop Mu-calcul quantifié, on peut énoncer le résultat suivant :

Théorème 3
Les problèmes de contrôle d'un système non-déterministe exprimés en Mu-calcul quantifié sont décidables.

Démonstration
Nous avons précédemment vu qu'un problème de contrôle d'un système non-déterministe se réduit en un problème de model-checking d'une formule de Loop Mu-calcul quantifié. Dans [Rie03] est énoncé un théorème sur la décidabilité de ce problème : [Rie03] se place dans le fragment des sentences de $\mathrm{Q}^{\circlearrowleft} L_\mu$ *appelé* $\mathrm{Q}^{\mathtt{Obs}} L_\mu$ *où la sous-expression* $[\Lambda \in \beta^{\circlearrowleft}]$ *avec* $\beta^{\circlearrowleft} \in L_\mu^{\circlearrowleft}$ *caractérisant le processus étiquetant se décompose en la conjonction d'une sentence* $\beta' \in L_\mu$ *et d'une sentence* $\mathtt{Obs}(\Sigma_o) = AG(\bigwedge_{uo \notin \Sigma_o} \circlearrowleft^{uo})$.

Les formules produites par la fonction de traduction Tr *s'inscrivent bien dans ce fragment en prenant* $\beta' = AG(x_\tau)$ *et* $\Sigma_o = \Sigma \setminus \{\tau\}$.

Ensuite, [Rie03] définit la forme normale des sentences de $\mathrm{Q}^{\mathtt{Obs}} L_\mu$ *sous laquelle* ***toutes*** *les sentences de* $\mathrm{Q}^{\mathtt{Obs}} L_\mu$ *peuvent être mises :*

$$Q_1[\Lambda \in \beta_1 \wedge \mathtt{Obs}(\Sigma_o^1)].Q_2[\Lambda \in \beta_2 \wedge \mathtt{Obs}(\Sigma_o^2)]. \ ... \ .Q_n[\Lambda \in \beta_n \wedge \mathtt{Obs}(\Sigma_o^n)].\beta$$

avec $Q_i \in \{\exists, \forall\}, n \geq 1$ *et :*

- $\forall i$, $\Sigma_o^i \neq \emptyset$;
- *pour* $1 < i < n$, *si* $Q_i = Q_{i+1}$ *alors on n'a pas* $\Sigma_o^{i+1} \subsetneq \Sigma_o^i$.

[Rie03] établit que le model-checking des sentences de $\mathrm{Q}^{\mathtt{Obs}} L_\mu$ *est décidable lorsque :*

$$\Sigma_o^1 \subseteq \Sigma_o^2 \subseteq ... \subseteq \Sigma_o^n.$$

Étant donné que τ *est le seul événement inobservable,* $\Sigma_o^i = \Sigma \setminus \{\tau\}$ *pour tout* i *et donc les problèmes de contrôles d'un système non-déterministe sont décidables.*

$\diamondsuit$

3.2.5 Récapitulatif : la démarche à suivre

Dans cette section, nous avons démontré la correction de la démarche suivante qui permet de résoudre un problème de contrôle d'un système non-déterministe $\mathcal{S}$ spécifié par la formule de Mu-calcul quantifié $\exists X.\varphi$:

1. convertir le système non-déterministe $\mathcal{S}$ en processus $\mathcal{S}^\tau$ grâce à la fonction de codage $cod_<$;
2. transformer l'énoncé logique du problème de contrôle $\exists X.\varphi$ grâce à la fonction Tr ;
3. résoudre le problème de contrôle d'un processus $\mathcal{S}^\tau \models \mathrm{Tr}(\exists X.\varphi)$ (cf. [Rie03]); on obtient un processus étiquetant $\mathcal{E}^{\circlearrowleft}$;
4. appliquer l à $\mathcal{E}^{\circlearrowleft}$ pour obtenir le processus étiquetant $\mathcal{E}$ solution au problème de contrôle $\exists X.\varphi$ d'un système non-déterministe $\mathcal{S}$.

Chapitre 4

Autre approche : utilisation des événements indistinguables

Dans cette section, nous proposons une autre façon de coder les systèmes déterministes en processus basée sur un renommage des événements possibles du système.

On introduit, de plus, un nouvel élément dans notre logique : le principe des *événements indistinguables* présent dans [ABW03]. Cette extension permet d'exprimer le fait que le contrôleur (ou le processus étiquetant) ne sache pas différencier deux événements a et b lorsqu'ils ont lieu dans le système. En d'autres mots, on désire que, indépendamment du fait que ce soit l´événement a ou b qui ait lieu, le contrôleur ait le même comportement. Cette propriété n'étant pas exprimable en Mu-calcul puisqu'elle n'est pas invariante par bisimulation, [ABW03] introduisent pour toute paire d'événements (a, b) un symbole propositionnel $\downdownarrows_{a,b}$ dont la sémantique sur un processus $\mathcal{S} =< \Gamma, S, s^0, t, L >$ est la suivante :

$$[\![\downdownarrows_{a,b}]\!]_{\mathcal{S}} = \{s \in S \mid \text{si } t(s,a) = s' \text{ et } t(s,b) = s'' \text{ alors } s' = s''\}$$

Dans cette partie nous allons dans un premier temps définir un codage des systèmes en processus par renommage des événements du système puis nous définirons les événements indistinguables du processus ainsi codé comme contrainte additionnelle de notre problème de contrôle.

4.1 Codage des systèmes non-déterministe en processus

On effectue un renommage des événements du système en apposant à l'étiquette de la transition l'état de destination. Ce principe est représenté dans la figure 4.1 suivante,

dans le cas d'un non-déterminisme en a.

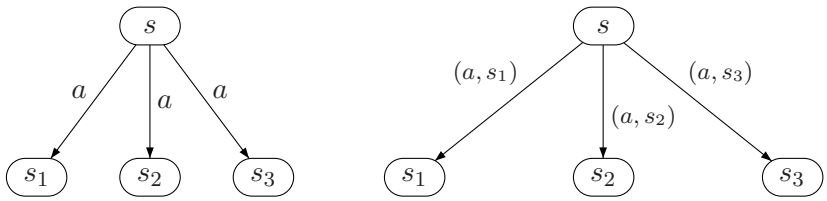


FIG. 4.1 – *Déterminisation par renommage.*

Ensuite, on considérera que les événements (a, s_1), (a, s_2) et (a, s_3) sont indistinguables pour le processus étiquetant. Ainsi une politique de contrôle unique sera décidée pour l'ensemble des a-transitions issues de s dans le système non-déterministe initial.

Le système obtenu est déterministe (ie. est un processus). Par la suite, on notera $cod_{\mathcal{S}}$ cette transformation et $\widehat{\mathcal{S}}$ le processus obtenu sur l'ensemble d'événements $\widehat{\Sigma} \subseteq \Sigma \times S$.

Complexité : pour évaluer la complexité de $cod_{\mathcal{S}}$ on considère la taille du système. Le nombre d'états du processus obtenu est égal au nombre d'état du système non-déterministe initial donc :

$$|\widehat{\mathcal{S}}| = O(|\mathcal{S}|).$$

4.2 Traduction des énoncés logiques des problèmes de contrôle

Comme dans le paragraphe 3.2.3, on définit une fonction de traduction $\mathrm{Tr}_{\mathcal{S}}$ permettant de passer de l'énoncé d'un problème de contrôle d'un système non-déterministe $\mathcal{S}$ à un problème de contrôle d'un processus $\widehat{S} = cod_{\mathcal{S}}(\mathcal{S})$. On considérera, dans un premier temps, le cas des formules de Mu-calcul. Ensuite, nous verrons celui des formules de Mu-calcul quantifié. La logique cible sera alors le $\mathrm{Q}^{\Downarrow} L_\mu$, une extension du Mu-calcul comprenant l'opérateur de quantification $\exists$ et les propositions atomiques du type $\downarrow\!\!\downarrow_{a,b}$.

Définition 17
Soit φ une formule de L_μ, $\mathrm{Tr}_{\mathcal{S}}$ est définie de manière inductive sur la structure de φ :

$\mathrm{Tr}_{\mathcal{S}}(\top) = \top$ $\quad$ $\mathrm{Tr}_{\mathcal{S}}(p) = p$
$\mathrm{Tr}_{\mathcal{S}}(X) = X$ $\quad$ $\mathrm{Tr}_{\mathcal{S}}(\langle a \rangle \varphi_1) = \bigvee_i \langle (a, s_i) \rangle . \mathrm{Tr}_{\mathcal{S}}(\varphi_1))$
$\mathrm{Tr}_{\mathcal{S}}(\neg\varphi_1) = \neg \mathrm{Tr}_{\mathcal{S}}(\varphi_1)$ $\quad$ $\mathrm{Tr}_{\mathcal{S}}(\varphi_1 \vee \varphi_2) = \mathrm{Tr}_{\mathcal{S}}(\varphi_1) \vee \mathrm{Tr}_{\mathcal{S}}(\varphi_2)$
$\mathrm{Tr}_{\mathcal{S}}(\mu X.\varphi_1(X)) = \mu X.\mathrm{Tr}_{\mathcal{S}}(\varphi_1(X))$

Le théorème suivant énonce la correction de la fonction de traduction $\mathrm{Tr}_{\mathcal{S}}$ proposée :

Théorème 4
Soit $\mathcal{S}$ un système non-déterministe, pour toute formule $\varphi \in L_\mu$, on a :

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\mathrm{Tr}_{\mathcal{S}}(\varphi)]\!]^{[val]}_{cod_{\mathcal{S}}(\mathcal{S})}$$

Démonstration
La seule traduction non-triviale concerne la modalité $\langle a \rangle \varphi$. Par la fonction de codage $cod_{\mathcal{S}}$ on a : $\{s_i \in S \mid s_i \in t(s,a)\} = \{s_i \in S \mid s_i \in \widehat{t}(s,(a,s_i))\}$ *d'où le résultat énoncé.*
♢

Pour les formules de Mu-calcul quantifié, la seule difficulté concerne l'opérateur de quantification. On souhaite que le processus étiquetant ne traite pas de manière différente les événements que nous avons désignés comme étant non-distinguables. On pose $I_{\mathcal{S}} = [\bigwedge_{a\in\Sigma} \bigwedge_{i,j} \Downarrow_{(a,s_i),(a,s_j)}]$:

Définition 18
Soit φ une formule de $\mathrm{Q}L_\mu$, $\mathrm{Tr}_{\mathcal{S}}$ est définie de manière inductive sur la structure de φ :

$\mathrm{Tr}_{\mathcal{S}}(\exists X.\alpha) = \exists \widehat{X} \in I_{\mathcal{S}}.\mathrm{Tr}_{\mathcal{S}}(\alpha)$
$\mathrm{Tr}_{\mathcal{S}}(\neg\varphi_1) = \neg\mathrm{Tr}_{\mathcal{S}}(\varphi_1)$
$\mathrm{Tr}_{\mathcal{S}}(\varphi_1 \vee \varphi_2) = \mathrm{Tr}_{\mathcal{S}}(\varphi_1) \vee \mathrm{Tr}_{\mathcal{S}}(\varphi_2)$
en posant $X = \{x_a \mid a \in \Sigma\}$ et $\widehat{X} = \{x_{\widehat{a}} \mid \widehat{a} \in \widehat{\Sigma}\}$.

Si φ est une formule de Mu-calcul, on se réfère à la fonction de traduction définie pour les formules de Mu-calcul.

Complexité : La taille de la formule est modifiée par la transformation $\mathrm{Tr}_{\mathcal{S}}$ dans le cas d'une modalité de type "diamant" $\langle a \rangle$. Compte tenu de la fonction de codage $cod_{\mathcal{S}}$, la longueur de la formule est proportionnelle à la taille du système non-déterministe initial :

$$|\mathrm{Tr}_{\mathcal{S}}(\varphi)| = O(|\varphi| \times |\mathcal{S}|).$$

Pour ce qui est de la classe $I_{\mathcal{S}}$ du processus étiquetant, la taille de la formule qui la caractérise est en :

$$O(|\mathcal{S}|^2 \times |\Sigma|)$$

Définition 19
Soit $\mathcal{E}_{\Downarrow}$ un processus étiquetant appartenant à $I_{\mathcal{S}}$, par application de la fonction bijective $l_{\mathcal{S}}$, on regroupe les (a, s_i)-transitions en a-transitions afin d'obtenir le processus étiquetant $\mathcal{E} = l_{\mathcal{S}}(\mathcal{E}_{\Downarrow})$:

pour tout couple d'états $(r, s) \subseteq S \times S$, si $t_{\mathcal{E}_{\Downarrow}}(r,(a,s_i)) = s$ alors $t_{\mathcal{E}}(r,a) = s$;

Remarque : Si $\mathcal{E}_{\Downarrow}$ est un processus étiquetant de $Lab_{\widehat{X}} \cap I_{\mathcal{S}}$ alors $\mathcal{E}$ est un processus étiquetant de Lab_X.

Propriété 6
La fonction $l_{\mathcal{S}}$ vérifie la propriété suivante :

$$l_{\mathcal{S}}(\mathcal{E}_{\Downarrow} \times \mathcal{E}'_{\Downarrow}) = l_{\mathcal{S}}(\mathcal{E}_{\Downarrow}) \times l_{\mathcal{S}}(\mathcal{E}'_{\Downarrow}) = \mathcal{E} \times \mathcal{E}'.$$

Démonstration
Soient $\mathcal{E}_{\downarrow\downarrow} =< \widehat{X}, E, \epsilon^0, t_{\mathcal{E}_{\downarrow\downarrow}}, L_{\mathcal{E}_{\downarrow\downarrow}} >$ *et* $\mathcal{E}'_{\downarrow\downarrow} =< \widehat{X}, E', \epsilon'^0, t_{\mathcal{E}'_{\downarrow\downarrow}}, L_{\mathcal{E}'_{\downarrow\downarrow}} >$, *on pose* $\mathcal{S}_1$ *et*$\mathcal{S}_2$ *respectivement les processus* $l_{\mathcal{S}}(\mathcal{E}_{\downarrow\downarrow} \times \mathcal{E}'_{\downarrow\downarrow})$ *et* $l_{\mathcal{S}}(\mathcal{E}_{\downarrow\downarrow}) \times l_{\mathcal{S}}(\mathcal{E}'_{\downarrow\downarrow})$. *Prouvons que* $\mathcal{S}_1$ *et*$\mathcal{S}_2$ *sont identiques :*

D'abord, $\mathcal{S}_1$ *et*$\mathcal{S}_2$ *ont le même ensemble d'états inclus dans* $E \times E'$ *et le même état initial* $(\epsilon^0, \epsilon'^0)$.

De même, par produit synchrone, pour tout état (ϵ, ϵ') *des processus* $\mathcal{S}_1$ *et* $\mathcal{S}_2$, *l'ensemble des propositions atomiques associé est :* $L_{\mathcal{E}_{\downarrow\downarrow}}(\epsilon) \cup L_{\mathcal{E}'_{\downarrow\downarrow}}(\epsilon')$.

Enfin pour la fonction de transition, on a (avec $a \neq \tau$*) :*

$$\begin{aligned} t_{\mathcal{S}_2}((\epsilon_1, \epsilon'_1), a) = (\epsilon_2, \epsilon'_2) \quad &\Leftrightarrow \quad t_{l(\mathcal{E}_{\downarrow\downarrow})}(\epsilon_1, a) = \epsilon_2 \text{ et } t_{l(\mathcal{E}'_{\downarrow\downarrow})}(\epsilon'_1, a) = \epsilon'_2 \\ &\Leftrightarrow \quad t_{\mathcal{E}_{\downarrow\downarrow}}(\epsilon_1, (a, s_i)) = \epsilon_2 \text{ et } t_{\mathcal{E}'_{\downarrow\downarrow}}(\epsilon'_1, (a, s_i)) = \epsilon'_2 \\ &\quad\ \forall (a, s_i) \in \widehat{\Sigma} \\ &\Leftrightarrow \quad t_{\mathcal{E}_{\downarrow\downarrow} \times \mathcal{E}'_{\downarrow\downarrow}}((\epsilon_1, \epsilon'_1), (a, s_i)) = (\epsilon_2, \epsilon'_2) \\ &\quad\ \forall (a, s_i) \in \widehat{\Sigma} \\ &\Leftrightarrow \quad t_{\mathcal{S}_1}((\epsilon_1, \epsilon'_1), a) = (\epsilon_2, \epsilon'_2) \end{aligned}$$

Donc $\mathcal{S}_1$ *et* $\mathcal{S}_2$ *ont la même fonction de transition et les processus* $l(\mathcal{E}_{\downarrow\downarrow} \times \mathcal{E}'_{\downarrow\downarrow})$ *et* $l(\mathcal{E}_{\downarrow\downarrow}) \times l(\mathcal{E}'_{\downarrow\downarrow})$ *sont identiques.*

♢

Lemme 4
Les processus $cod_{\mathcal{S}}(\mathcal{S} \times \mathcal{E})$ *et* $cod_{\mathcal{S}}(\mathcal{S}) \times \mathcal{E}_{\downarrow\downarrow}$ *sont identiques.*

Démonstration
Les processus $cod_{\mathcal{S}}(\mathcal{S} \times \mathcal{E})$ *et* $cod_{\mathcal{S}}(\mathcal{S}) \times \mathcal{E}_{\downarrow\downarrow}$ *ont le même ensemble d'états inclus dans* $S \times E$, *le même état initial* (s^0, ϵ^0) *et chaque état à le même ensemble de propositions atomiques.*

Pour la fonction de transition, on a d'une part, pour le processus $cod_{\mathcal{S}}(\mathcal{S}) \times \mathcal{E}_{\downarrow\downarrow}$ *:*

$$\begin{aligned} t((s, \epsilon), (a, s_i)) \quad &= \quad (\widehat{t}(s, (a, s_i)), t_{\mathcal{E}_{\downarrow\downarrow}}(\epsilon, (a, s_i))) \\ &= \quad (t(s, a), t_{\mathcal{E}_{\downarrow\downarrow}}(\epsilon, (a, s_i))) \end{aligned}$$

D'autre part, pour le processus $cod_{\mathcal{S}}(\mathcal{S} \times \mathcal{E})$, *on a :*
$\widehat{t}((s, \epsilon), (a, (s_i, \epsilon_i))) = t_{\mathcal{S} \times \mathcal{E}}((s, \epsilon), a) = (s_i, \epsilon_i)$.
or : $t_{\mathcal{S} \times \mathcal{E}}((s, \epsilon), a) = (t(s, a), t_{\mathcal{E}}(\epsilon, a))$
Par définition de $l_{\mathcal{S}}$, *on a* $t_{\mathcal{E}}(\epsilon, a) = t_{\mathcal{E}_{\downarrow\downarrow}}(\epsilon, (a, s_i))$ *donc* $cod_{\mathcal{S}}(\mathcal{S} \times \mathcal{E})$ *et* $cod_{\mathcal{S}}(\mathcal{S}) \times \mathcal{E}_{\downarrow\downarrow}$ *ont la même fonction de transition et sont identiques.*

♢

Comme pour la propriété 5, la propriété suivante énonce un critère de correction pour la traduction des énoncés logiques proposée dans la définition 18.

Propriété 7 (Équivalence des solutions)
$\mathcal{E}_{\downarrow\downarrow} \in I_{\mathcal{S}} \cap Lab_{\widehat{X}}$ *est un processus étiquetant pour le processus* $\widehat{\mathcal{S}} = cod_{\mathcal{S}}(\mathcal{S})$ *et pour la sentence* $\mathrm{Tr}_{\mathcal{S}}(\alpha) \in \mathrm{Q}^{\circlearrowleft} L_\mu$ *si et seulement si* $\mathcal{E} = l_{\mathcal{S}}(\mathcal{E}_{\downarrow\downarrow})$ *est un processus étiquetant sur* X *pour le système non-déterministe* $\mathcal{S}$ *et la sentence* $\alpha \in \mathrm{Q}L_\mu$.

Démonstration
La démonstration de cette propriété est similaire à celle de la propriété 5 : on procède par induction sur la structure de α. Si α est une formule du Mu-calcul pur, on conclue à l'aide du lemme 4 et du théorème 4. Pour le cas où α est de la forme $\exists X.\alpha'$, on utilise la définition de $l_{\mathcal{S}}$ et la propriété 6.

$\diamondsuit$

Théorème 5
Soit $\mathcal{S}$ un système non-déterministe, pour toute formule $\varphi \in \mathrm{Q}L_\mu$, on a :

$$[\![\varphi]\!]^{[val]}_{\mathcal{S}} = [\![\mathrm{Tr}_{\mathcal{S}}(\varphi)]\!]^{[val]}_{cod_{\mathcal{S}}(\mathcal{S})}$$

Démonstration
La démonstration de ce théorème est similaire à celle du théorème 2 qui utilise le résultat de la propriété 5 : on procède par induction sur la structure de α et on utilise la propriété 7 pour conclure dans le cas de la quantification $\exists X.\alpha$.

$\diamondsuit$

Dans cette section, nous avons vu une seconde façon d'envisager la réduction d'un problème de contrôle d'un système non-déterministe en un problème de contrôle d'un processus basée sur les travaux de [ABW03]. Nous proposons, dans une ultime partie, de comparer ces deux approches en terme de complexité ce qui présentera les inconvénients de cette seconde approche.

Chapitre 5

Conclusion

Dans ce mémoire nous avons étudié deux réductions possibles d'un problème de contrôle $\exists X.\varphi$ d'un système non-déterministe $\mathcal{S}$ en un problème de contrôle d'un processus :

1. la première approche se base sur un codage $cod_<$ des systèmes non-déterministes en processus utilisant un dépliage restreint du système puis l'ajout d'une nouvelle action τ inobservable et incontrôlable pour régler les cas de non-déterminisme.
   La fonction de traduction Tr permet de convertir l'énoncé en Mu-calcul quantifié du problème de contrôle.
   Le problème de contrôle $\mathcal{P} : \mathcal{S} \models \exists X.\varphi$ se réduit alors en un problème de contrôle $\mathcal{P}_1 : cod_<(\mathcal{S}) \models \exists X \cup \{x_\tau\} \in Loop(\tau).\mathrm{Tr}(\varphi)$. L'algorithme de synthèse d'un processus étiquetant permet de résoudre $\mathcal{P}_1$ (cf. [Rie03]). On obtient une solution pour $\mathcal{P}$ à partir de la solution de $\mathcal{P}_1$ grâce à la fonction l qui supprime les boucles de τ ;

2. la seconde approche se base sur une fonction de codage $cod_\mathcal{S}$ qui exploite un renommage des événements du système afin de résoudre les cas de non-déterminisme.
   Le problème de contrôle $\mathcal{P} : \mathcal{S} \models \exists X.\varphi$ se réduit alors en un problème de contrôle $\mathcal{P}_2 : cod_\mathcal{S}(\mathcal{S}) \models \exists \widehat{X}.\mathrm{Tr}_\mathcal{S}(\varphi)$.
   Comme précédemment, la fonction $l_\mathcal{S}$ permet d'établir une correspondance entre les solutions des problèmes de contrôle $\mathcal{P}$ et $\mathcal{P}_2$ en regroupant les a-transitions que l'on avait renommées.

Comparons ces deux approches en terme de complexité. On a déjà vu au cours de ce rapport que :

	Réduction en $\mathcal{P}_1$	Réduction en $\mathcal{P}_2$
Fonction de codage en processus	$O(\lvert\mathcal{S}\rvert^2 \times \lvert\Sigma\rvert)$	$O(\lvert\mathcal{S}\rvert)$
Traduction des énoncés logiques de $\mathrm{Q}L_\mu$	$O(\lvert\varphi\rvert)$	$O(\lvert\mathcal{S}\rvert \times \lvert\varphi\rvert)$
Classe du processus étiquetant	$O(1)$	$O(\lvert\mathcal{S}\rvert^2 \times \lvert\Sigma\rvert)$

La démarche générale pour synthétiser un processus étiquetant pour la formule $\mathcal{R} \models \exists X \in \psi.\phi$ est la suivante (cf. [Rie03]) :

- on construit l'automate alternant $\mathcal{A}_\phi$ équivalent à ϕ (cf. [Wal01]) ;
- on construit l'automate alternant $\mathcal{A}_\phi/\mathcal{R} \wedge \mathcal{A}_\psi$ (cf. le quotient d'un automate par un processus dans [AVW03]) ;
- on construit un automate non-déterministe $\mathcal{A}$ à partir de l'automate alternant $\mathcal{A}_\phi/\mathcal{R} \wedge \mathcal{A}_\psi$ (théorème de simulation, cf. [EJ91] et [MS95]) ;
- on calcule un modèle de l'automate non-déterministe $\mathcal{A}$ en recherchant une stratégie gagnante dans le jeu de parité $\mathcal{G}(\mathcal{A})$ associé à $\mathcal{A}$ (théorème de vacuité des automates non-déterministes, cf. [AVW03]).

D'un point de vue complexité, [AVW03] indique que la taille de l'automate non-déterministe $\mathcal{A}$ équivalent à l'automate alternant $\mathcal{A}_\phi/\mathcal{R} \wedge \mathcal{A}_\psi$ est $|\mathcal{R}| \times 2^{O(m.log(m))}$ avec $m = |\mathcal{A}_\phi| + |\mathcal{A}_\psi| = |\phi| + |\psi|$.

Appliqué à notre contexte, ceci donne :

- Réduction de $\mathcal{P}$ en $\mathcal{P}_1$: $O(|\mathcal{S}|^2|\Sigma| \times 2^{O(m.log(m))})$ avec $m = |\varphi|$
- Réduction de $\mathcal{P}$ en $\mathcal{P}_2$: $O(|\mathcal{S}| \times 2^{O(m.log(m))})$ avec $m = |\mathcal{S}||\varphi| + |\mathcal{S}|^2|\Sigma|$

Ainsi, en dépit du fait que la taille du système non-déterministe croisse de manière polynomiale lors du codage en processus, la première réduction proposée a un coût moindre lorsque l'on considère la recherche de solution puisqu'elle est polynomiale en la taille du système non-déterministe initial (alors que la réduction en $\mathcal{P}_2$ est exponentielle pour la recherche de solution).

Pour finir, une extension envisageable de ces travaux concerne les problèmes de contrôle sous observation partielle. Ces derniers peuvent être spécifiés dans la logique du Loop Mu-calcul quantifié (cf. [RP03a]). Il faudrait alors définir, à l'image de ce qui a été fait pour les formules de Mu-calcul quantifié, une sémantique de la logique pour les systèmes non-déterministes et une fonction de traduction qui prenne en compte les classes des contrôleurs spécifiés. De notre point de vue, cet extension ne pose pas de problème particulier.

Bibliographie

[ABW03] Arnold (A.), Briand (X.) et Walukiewicz (I.). – Synthesis of decentralized controllers : decidable and undecidable cases. *In : ATPN - Workshop on DES control*. – 2003. 24th Int. conf. on Application Theory of Petri Nets (ATPN 2003), Eindhoven.

[AVW03] Arnold (A.), Vincent (A.) et Walukiewicz (I.). – Games for synthesis of controllers with partial observation. *Theorical Computer Science*, vol. 303, n° 1, 2003, pp. 7–34.

[Ber93] Bergeron (A.). – A unified approach to control problems in discrete event processes. *Theorical Informatics and Applications*, vol. 27, n° 6, 1993, pp. 555–573.

[EJ91] Emerson (E.A.) et Jutla (C.S.). – Tree automata, mu-calculus ans determinacy. *In : IEEE Computer Society Press*, pp. 81–98. – 1991.

[KMTV00] Kupferman (Orna), Madhusudan (P.), Thiagarajan (P. S.) et Vardi (Moshe Y.). – Open systems in reactive environments: Control and synthesis. *Lecture Notes in Computer Science*, vol. 1877, 2000, pp. 92–107.

[Koz83] Kozen (D.). – Results on the propositional mu-calculus. *Theorical Computer Science*, vol. 27, 1983, pp. 333–354.

[Mil89] Milner (R.). – *Communication and Concurrency*. – Prentice Hall, 1989.

[MS95] Muller (D.E.) et Schupp (P.E.). – Simulating alternating tree automata by nondeterministic automata : New results and new proofs of the theorems of rabin, mc naughton and safra. *Theorical Computer*, vol. 141, 1995, pp. 69–107.

[Rie03] Riedweg (S.). – *Logiques pour le contrôle d'automatismes discrets*. – Thèse de PhD, IRISA Rennes, 2003.

[RP03a] Riedweg (S.) et Pinchinat (S.). – Quantified loop-mu-calculus for control under partial observation. – Septembre 2003. Rapport de recherche INRIA Rennes n°4949.

[RP03b] Riedweg (S.) et Pinchinat (S.). – Quantified mu-calculus for control synthesis. *In : Mathematical Foundations of Computer Science*. – Bratislava, Slovak Republic, août 2003.

[RW87a] Ramadge (P.J.) et Wonham (W.M.). – On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, vol. 25, n° 3, 1987, pp. 637–659.

[RW87b] Ramadge (P.J.) et Wonham (W.M.). – Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, vol. 25, n° 1, 1987, pp. 206–230.

[RW89] Ramadge (P.J.) et Wonham (W.M.). – The control of discrete event systems. *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, vol. 27, n° 1, 1989, pp. 81–98.

[Tho97] Thomas (W.). – Languages, automata and logic. *In : Handbook of Formal Languages*, éd. par Salomaa (A.) et Rozenberg (G.). – Berlin, Springer-Verlag, 1997.

[TW94a] Thistle (J.G.) et Wonham (W.M.). – Control of infinite behavoir of finite automata. *SIAM Journal of Control and Optimization*, vol. 32, n° 4, 1994, pp. 1075–1097.

[TW94b] Thistle (J.G.) et Wonham (W.M.). – Supervision of infinite behavoir of discrete-event systems. *SIAM Journal of Control and Optimization*, vol. 32, n° 4, 1994, pp. 1098–1113.

[Wal01] Walukiewicz (I.). – Automata and logic. *Notes from EEF Summer school'01*, 2001.