Discipline : Informatique

Titre : Sur la décidabilité de certains problèmes de synthèse de contrôleurs.

Résumé :
Cette thèse s'incrit dans les travaux sur le contrôle des systèmes à événements discrets (SED). Il s'agit de modifier, à l'aide de contrôleurs, le comportement d'un système, le plus souvent modèlisé par un automate, afin de réaliser une spécification souhaitée. Nous étudions ici le contrôle centralisé et décentralisé avec information partielle d'un SED. Les spécifications ainsi que les contraintes de contrôlabilité des contrôleurs (événements incontrôlables ou forçables, contrôle dynamique) sont exprimées avec une logique décidable : μ-calcul modal. L'information partielle sur les comportements d'un SED s'exprime grâce à des extensions de cette logique permettant de spécifier les notions usuelles d'événements inobservables et indiscernables. Nous montrons que ces extensions sont décidables ainsi que le contrôle centralisé. Nous prouvons que le contrôle décentralisé est indécidable si certains événements sont inobservables ou indiscernables. Nous exhibons deux cas décidables de contrôle décentralisé (restriction des spécifications ou des communications). Nous montrons enfin qu'étendre le μ-calcul modal afin de spécifier des comportements inobservables ou indiscernables rend cette logique indécidable.

Title : On the decidability of certains problems of controllers synthesis

Abstract :
This thesis studies the control of discrete events systems (DES). The aim of this theory is to modify, with controllers, the behavior of a system, which is most of the cases an automaton, in order to satisfy a specification. We consider here the centralized and decentralized control with partial information on the DES. The specifications as well as the controllability constraints for the controllers (uncontrollable or forced events, dynamic control) are expressed with a decidable logic : the modal μ-calculus. The partial information on the behaviors of the DES is modeled with an extension of this logic which allow to express the usual notion of unobservable events and indistinguishable events. We show that these extensions are decidable as well as the centralized control. We prove that the decentralized control with unobservable or indistinguishable events is undecidable. We exhibit two particular cases of decentralized control that are decidable (restriction of the specifications or of the communications). Finally, we show that the extensions of the modal μ-calculus that permit to express unobservable or indistinguishable behaviors are undecidable logics.

Xavier BRIAND. 28 juin 2006. Sur la décidabilité de certains problèmes de synthèse de contrôleurs.

N° d'ordre : 3194

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **XAVIER BRIAND**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

SUR LA DÉCIDABILITÉ DE CERTAINS PROBLÈMES DE SYNTHÈSE DE CONTRÔLEURS

Soutenue le : 28 juin 2006

Après avis des rapporteurs :

Rapporteurs :

Joachim Niehren	Directeur de Recherche INRIA
Sophie Pinchinat	Maître de Conférence HDR

Devant la commission d'examen composée de :

Bruno Courcelle	Professeur	Examinateur
Jean-michel Couvreur	Professeur	Examinateur
Joachim Niehren	Directeur de Recherche INRIA	Rapporteur
Sophie Pinchinat	Maître de Conférence HDR ...	Rapporteur
Igor Walukiewicz	CR CNRS, HDR	Directeur de thèse
Pascal Weil	Directeur de Recherche CNRS	Examinateur

– 2006 –

Sur la décidabilité de certains problèmes de synthèse de contrôleurs

Xavier Briand [1]

Directeurs de thèse :
André Arnold et Igor Walukiewicz

[1]LaBRI, Université Bordeaux I and CNRS (UMR 5800)

Table des matières

Première partie

Le problème de la synthèse de contrôleur

Remerciements

Merci à Sophie Pinchinat et Joachim Niehren d'avoir accepté d'être les rapporteurs de cette thèse. Je remercie plus particulièrement Sophie Pinchinat pour ses notes très détaillées qu'elle m'a transmise pour la version finale.

Merci à Bruno Courcelle, Jean-michel Couvreur et Pascal Weil pour avoir accepté de participer au jury de la soutenance de thèse.

Merci à mon premier directeur de thèse André Arnold pour toute sa diponibilité et tous ces conseils qui ont guidé ce travail de recherche aussi bien avant qu'après son départ en retraite.

Merci à Igor Walukiewicz, mon deuxième directeur de recherche, d'avoir accepté de continuer et de faire aboutir les recherches déjà entreprises.

0.1 Introduction

Nous étudions dans ce travail certaines questions de décidabilité et d'indécidabilité de la théorie du contrôle des système à événements discrets (SED). Introduisons d'abord ces systèmes et les problèmes de contrôle associés.

Un système à événements discrets modélise les changements d'états d'un système réagissant à certains événements. Ces changements sont considérés à la fois comme discrets et instantanés. L'occurrence d'un événement est généralement indépendante du système. Dans tout ce qui suit, nous nous intéressons aux comportements qualitatifs, i.e. nous ne prenons pas en compte le temps et en particulier les dates auxquelles se produisent les changements d'états du système. De plus, les systèmes étudiés sont déterministes : nous considérons que les événements ne peuvent induire qu'une unique réaction du système dépendant de l'état dans lequel est celui-ci. Le contrôle d'un système à événements discrets consiste à modifier les comportements possible du SED afin qu'il puisse réaliser une spécification donnée.

Ramadge et Wonham ont introduit la théorie du contrôle des systèmes à événements discrets dans [RW87] en utilisant la théorie des langages formels (voir aussi [RW89] ainsi que les livres [CL99] et [KG95]). Dans ce cadre, les événements auxquels un SED réagit sont symbolisés par un ensemble fini A appelé, pour simplifier, l'ensemble des événements ou alphabet des événements. Notons A^* l'ensemble des mots finis sur l'alphabet A contenant le mot vide ε, et A^ω l'ensemble de mots infinis sur A. Un SED étant ici déterministe, ses comportements finis (qualitatifs) peuvent être identifiés aux suites d'événements auxquelles il réagit et donc à un sous-ensemble de A^*, c'est à dire à un langage clos par préfixes sur l'alphabet A.

Comme dans la plupart des travaux, nous allons utiliser les automates déterministes pour représenter le langage associé aux comportements d'un SED. Un SED est alors modélisé par un ensemble d'états et de transitions entre ces états, chacunes étiquetées par un événement de l'alphabet A. Lorsque aucun événement n'a fait évoluer le système, celui-ci est dans un état particulier, appelé état initial.

Poursuivant ici la démarche entreprise dans [AVW03], nous associons à chaque état de l'automate un ensemble de symboles appartenant à un ensemble fini Λ, modélisant ainsi certaines propriétés des états du SED. Nous appelons par la suite ces automates des *processus*.

Un processus P permet donc de modéliser à la fois un ensemble de comportements finis $\mathcal{L}(P)$ (un langage clos par préfixe inclus dans A^*), un ensemble de comportements infinis $\mathcal{L}^\omega(P)$ (un sous-ensemble de A^ω) et aussi

des familles de comportements dans $A^* \cup A^\omega$ dont les suites d'états du SED auxquelles elles correspondent, satisfont certaines propriétés dans Λ. Nous pouvons, en particulier, modéliser les langages "marqués" de l'approche de Ramadge et Wonham. Si $m \in \Lambda$, on peut s'intéresser au langage "marqué" par m, noté $\mathcal{L}_m(P)$, c'est à dire à l'ensemble des comportements finis atteignant un état satisfaisant la propriété m.

La principale restriction que nous faisons sur ces automates est de considérer que l'ensemble des états est *fini*. Nous nous restreignons donc aux langages réguliers.

La notion de SED présentée plus haut modélise donc un système qui engendre des suites d'événements sans qu'il soit possible d'effectuer un contrôle sur l'occurrence de ces événements. Afin de contrôler un SED, on suppose que les occurrences des événements, ou au moins certaines d'entre elles, peuvent être interdites. Le contrôle d'un SED consiste alors à interdire l'occurrence de certains événements en fonction de l'état du SED afin que celui-ci ne puisse réaliser que certains comportements souhaités.

Le système contrôlé, appelé aussi système supervisé, est ici le produit synchrone $P \times Q$ de deux processus, le SED P et le contrôleur Q réagissant en même temps aux mêmes événements. Plus précisément, $P \times Q$ est un processus dont les états sont les couples formés des états de P et de Q, et dont les transitions sont celles possibles pour P et Q. Ainsi, dans un état donné, le contrôleur interdit l'occurrence d'un événement si la transition par cet événement n'existe pas. De plus, nous considérons que chaque état du système supervisé est étiqueté par l'union des étiquettes de l'état de P et de l'état de Q. En quelque sorte, le système supervisé "accumule" les propriétés locales des états de P et de Q. Cette définition permet de ne pas différencier le SED du contrôleur (le produit $P \times Q$ est commutatif). On peut, par conséquent, envisager que le contrôleur fasse lui-même l'objet d'un contrôle. Cependant, nous considérerons souvent, à travers de nombreux exemples, que les contrôleurs sont des processus sans propriété locale : leurs états ne sont pas étiquetés.

Pour formaliser les problèmes de contrôle, nous introduisons, suivant encore les travaux de [AVW03], non seulement la spécification $\mathcal{S}$ que doit réaliser le système supervisé mais aussi une spécification $\mathcal{C}$ représentant les contraintes de contrôlabilité du contrôleur. Nous détaillerons plus loin ces spécifications. Le problème du contrôle *centralisé* peut alors s'énoncer ainsi :

Soit $\mathcal{S}$ et $\mathcal{C}$ des spécifications et P un SED. Existe-t-il un contrôleur Q satis-

faisant $\mathcal{C}$ tel que $P \times Q$ satisfait $\mathcal{S}$?

Le problème de la *synthèse* de contrôleur est alors de construire le contrôleur Q si il existe.

Ce problème de contrôle est qualifié de centralisé car un seul contrôleur modifie les comportements possibles du SED. On peut envisager, comme l'on fait les premières études sur le contrôle d'un SED (voir [RW88] et [CDFV88]), que le contrôle du SED soit modulaire. Dans ce contexte, le blocage ou l'autorisation de l'occurrence d'un événement est la résultante de l'action de plusieurs contrôleurs. La définition d'un cadre général pour l'étude du contrôle modulaire fait aujourd'hui encore l'objet de différents travaux (voir [LY02] et [PR05b]). On peut ici très naturellement présenter un cadre particulier du contrôle modulaire comme suit.

Soit $\mathcal{S}$ et $\mathcal{C}_1, \ldots, \mathcal{C}_n$ des spécifications et P un SED. Existe-t-il des contrôleurs $Q_1, \ldots Q_n$ satisfaisant respectivement $\mathcal{C}_1, \ldots, \mathcal{C}_n$ tels que $P \times Q_1 \times \cdots Q_n$ satisfait $\mathcal{S}$?

Ainsi les contrôleurs suppriment l'occurrence d'un événement si au moins l'un d'eux la supprime. On parlera alors du problème de contrôle *décentralisé*, le problème de la synthèse étant là aussi de construire les contrôleurs $Q_1, \ldots Q_n$.

Nous allons par la suite étudier différents problèmes de contrôle dépendant du type des spécifications pour les contrôleurs et le système supervisé. Revenons d'abord sur les principales spécifications prises en compte dans l'approche de Ramadge et Wonham.

Dans les premiers travaux sur la théorie du contrôle, la spécification pour un système supervisé $P \times Q$ (produit du SED P et du contrôleur Q) a consisté à fixer un langage $L \subset \mathcal{L}(P) \subset A^*$ tel que l'ensemble des comportements finis du système supervisé $\mathcal{L}(P \times Q)$ soit le plus grand possible (pour la relation d'inclusion) parmi ceux admissibles par L ou alors, inclus entre un langage minimal "requis" et L.

Une autre exigence a été aussi très rapidement prise en compte, celle du non blocage (voir par exemple [CL91]). Il s'agit de considérer un langage marqué du SED P par un symbole $m \in \Lambda$ et de faire en sorte que les comportements finis $\mathcal{L}(P \times Q)$ du système supervisé puissent atteindre

des états "marqués" par m. C'est à dire $\mathcal{L}(P \times Q) \subseteq Pref(\mathcal{L}_m(P \times Q))$ où $Pref(\mathcal{L}_m(P \times Q))$ est l'ensemble des préfixes des mots de $\mathcal{L}_m(P \times Q)$ (On considère ici que les états du contrôleur n'ont pas la propriété m).

Ces spécifications se sont aussi étendues au cas des comportements infinis du système supervisé $\mathcal{L}^\omega(P \times Q)$ (voir [R89, TW94, TW94b]). Ceux-ci doivent, par exemple, réaliser certains des comportements infinis $L^\omega \subseteq \mathcal{L}^\omega(P)$ du SED, c'est à dire $\mathcal{L}^\omega(P \times Q) = L^\omega$.

Dans le cadre proposé par Ramadge et Wonham, deux types de contraintes pour les contrôleurs ont été introduites, des contraintes de contrôlabilité et des contraintes d'observabilité.

Les premières consistent à supposer qu'une partie $A_{uc} \subseteq A$ des événements est *incontrôlable*, c'est à dire que le contrôleur ne peut interdire l'occurrence des événements de A_{uc}. Dans ce contexte, les problèmes de contrôle consiste donc à trouver des contrôleurs qui ne peuvent interdire les transitions par des événements incontrôlables tout en restreignant les comportements du SED afin de satisfaire une spécification donnée.

Le second type de contraintes, introduit dans [CDFV88] et [LW88], permet de prendre en compte le cas où le contrôleur a une information partielle du comportement antérieur du SED. Le contrôleur ne "voit" le SED qu'à travers un *masque* $M : A \to \Delta \cup \{\varepsilon\}$ qui associe à un événement, soit le mot vide ε, soit un symbole dans l'ensemble Δ. Plus précisément, si $M(a) = M(b)$, alors le contrôleur ne peut distinguer les événements a et b : ils sont *indiscernables*. De plus, une partie des événements $M^{-1}(\varepsilon) = A_{uo}$ est *inobservable* : le contrôleur ne peut détecter s'ils se sont produits. Un contrôleur ayant, pour contrainte, le masque M doit donc se comporter de façon identique après deux événements indiscernables et avoir le même comportement si un événement inobservable s'est produit ou non.

La démarche entreprise avec le formalisme de Ramadge et Wonham peut se décomposer en trois types de problèmes. Tout d'abord, certains travaux ont permis de définir des conditions nécessaires et suffisantes à l'existence de solutions pour les différents problèmes de contrôle évoqués plus haut. D'autres ont recherché les solutions optimales et leurs calculabilités : comportement admissible maximum ou solutions maximales et comportement requis minimum ou solutions minimales. On pourra se référer, entre autres, aux travaux de synthèse [RW89], [CL99] et [KG95] pour plus de détails.

Dans [AVW03], Arnold, Vincent et Walukiewicz introduisent un nouveau cadre pour étudier le contrôle d'un SED qui permet d'utiliser un même type de spécifications pour le système supervisé, les contrôleurs et même, lorsqu'elles existent, les solutions aux problèmes de contrôle. C'est cette approche que nous allons rappeler et poursuivre ici.

Dans [AVW03], les auteurs proposent d'utiliser une logique, le *μ-calcul modal*, introduit dans [Koz83] (voir aussi [AN01]), pour les spécifications qui permet d'exprimer toutes les spécifications du système supervisé ainsi que les contraintes de contrôlabilité du contrôleur déjà évoquées, pourvu que celles-ci soient des langages réguliers. Une formule Φ du μ-calcul modal permet de spécifier non seulement un ensemble régulier de comportements (un langage marqué d'un processus) mais aussi une famille régulière de processus : ce sont tous les processus P satisfaisant la formule Φ, noté $P \models \Phi$. Nous détaillerons l'expressivité du μ-calcul modal et sa décidabilité plus tard. Nous pouvons remarquer pour l'instant que cette logique n'a pas suffi aux auteurs de [AVW03] pour exprimer les contraintes d'observabilité des contrôleurs que l'approche de Ramadge et Wonham a permis d'étudier. Ils ont alors introduit une extension de cette logique, que l'on peut appeler le *μ-calcul modal avec événements inobservables*, qui permet de spécifier qu'un processus, après une transition par un événement a donné, reste dans le même état. Le processus se comporte donc de la même façon que l'événement a se produise ou non : l'événement a est inobservable. Grâce à cette extension, les contraintes d'observabilité classiques peuvent être exprimées.

La démarche de [AVW03] n'est pas de déterminer des conditions nécessaires et suffisantes pour l'existence de solutions aux problèmes de contrôle, mais d'étudier la décidabilité des problèmes de contrôle. Ils montrent que le contrôle centralisé avec des spécifications formulées en μ-calcul modal avec événements inobservables, c'est à dire permettant d'exprimer le fait que des événements sont inobservables, est décidable. Le problème de la synthèse de contrôleur est lui aussi décidable. L'ensemble des contrôleurs qui sont solutions du problème de contrôle, est lui-même exprimable par une formule du μ-calcul modal avec événements inobservables que nous détaillerons plus loin. Le problème décentralisé n'est décidable que si un seul des contrôleurs a une spécification en μ-calcul modal avec événements inobservables, les autres ayant pour spécifications une formule du μ-calcul modal. L'indécidabilité du contrôle décentralisé lorsque au moins deux contrôleurs ont des spécifications en μ-calcul modal avec événements inobservables est aussi démontrée.

Nous allons poursuivre ce travail en prenant en compte la notion d'événements indiscernables. Nous introduisons une extension du μ-calcul modal permettant de spécifier qu'un processus, après une transition par deux événements a et b donné, atteint le même état. Ainsi, que a ou b se produisent, le processus se comporte de la même manière : ces événements sont indiscernables. Nous appelons *μ-calcul modal avec événements indiscernables*, le μ-calcul modal avec ces nouvelles contraintes et *μ-calcul modal étendu* le μ-calcul modal avec événements inobservables et indiscernables.

Avec le même type de technique que [AVW03], nous montrons que le problème de contrôle centralisé et le problème de la synthèse avec des spécifications en μ-calcul modal étendu, sont décidables. Le cas décentralisé est décidable si un seul des contrôleurs a pour spécification une formule du μ-calcul modal étendu, les autres ayant des spécifications en μ-calcul modal. Nous montrons que si deux contrôleurs ont des contraintes d'indiscernabilité, c'est à dire que leurs spécifications sont des formules du μ-calcul modal avec événements indiscernables, le problème du contrôle décentralisé est indécidable. De même, si un contrôleur a une formule du μ-calcul modal avec événements indiscernables et un autre a une formule du μ-calcul modal avec événements inobservables, le contrôle décentralisé est indécidable.

L'indécidabilité du contrôle décentralisé avec événements inobservables *ou* indiscernables nous ont amené naturellement à rechercher un autre cadre décidable pour les spécifications. Dans les travaux de [CDFV88] et [RuWo92], les auteurs mettent en évidence des cas de contrôle décentralisé décidables avec l'approche de Ramadge et Wonham. [AVW03] montre que le cas de [RuWo92] revient à n'admettre qu'un type très *restreint* de spécifications en μ-calcul modal pour le système supervisé et les contrôleurs. Nous présentons ici un cas de contrôle décentralisé où les spécifications autorisées sont un peu plus générales que celles proposées dans [AVW03].

Une autre restriction possible pour rendre décidable le problème du contrôle décentralisé consiste à restreindre les transitions possibles du produit de processus. Les transitions du produit de processus introduit plus haut ne dépendent que de l'existence de ces transitions pour chacun des processus. On peut envisager que les transitions autorisées du produit de processus dépendent aussi des états qu'elles atteignent. Nous définirons pour cela un produit de processus qualifié de *bien synchronisé* qui représente en quelque sorte une communication entre ces processus car ceux-ci échangent implicitement de l'information sur leurs états respectifs. Nous prouverons qu'il est décidable de trouver les contrôleurs, solutions du contrôle décentralisé avec spécifications en μ-calcul étendu, si ceux-ci forment un système supervisé

bien synchronisé. Cette approche provient d'une communication personnelle d'Igor Walukiewicz, introduisant la notion de bonne synchronisation de processus et qui prouve, pour ce type de restrictions, la décidabilité du problème de contrôle décentralisé avec événements inobservables. Cette notion de bonne synchronisation est ici étendue pour prendre en compte le cas des événements indiscernables.

La notion d'information partielle du contrôleur sur le comportement du SED est réduit, dans l'approche de Ramadge et Wonham ou celle du μ-calcul étendu, aux cas des événements inobservables ou indiscernables. Nous allons aussi étudier un autre type de contraintes où le contrôleur a une information partielle non pas sur les événements mais sur les comportements du SED. D'un point de vue formel, on peut considérer une fonction de *masque* non pas sur les événements mais sur les comportements $M : A^* \to \Delta \cup \{\varepsilon\}$. Ainsi $M^{-1}(\varepsilon)$ est l'ensemble des comportements inobservables et, si $M(u) = M(v)$, alors les comportements u et v sont indiscernables. De plus, il semble réaliste de considérer que des comportements soient indiscernables bien que les événements puissent être discernés. En effet, supposons qu'un contrôleur ne détecte pas l'ordre dans lequel se produise les événements a et b et les événements a et c mais discerne le "paquet" $\{a, b\}$ du "paquet" $\{a, c\}$. Dans ce cas, le contrôleur ne discerne pas les comportements ab et ba mais doit discerner les événements a et b.

Afin de modéliser ces contraintes, nous proposons une extension du μ-calcul modal qui permet de spécifier qu'après un comportement donné un processus revient dans le même état, ce sont les comportements inobservables. Les comportements indiscernables sont eux modélisés en spécifiant que deux comportements atteignent le même état. Nous prouvons que ces extensions du μ-calcul modal rendent cette logique indécidable. Les problèmes de contrôle avec ces spécifications sont donc indécidables. Dans cette étude, un cas reste cependant ouvert, celui où le contrôleur ne peut pas observer certains comportements de longueur 2. En d'autre termes, soit on a $M(u) = u$ pour $u \in A^*$ ou soit $M(u) = \varepsilon$ et $u \in A^2$.

Revenons maintenant plus en détail sur le formalisme et l'expressivité des spécifications développées ici et tout d'abord, sur le μ-calcul modal.

A l'image de [AVW03], nous définissons les spécifications non pas par des formules du μ-calcul modal mais avec des automates, appelé *automates*

simples. On pourra se référer à [Wal01] pour montrer l'équivalence d'expressivité entre automates simples et μ-calcul modal sur les processus. Si l'on ne considère d'un processus que l'arbre représentant l'ensemble de comportements depuis l'état initial, la notion d'automate simple correspond à la notion standard d'automate alternant sur les arbres de degré borné (par l'alphabet des événements A). Un automate simple $\mathcal{A}$ définit un ensemble régulier de processus : l'ensemble des processus P ayant un calcul acceptant, c'est à dire l'ensemble des processus P satisfaisant la spécification $\mathcal{A}$, encore appelé modèles de $\mathcal{A}$. Un automate simple définit ses modèles à *bisimulation près*, il ne spécifie que des ensembles possibles de comportements pour un processus et ne spécifie rien sur la façon dont celui-ci les réalise. Pour plus de compréhension, nous nous permettons dans cette introduction de détailler de façon informelle ces automates.

Un calcul de $\mathcal{A}$ sur un processus P est un ensemble de suites de transitions entre les états de $\mathcal{A}$, associées chacunes à un comportement de P. Les transitions possibles entre les états de $\mathcal{A}$ sont soit internes à $\mathcal{A}$, soit dépendantes des événements formant les comportements de P mais aussi des propriétés locales des états de P (un sous-ensemble de Λ). De plus les états Q de $\mathcal{A}$ sont partionnés en états existentiels $Q^{\exists}$ et en états universels $Q^{\forall}$. Lorsque l'automate $\mathcal{A}$ est dans un état existentiel, il *choisit* une transition parmi celles possibles et, dans un état universel, il doit continuer son calcul sur *toutes* les transitions possibles. Dans chaque état universel, $\mathcal{A}$ peut aussi localement spécifier un ensemble de transitions obligatoires et un autre ensemble de transitions interdites. Un calcul est acceptant si chaque suite de transitions soit se termine dans un état universel ou soit est infinie et appartient à un langage ω-régulier $Acc_{\mathcal{A}} \subset Q^{\omega}$ défini par l'automate $\mathcal{A}$. Ainsi, $\mathcal{A}$ spécifie localement des transitions obligatoires, possibles ou interdites, et globalement des comportements autorisés. Nous utiliserons la plupart du temps, pour formaliser l'ensemble $Acc_{\mathcal{A}}$ les *conditions de parité*. On associe alors à chaque état de $\mathcal{A}$ un entier naturel. Les suites infinies acceptantes sont celles où le plus grand entier apparaissant infiniment souvent est pair.

Pour définir formellement la sémantique des automates simples, nous utilisons la théorie des jeux. Ce sont les résultats de décidabilité de la théorie des jeux qui nous permettront de prouver la décidabilité de certains problèmes de contrôle. On associe à P et $\mathcal{A}$, le jeu $G(\mathcal{A}, P)$, appelé *jeu d'acceptation* qui représente l'ensemble des calculs possibles de $\mathcal{A}$ sur P. C'est un graphe muni de la condition infinitaire $Acc_{\mathcal{A}}$ qui associe chaque calcul de l'automate au comportement correspondant du processus. Un calcul est défini par une *stratégie* σ qui choisit, pour chaque état existentiel, la transition à

effectuer. Un processus P satisfait l'automate $\mathcal{A}$, noté $P \vDash \mathcal{A}$, si il existe une stratégie *gagnante*, c'est à dire si il existe σ tel que le sous-graphe engendré par σ correspond à un ensemble de suites de transitions terminant dans $Q^{\forall}$ ou appartenant à $Acc_{\mathcal{A}}$. La théorie des jeux nous assure qu'il est décidable de savoir si il existe une stratégie gagnante dans un jeu et que cette stratégie est calculable (voir, par exemple, [Buc83, GH82, Tho97, EJ91, AN01]). On peut donc décider si un processus satisfait une spécification définie par un automate simple.

Ce résultat de la théorie des jeux assure aussi la décidabilité de la satisfiabilité des automates simples, c'est à dire savoir si un automate a un modèle. En effet, comme le montre les auteurs dans [AVW03], on peut associer à un automate $\mathcal{A}$ un jeu $G(\mathcal{A})$ tel que les modèles de $\mathcal{A}$ soient en bijection avec les stratégie gagnantes de $G(\mathcal{A})$. Cependant, la construction du jeu $G(\mathcal{A})$ n'est possible ici que si $\mathcal{A}$ a une forme particulière, celle d'un automate *non-deterministe*. Un résultat fondamental de la théorie des automates, appelé parfois *théorème de simulation*, prouve que tout automate simple alternant est équivalent à un automate nondéterministe avec condition de parité que l'on peut construire (voir, par exemple, [AN01, Tho97, MS95]).

Nous expliciterons avec des automates simples les spécifications classiques régulières de la théorie du contrôle pour le système supervisé. En particulier, nous montrerons que le contrôle avec *événements forcés* est possible. Ce type de contrôle, introduit dans [GR87], consiste non pas seulement à interdire l'occurrence d'événements mais aussi à forcer le SED à effectuer certaines transitions. (Ce problème est modélisable car les transitions effectuées par les automates simples peuvent dépendre des propriétés locales des états du système supervisé.)

Nous présentons aussi certaines propriétés des automates simples, notamment le fait que les combinaisons booléennes (union, intersection, complémentation) d'ensembles de processus spécifiés par des automates simples sont définissables par des automates simples. On peut, par exemple, "accumuler" deux spécifications définis par les automates $\mathcal{A}$ et $\mathcal{B}$ en construisant un automate $\mathcal{A} \wedge \mathcal{B}$, dont les modèles sont les processus satisfaisant $\mathcal{A}$ et $\mathcal{B}$.

Nous avons souligné précédemment qu'un automate simple pouvait spécifier que certaines transitions d'un processus doivent exister. C'est principalement cette propriété qui permet de modéliser les contraintes de contrôlabilité d'un contrôleur par un automate simple. Par exemple, un automate simple $\mathcal{B}$ peut spécifier que tous ses modèles ont toujours des transitions pour les événements $A_{uc} \subseteq A$. Ainsi les contrôleurs satisfaisant $\mathcal{B}$ ne peuvent contrôler les événements de A_{uc}. Un automate simple peut aussi définir les événements

incontrôlables de façon *dynamique*, c'est à dire en fonction du comportement antérieur. On peut, par exemple, spécifier, qu'après une erreur e, un événement c devienne incontrôlable. On peut aussi définir un automate simple $\mathcal{B}$ dont les modèles sont les processus ayant, dans chaque état, toujours une transition par a *ou* par b. Dans ce dernier cas, les événements a et b ne peuvent être contrôlés en même temps.

La décidabilité du contrôle centralisé pour des spécifications avec des automates simples, c'est à dire le contrôle centralisé avec information totale, est prouvé dans [AVW03] par une opération de quotient. Avec un processus P et un automate $\mathcal{A}$, on peut construire un *automate simple* "quotient" de l'automate par le processus, noté $\mathcal{A}/P$ tel que :

$$Q \vDash \mathcal{A}/P \text{ si et seulement si } P \times Q \vDash \mathcal{A}$$

Ainsi, si un automate simple $\mathcal{B}$ définit les contraintes du contrôleur, les modèles de $\mathcal{A}/P \wedge \mathcal{B}$ sont *toutes* les solutions du problème de contrôle centralisé. Le problème du contrôle centralisé est donc réduit à la satisfiabilité d'un automate simple et donc à l'existence d'une stratégie gagnante dans un jeu associé. Nous verrons aussi que même si les automates $\mathcal{A}$ et $\mathcal{B}$ sont nondéterministes, l'opération de conjonction $\wedge$ rend alternant l'automate $\mathcal{A}/P \wedge \mathcal{B}$. On doit donc utiliser le théorème de simulation évoqué précédemment pour décider de la satisfiabilité de l'automate $\mathcal{A}/P \wedge \mathcal{B}$.

La décidabilité du problème du contrôle décentralisé pour des spécifications définis par des automates simples, est elle aussi décrite dans [AVW03] grâce à un opération de "quotient" de deux automates. Avec deux automates $\mathcal{A}$ et $\mathcal{B}$, on construit un automate simple "quotient" de $\mathcal{A}$ et $\mathcal{B}$, noté $\mathcal{A}/\mathcal{B}$ tel que :

$$P \vDash \mathcal{A}/\mathcal{B} \text{ si il existe } Q \vDash \mathcal{B} \text{ tel que } P \times Q \vDash \mathcal{A}$$

Nous rappellerons comment, avec une application successive de cette opération de quotient, le contrôle décentralisé est réduit à la satisfiabilité d'un automate simple.

La question de l'optimalité des solutions aux problèmes de contrôle ne sera pas traité ici. On pourra se référer aux travaux de [Rie03] et [PR05] qui montrent que ces problèmes peuvent aussi être réduits à la satisfiabilité de formule de μ-calcul modal et, dans le cas du contrôle avec information partielle, de son extension, le μ-calcul modal étendu.

Pour traiter les problèmes de contrôle avec information partielle nous introduisons des automates, les *automates étendus*, permettant de spécifier les contraintes d'observabilité du μ-calcul étendu.

Nous avons vu que les automates simples permettaient localement de spécifier un ensemble de transitions interdites et un ensemble de transitions obligatoires. Les automates étendus sont des automates simples à qui l'on permet, localement, de spécifier deux nouveaux types de contraintes. Le premier permet de spécifier qu'un processus, dans un état donné et par un événement fixé, reste dans le même état si cette transition existe. C'est ce qu'ont appelé les auteurs de [AVW03], les automates à boucles (loop automata). On peut donc spécifier qu'un contrôleur pour certains événements reste dans le même état et que par conséquent ces événements sont inobservables. Le deuxième type spécifie qu'un processus, dans un état donné et par deux événements fixés, atteint le même état si ces transitions existent. En introduisant ici ces contraintes nous permettons aux automates étendus de spécifier que des événements sont indiscernables. Nous explicitons, par exemple, comment fixer un ensemble d'événements inobservables ou une partition des événements indiscernables. Bien sûr, une version dynamique des contraintes d'observabilité est possible. On peut aussi spécifier qu'un événement inobservable est incontrôlable.

Nous montrons que le μ-calcul étendu reste un cadre décidable de spécifications. La sémantique et la satisfiabilté des automates étendus est aussi présentée sous forme de jeux. Pour un processus P et un automate étendu $\mathcal{A}$, le jeu d'acceptation $G(\mathcal{A}, P)$ est un graphe identique à ceux des automates simples auquel on a simplement ajouté certaines arêtes pour vérifier les propriétés locales d'observabilité. Le jeu $G(\mathcal{A})$, permettant de décider de la satisfiabilité, est beaucoup plus délicat à construire. Nous avions souligné, pour le cas des automates simples, que ce jeu n'était défini que pour les automates sans alternance, appelé nondéterministe. Cette forme particulière des automates est possible grâce au théorème de simulation. Nous allons ici introduire une notion d'automate étendu nondéterministe auquel il est plutôt aisé d'associer une jeu de "satisfiabilité" $G(\mathcal{A})$. Puis nous introduisons une extension du théorème de simulation prouvant qu'il est possible d'associer à tout automate étendu, un automate étendu nondéterministe d'expressivité équivalente. La démarche principale de cette preuve très technique est de parvenir, par une succession de changements d'écriture des automates étendus, à utiliser le théorème de simulation classique des automates simples.

La décidabilité du contrôle centralisé est assurée par l'opération de quotient $\mathcal{A}/P$ d'un automate simple $\mathcal{A}$ par un processus P et par la clotûre par opérations booléennes des automates étendus. On peut en effet construire un automate étendu $\mathcal{A}/P \wedge B$ où $\mathcal{B}$ est un automate étendu spécifiant les contraintes du contrôleur, dont les modèles sont toutes les solutions au problème du contrôle centralisé avec information partielle. Nous détaillons cependant l'opération plus générale de quotient $\mathcal{A}/P$ d'un automate étendu $\mathcal{A}$

par un processus P.

Nous détaillons aussi un peu plus l'expressivité des automates étendus. L'ensemble des modèles des automates étendus ne sont pas définis à bisimulation près comme c'est le cas pour les automates simples. En effet, cette extension permet, pour certaines transitions, de spécifier dans quel état doit être un processus. On pourrait envisager de définir les contraintes concernant les événements inobservables non pas en forcant un contrôleur à rester dans le même état, mais simplement à avoir les mêmes comportements qu'un événements inobservable se produisent ou non. Cela reviendrait à considérer tous les modèles en bisimulation avec ceux définis par un automate à boucles. Nous montrons que ces ensembles de modèles ne sont pas définissables par des automates simples. La conclusion est similaire pour le cas des événements indiscernables.

Nous avons aussi essayé d'étendre le μ-calcul modal pour spécifier que certains *comportements* sont inobservables ou indiscernables. Il s'agit non pas seulement d'avoir une information partielle sur les événements mais aussi sur les comportements.

Nous introduisons les *automates k-inobservables* et les *automates k,p-indiscernables* qui sont de nouvelles extensions des automates simples. Les automates k-inobservables peuvent spécifier localement qu'après un comportement de longueur k, dans A^k, un processus revient dans le même état. Les automates k,p-indiscernables peuvent spécifier localement qu'un processus, après un comportement u de longueur k, ($u \in A^k$) et comportement v de longueur p ($v \in A^p$), atteint le même état. Bien sûr, si $k = p = 1$, ce sont les automates avec événements inobservables et indiscernables. Nous prouvons que si $k \geq 3$ alors la satisfiabilité des automates k-inobservables est indécidable. De même, si $k \geq 2$ alors la satisfiabilité des automates k,p-indiscernables est indécidable.

Les preuves d'indécidabilité sont des réductions du problème de Post. Nous présentons une étude détaillée du problème de Post qui présente une caractérisation en termes de transitions dans un processus. Ce travail permet d'extraire une propriété telle que si une extension du μ-calcul modal permet de spécifier cette propriété, alors la satisfiabilité de cette extension est indécidable. Il s'agit d'une sorte de propriété de "commutativité" qui permet de spécifier qu'après les comportements ab et ba ($a, b \in A$), un processus atteint le même état. Dans cette extension les événements a et b doivent cependant pouvoir être discernés. Par exemple, avec les automates 3-inobservables, on peut spécifier que zab et zba sont inobservables. Ainsi, nécessairement, les comportements ab et ba atteingnent le même état.

Ces résultats nous permettent aussi de prouver certains résultats d'indiscernabilité du contrôle décentralisé avec information partielle que nous allons maintenant évoqué.

Avec les contraintes d'information partielle définies par des automates étendus, on peut définir le problème de contrôle décentralisé suivant.

Soit P un SED, $\mathcal{A}$ un automate simple et $\mathcal{B}_1, \mathcal{B}_2$ des automates étendus. Existe-t-il des contrôleurs Q_1 et Q_2 tel que $Q_1 \models \mathcal{B}_1$, $Q_2 \models \mathcal{B}_2$ et $P \times Q_1 \times Q_2 \models \mathcal{A}$?

Nous montrons que ce problème est indécidable même si $\mathcal{B}_1$ et $\mathcal{B}_2$ peuvent spécifier uniquement des événements inobservables ou uniquement des événements indiscernables. Notons que le cas où $\mathcal{B}_1$ et $\mathcal{B}_2$ sont des automates à boucle a déjà été prouvé dans [AVW03]. Si l'une des spécifications $\mathcal{B}_1$ ou $\mathcal{B}_2$ est un automate à boucle, l'indécidabilité du contrôle décentralisé est prouvée grâce à un réduction du problème de Post avec les travaux déjà présentés. Sans rentrer dans les détails, nous présentons des automates étendus particuliers $\mathcal{B}_1, \mathcal{B}_2$ avec lesquels on peut spécifier que le processus produit des contrôleurs $Q_1 \times Q_2$ satisfait la propriété de "commutativité" rendant l'extension du μ-calcul modal indécidable.

Le cas où les automates $\mathcal{B}_1$ et $\mathcal{B}_2$ expriment uniquement des contraintes d'indiscernabilité est traité différement. La preuve d'indécidabilité est une réduction du problème de l'arrêt d'une machine de Turing. Expliquons très brièvement notre démarche. Avec des automates particuliers, nous codons dans les transitions des contrôleurs des configurations d'une machine de Turing. On peut extraire de toutes ces configurations une suite infinie de configuration successives de la machine de Turing depuis l'état initial. De plus, ces configurations sont de taille bornée au moins par le nombre d'états des contrôleurs. Ceci permet, dans une partie très technique, de montrer que ce problème de contrôle est la réduction d'un problème indécidable, celui de savoir si une machine de Turing effectue ces calculs avec une mémoire finie.

Les spécifications permettant de prouver l'indécidabilité du contrôle décentralisé sont des restrictions importantes de l'expressivité des automates étendus. Dans les trois cas, les événements inobservables ou indiscernables sont des sous-ensembles d'événements fixés. De plus pour les automates à boucle les spécifications n'autorisent que les comportements finis, les conditions infinitaires sont vide $Acc = \emptyset$.

Le contôle décentralisé avec information partielle peut cependant être décidable dans des cas particuliers que nous allons maintenant détailler un peu plus.

Nous avons d'abord cherché à restreindre l'expressivité des automates étendus. Nous introduisons pour cela une notion d'automate étendu *déterministe*. Ces automates n'ont pas d'états existentiels, il ne peuvent effectuer qu'un unique calcul sur un processus. En terme de jeu, il n'y a qu'une seule stratégie possible. Cependant ces automates peuvent avoir plusieurs modèles car certaines transitions d'un automate étendu ne sont ni obligatoires ni interdites, elles sont juste *possibles*. Un automate étendu déterministe $\mathcal{A}$ a un modèle minimal $M_{\mathcal{A}}$; c'est à dire le modèle ayant le moins de comportements. Nous définissons aussi la notion d'automate étendu *conique*. Un automate étendu conique $\mathcal{A}$ est un automate étendu déterministe tel que, si un processus P est un modèle de $\mathcal{A}$, alors tous les processus dont les comportements sont inclus entre ceux du modèle minimal $M_{\mathcal{A}}$ et ceux de P, sont des modèles de $\mathcal{A}$.

Nous montrons que le contrôle décentralisé est décidable si les spécifications des contrôleurs sont des automates étendus déterministes et la spécification du système supervisé est un automate étendu conique. La preuve de ce résultat consiste à considérer certains contrôleurs "spéciaux" tels que si il existe une solution à ce problème de contrôle, alors ces contrôleurs "spéciaux" sont des solutions.

Nous pouvons avec les automates étendus déterministes exprimer les exemples de spécifications classiques pour les contrôleurs. Les automates étendus coniques semblent être une restriction très grande des spécifications du système supervisé. Nous montrons qu'il est tout de même possible d'exprimer que les comportements du système supervisé sont inclus (pour l'inclusion) entre deux langages finis réguliers clos par préfixes.

Nous étudions aussi un autre type de restrictions pour rendre le contrôle décentralisé décidable. Il s'agit d'une restricition implicite des communications entre les processus. Nous définissons une notion de produit de processus $P \times Q$ *bien synchronisé* où les transitions possibles dépendent des états que chacun des processus atteint. Plus présisément, dans le produit $P \times Q$ de deux processus, deux types de transitions sont autorisés. Soit les événements que distinguent P (ceux qui ne sont pas indiscernables) sont aussi distingués par Q, soit, si Q n'observe pas un événement a qui est observable par P (a n'est pas indiscernable pour P), alors tous les événements dans le même cas (observables par P mais pas par Q) doivent être indiscernables. On peut

dire, de façon grossière, que le processus Q doit avoir une information partielle plus grande que le processus P excepté pour un ensemble d'événements indiscernables.

Nous définissons alors un automate étendu $\mathcal{A}/syn/\mathcal{B}$ par une opération de quotient de deux automates étendus $\mathcal{A}$ et $\mathcal{B}$ tel que :

$P \vDash \mathcal{A}/syn/\mathcal{B}$ si il existe $Q \vDash \mathcal{B}$ tel que $P \times Q \vDash \mathcal{A}$ et $P \times Q$ bien synchronisé

Nous montrons, par des méthodes similaires à celles employées pour le quotient de deux automates simples, que cette opération de quotient appliquée plusieurs fois permet de trouver les solutions bien synchronisées d'un problème de contrôle donné. Notons que cette propriété de synchronisation n'est pas commutative. Lorsqu'on applique plusieurs fois ce quotient afin de construire, par exemple, un système supervisé $P \times Q_1 \times Q_2$, le produit $P \times Q_1$ et le produit $\big(P \times Q_1\big) \times Q_2$ sont bien synchronisés.

Nous montrons que cette opération prend en compte un cas présenté par S. Riedweg dans sa thèse [Rie03], ici appelé *observation chainée*. Dans ce cas, nous avons un certain nombre de contrôleurs $Q_1, Q_2, \ldots, Q_n$ tels que tous les événements inobservables pour Q_i sont inobservables pour Q_{i-1}. Nous montrons que cette propriété implique que les contrôleurs sont bien synchronisés.

Nous présentons aussi le cas du *pipeline* étudié par de nombreux auteurs (voir par exemple [PnRo90, MT01, MW03]). Dans ce cadre, le contrôleur Q_i "reçoit" les événements autorisés par le contrôleur Q_{i-1} et envoie ceux qu'il autorise au contrôleur Q_{i+1}. Nous montrons que c'est aussi un cas où les contrôleurs sont bien synchronisés.

0.2 Plan

La fin de cette partie I est consacrée à formaliser les notions de processus, de produit de processus et de bisimulation.

Puis, dans la partie II, nous étudions le cas du contrôle avec information totale, c'est à dire le cas où les contrôleurs ont une connaissance exacte du comportement antérieur du SED. Dans le chapitre 1, nous introduisons des spécifications avec les *automates simples*. Les calculs acceptants de ces automates sont formulés sous forme de *stratégie gagnante dans des jeux* définis dans les sections 1.3 et 1.4. Le théorème 1.3.5 assure la décidabilité de l'existence de stratégie gagnante dans ces jeux et leurs synthèses. Le théorème de simulation 1.5.3 est aussi présenté.

Au chapitre 2, dans la section 2.3, nous présentons différentes contraintes de contrôlabilité pour les contrôleurs. Les sections 2.4, 2.5, 2.6 et 2.7 pré-

sentent des exemples de spécifications du système supervisé exprimables par des automates simples. La décidabilité des problèmes de contrôle centralisé et décentralisé avec des spécifications définies par des automates simples est présentée dans la section 2.2.

Nous allons ensuite étudier dans la partie III les problèmes de contrôle avec information partielle.

Nous définissons dans le chapitre 3 les *automates étendus*. Nous montrons certaines contraintes classiques associées aux problèmes d'information (voir section 3.3). La décidabilité de la satisfiabilité des automates étendus est obtenue dans la section 3.5 grâce à l'extension du théorème de simulation aux automates étendus dans la section 3.4. La décidabilité du contrôle *centralisé* avec événements inobservables et indiscernables est présentée dans la section 3.6. Certaines propriétés des automates étendus utiles pour les problèmes de contrôles et représentant des extensions de celle des automates simples sont présentées aux sections 3.7 et 3.8.

Au chapitre 4, nous étudions le cas où les contrôleurs ont une information partielle non plus seulement sur les événements mais sur les comportements. Ces nouvelles contraintes nous amènent à définir de nouvelles extensions des automates simples dans la section 4.3 où l'on prouve que la satisfiabilité pour ce type d'automate est indécidable. Ce résultat est prouvé grâce aux travaux de décomposition du problème de Post de la section 4.2.

Dans la partie IV, nous revenons aux problèmes de contrôle avec événements inobservables et indiscernables. Nous étudions cette fois-ci le problème du contrôle *décentralisé*.

Nous montrons dans le chapitre 5 que ce problème est indécidable même si il n'y a que deux contrôleurs. Les sections 5.1, 5.2 et 5.3 présentent les trois cas possibles où chaque contrôleur n'a que des contraintes d'observabilité ou d'indiscernabilité.

Le chapitre 6 présente deux cas décidables. Celui de la section 6.1, est obtenu en effectuant des *restrictions sur les spécifications*. La section 6.2 est dédiée à l'étude d'une forme plus restreinte de *synchronisation des processus* décrite plus haut.

0.3 Systèmes à événements discrets

Nous formalisons maintenant la notion de processus.

Définition 0.3.1 Soit Λ un ensemble fini de *symboles propositionnels*. Un *processus* est un tuple $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ avec S un ensemble fini d'états

et e une fonction partielle de $S \times A$ dans S définissant les transitions du processus. L'état initial est l'état $s_0 \in S$ du processus et λ est une fonction de S dans 2^Λ définissant les propriétés locales des états. On note $PROC(A, \Lambda)$ l'ensemble des processus sur l'alphabet A avec leurs propriétés locales dans Λ.

Définition 0.3.2 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un processus. Nous notons $D_P(s)$, l'ensemble $\{a : e(s, a) \text{ est défini}\}$ et $N_P(s) = A - D_P(s)$ son complémentaire. Afin de simplifier les notations, nous étendons la fonction e à l'ensemble $S \times A^+$ comme suit : pour tout $s \in S$, $u \in A^+$ et $a \in A$, $e(s, ua) = e(e(s, u), a)$ si $e(s, u)$ et $e(e(s, u), a)$ sont définis. On dit alors qu'un état $s \in S$ est *accessible* si il existe $u \in A^*$ tel que $e(s_0, u) = s$. Si $\lambda : S \to \{\emptyset\}$, nous noterons $P = \langle A, S, s_0, e \rangle$.

Remarque 0.3.3 Les comportements finis d'un processus P, c'est à dire les éléments de l'ensemble :

$$\mathcal{L}(P) = \{u \in A^* : e(s_0, u) \text{ est défini}\}$$

définissent donc un langage régulier de mots finis clos par préfixes. P décrit aussi un langage de mots infinis :

$$\mathcal{L}^\omega(P) = \{a_1 a_2 \cdots a_n \cdots \in A^\omega : \text{pour tout } n, e(s_0, a_1 \cdots a_n) \text{ est défini}\}$$

Pour chaque $E \subseteq \Lambda$, on peut aussi définir un langage *marqué* :

$$\mathcal{L}_E(P) = \{u \in A^* : e(s_0, u) \text{ est défini et } \lambda(e(s_0, u)) \subseteq E\} \subseteq \mathcal{L}(P)$$

représentant les comportements finis de P finissant dans un état satisfaisant la propriété E.

0.4 Système supervisé et contrôleur

La supervision d'un SED consiste à modifier ses comportements car certains sont considérés comme insatisfaisants. Nous définirons dans le chapitre suivant un cadre pour spécifier les comportements souhaités d'un SED.

Dans la plupart des travaux, il s'agit de *restreindre* les comportements d'un SED. On considère en effet qu'il est possible de contraindre l'occurrence de certains évènements, appelés évènements *contrôlables* (les autres évènements étant incontrôlables). Le système supervisé (ou contrôlé) est alors un sous-ensemble des comportements d'un SED induit par un contrôleur qui supprime l'occurence de certains évènements contrôlables. Comme dans [AVW03], le système supervisé est ici défini comme le produit de deux processus, le SED et le contrôleur.

Définition 0.4.1 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ et $P' = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ deux processus. Leur produit $P \times P'$ est un processus $\langle A, \Lambda, S \times S', (s_0, s'_0), e \times e', \lambda'' \rangle$ où $\lambda''(s, s') = \lambda(s) \cup \lambda'(s')$ et $e \times e'((s, s'), a)$ est défini et égale à $(e(s, a), e'(s', a))$ si et seulement si $a \in D_P(s) \cap D_{P'}(s')$.

Ainsi, si P est le SED et P' le contôleur, dans l'état (s, s') du système supervisé $P \times P'$, le contrôleur supprime l'occurence d'un évènement $a \in A$ si la transition $e'(s', a)$ n'est pas défini. Notons qu'avec cette définition le contrôle peut être *dynamique* car les évènements supprimés par le contrôleur peuvent dépendre du comportement antérieur du SED.

Nous définissons maintenant une relation sur les processus qui va nous permettre de comparer les comportements des processus par la suite.

Définition 0.4.2 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ et $P' = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ deux processus. On note $P \leq P'$ si, pour tout état accessible (s, s') depuis (s_0, s'_0) du produit $P \times P'$ (i.e. il existe $u \in A^*$ tel que $e \times e'((s_0, s'_0), u) = (s, s')$), si $e(s, a)$ est défini alors $e'(s', a)$ est défini et, de plus, $\lambda(s) \supseteq \lambda'(s')$.

Proposition 0.4.3 $\leq$ est un préordre.

Preuve : Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$, $P' = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ et $P'' = \langle A, \Lambda, S'', s''_0, e'', \lambda'' \rangle$ des processus. $\leq$ est évidemment réflexive. Montrons que $\leq$ est transitive. Supposons que $P \leq P'$ et $P' \leq P''$. Soit (s, s'') un état de $P \times P''$ accessible depuis (s_0, s''_0) avec le chemin $u \in A^*$ tel que $e(s, a)$ est défini. Puisque $P \leq P'$, il existe un état s' tel que $e'(s'_0, u)$ est défini et égale à s' et $e'(s', a)$ est défini. Puisque $P' \leq P''$, $e''(s'', a)$ doit être aussi défini. De plus, $\lambda(s) \supseteq \lambda'(s') \supseteq \lambda''(s'')$. Donc $P \leq P''$. □

Un outil classique pour comparer des automates est la notion de bisimulation que nous rappelons ici pour les processus.

Définition 0.4.4 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ et $P' = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ deux processus. On dit que P et P' sont en *bisimulation* ou *bisimilaires*, et on note $P =_b P'$, si il existe une relation $R \subseteq S \times S'$ tel que $(s_0, s'_0) \in R$ et, pour tout $(s, s') \in R$ et $a \in A$, on a $\lambda(s) = \lambda'(s')$ et :

- Si il existe s_1 tel que $e(s, a) = s_1$, alors il existe s'_1 tel que $e'(s', a) = s'_1$ et $(s_1, s'_1) \in R$.
- Si il existe s'_1 tel que $e'(s', a) = s'_1$, alors il existe s_1 tel que $e(s, a) = s_1$ et $(s_1, s'_1) \in R$.

Puisque les processus sont des automates déterministes, on peut aisément vérifier le lemme 0.4.5.

Lemme 0.4.5 P et P' sont bisimilaires si et seulement si $P \leq P'$ et $P' \leq P$.

Deuxième partie

Contrôle avec information totale

Chapitre 1

Spécifications : μ-calcul

Nous allons dans ce chapitre définir un cadre identique à celui défini dans [AVW03] pour les spécifications des systèmes supervisés et la contrôlabilité des évènements. Il s'agit de définir des ensembles de processus grâce au *μ-calcul modal* introduit par [Koz83] (voir aussi [AN01]). Le μ-calcul modal nous permet de prendre en compte les comportements infinis des processus et toutes les spécifications régulières. C'est notamment une généralisation des comportement finis réguliers étudiés dans [RW89] mais aussi des logiques temporelles CTL [CE81] et CTL^* [EH86]. Ces dernières étant dédiées le plus souvent à l'étude des systèmes réactifs [MP92], c'est à dire à des systèmes modélisant, par leurs comportements infinis, l'interaction permanente avec leur environnement.

Comme dans [AVW03], au lieu de définir la logique du μ-calcul modal et son interprétation sur les processus, nous utilisons le formalisme des automates, très souvent utilisé pour construire des procédures de décision des logiques. Nous définissons dans la partie 1.1 suivante la notion d'automate simple correspondant à la notion d'automate alternant sur les arbres de degré borné (par l'alphabet des évènements A). La notion de calcul de ces automates est étendue aux processus par l'intermédiaire de jeux, introduits dans la partie 1.3, qui donne ainsi la sémantique des automates simples dans la partie 1.4. On peut remarquer que ces automates doivent vérifier l'existence des arêtes (ici les transitions) puisque, si l'on interprète un processus comme un graphe étiqueté, les noeuds des processus peuvent avoir des degrés différents.

L'équivalence d'expressivité entre automates simples et μ-calcul modal sur les processus ne sera pas démontrée ici. On pourra se référer à [Wal01] ainsi qu'aux travaux [AN01, ES89, Rie03].

1.1 Définition d'un automate simple

Définition 1.1.1 Un *automate simple* sur les processus est un tuple :

$$\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$$

où $Q^{\exists}$ et $Q^{\forall}$ forment une partition de l'ensemble fini des états Q en états existentiel et universel. $Acc \subseteq Q^{\omega}$ est un ensemble de suites infinies d'états que nous appelons la *condition d'acceptation*. L'ensemble A des actions est l'ensemble des directions dans lesquelles l'automate effectue ses calculs. L'état q_0 est l'état initial de l'automate et δ est la fonction de transition définie comme suit :

$$\delta : Q \times 2^{\Lambda} \to \mathcal{P}\Big(\big(A \times Q\big) \cup \big(\{\varepsilon\} \times Q\big) \cup \big(A \times \{\to, \not\to\}\big) \Big)$$

On note $AUTO(A, \Lambda)$ l'ensemble des automates simples sur l'alphabet A avec propriétés locales sur Λ.

Notation 1.1.2 Si pour tout $q \in Q$ et pour tout $E, E' \subseteq \Lambda$, $\delta(q, E) = \delta(q, E')$, on notera $\mathcal{A} = \langle A, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$ et

$$\delta : Q \to \mathcal{P}\Big(\big(A \times Q\big) \cup \big(\{\varepsilon\} \times Q\big) \cup \big(A \times \{\to, \not\to\}\big) \Big)$$

C'est le cas, entre autre, lorsque $\Lambda = \emptyset$.

Notation 1.1.3 Pour simplifier les écritures, nous noterons $\to_a$ et $\not\to_a$ au lieu respectivement de $(a, \to)$ et $(a, \not\to)$.

La sémantique d'un automate simple est décrite dans la partie 1.4 grâce à la notion de jeu défini dans la partie 1.3. Mais nous précisons tout d'abord la condition d'acceptation ou *condition infinitaire Acc*.

1.2 Conditions d'acceptation *Acc*

Nous rappelons ici les principales formes de condition infinitaire *Acc* dans la théorie des automates. Nous supposerons par la suite que *Acc* est l'une de ces conditions. Nous montrons dans cette partie qu'il est possible de définir ces conditions d'acceptations par des combinaisons booléennes de conditions simples.

Définition 1.2.1 Soit $p = s_0 s_1 s_2 \cdots s_n \cdots \in S^{\omega}$. On note $Inf(p)$ l'ensemble des états du chemin p qui apparaissent infiniment souvent.

- $Acc \subseteq S^\omega$ est une *condition de Büchi* si il existe $F \subseteq S$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \cap F \neq \emptyset\}$$

- $Acc \subseteq S^\omega$ est une *condition de Co-Büchi* si il existe $F \subseteq S$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \cap F = \emptyset\}$$

- $Acc \subseteq S^\omega$ est une *condition de Rabin* si il existe des couples (R_1, G_1), (R_2, G_2),...,$(R_n, G_n) \subseteq S \times S$ tel que :

$$Acc = \{p \in S^\omega : \exists i. Inf(p) \cap R_i = \emptyset \text{ et } Inf(p) \cap G_i \neq \emptyset\}$$

- $Acc \subseteq S^\omega$ est une *condition de Streett* si il existe des couples (R_1, G_1), (R_2, G_2),...,$(R_n, G_n) \subseteq S \times S$ tel que :

$$Acc = \{p \in S^\omega : \forall i. Inf(p) \cap R_i = \emptyset \text{ ou } Inf(p) \cap G_i \neq \emptyset\}$$

- $Acc \subseteq S^\omega$ est une *condition de parité (ou de Mostowski)* si il existe une fonction $r : S \to \mathbb{N}$ tel que :

$$Acc = \{p \in S^\omega : Max_{q \in Inf(p)}(r(q)) \text{ est pair}\}$$

- $Acc \subseteq S^\omega$ est une *condition de Muller* si il existe $\mathcal{F} \subseteq 2^S$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \in \mathcal{F}\}$$

Définition 1.2.2 On dit qu'un automate simple est un automate simple de Co-Büchi, de Büchi, de Streett, etc... si Acc est une condition de Co-Büchi, de Büchi, de Streett, etc...

Si $Acc = \emptyset$ on dit que l'automate est à condition d'accessibilité.

Définition 1.2.3 Soit $\mathcal{B}(S)$ l'ensemble des conditions de parité de la forme $r : S \to \{1, 2\}$, appelé parité de Büchi. Soit $\mathcal{C}(S)$ l'ensemble des conditions de parité de la forme $r : S \to \{0, 1\}$, appelé parité de Co-Büchi.

Soit I et J deux ensembles finis et pour tout $i \in I$ et $j \in J$, $X_{i,j}$ des éléments de $(\mathcal{B}(S) \cup \mathcal{C}(S))$. On note $Inf(p) \vDash \bigvee_i \bigwedge_j X_{i,j}$ si il existe $i \in I$ tel que pour tout $j \in J$, $Max_{q \in Inf(p)}(X_{i,j}(q))$ est pair.

$Acc \subseteq S^\omega$ est une *disjonction de conjonctions de parité de Büchi et de Co-Büchi* si il existe un ensemble fini de fonctions $\{X_{i,j}\} \subset (\mathcal{B}(S) \cup \mathcal{C}(S))$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \vDash \bigvee_i \bigwedge_j X_{i,j}\}$$

Proposition 1.2.4

- *Acc* est une condition de Street avec n paires si et seulement si il existe $b_i \in \mathcal{B}(S)$ et $c_i \in \mathcal{C}(S)$ pour $i = 1, 2, \ldots, n$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \vDash \bigwedge_{i=1}^{n} (c_i \vee b_i)\}$$

- *Acc* est une condition de Rabin avec n paires si et seulement si il existe $b_i \in \mathcal{B}(S)$ et $c_i \in \mathcal{C}(S)$ pour $i = 1, 2, \ldots, n$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \vDash \bigvee_{i=1}^{n} (c_i \wedge b_i)\}$$

- *Acc* est une condition de parité dans $\{0, 1, \ldots, 2n\}$ si et seulement si il existe $b_i \in \mathcal{B}(S)$ et $c_i \in \mathcal{C}(S)$ pour $i = 1, 2, \ldots, n$ tel que :

$$Acc = \{p \in S^\omega : Inf(p) \vDash b_n \vee (c_n \wedge (b_{n-1} \vee (c_{n-1} \wedge \cdots \wedge (b_1 \vee c_1) \cdots)))\}$$

- *Acc* est une condition de Muller si et seulement si *Acc* est une disjonction de conjonctions de parité de Büchi et de Co-Büchi.

Preuve : Dans le cas des conditions de Rabin et de Street, on identifie R_i à $c_i^{-1}(1)$ et G_i à $b_i^{-1}(2)$.

Dans le cas d'une condition de parité r, on identifie $r^{-1}(2i)$ à l'ensemble des $q \in b_i^{-1}(2)$ tel que, pour tout $j > i$, $q \notin b_j^{-1}(2)$ et on identifie . $r^{-1}(2i-1)$ à l'ensemble des $q \in c_i^{-1}(1)$ tel que, pour tout $j > i$, $q \notin c_j^{-1}(1)$

Une condition de Muller $\mathcal{F} \subset 2^S$ est équivalente à la condition :

$$\bigvee_{E \subseteq \mathcal{F}} c_E \wedge \bigwedge_{q \in E} b_q \text{ où } c_E(q) = \begin{cases} 0 & \text{si } q \in E \\ 1 & \text{sinon} \end{cases} \quad \text{et } b_q(q') = \begin{cases} 2 & \text{si } q = q' \\ 1 & \text{sinon} \end{cases}$$

Si *Acc* est une disjonction de conjonctions de parité de Büchi et de Co-Büchi $\bigvee_i \bigwedge_j X_{i,j}$ alors on peut l'identifier à la condition de Muller $\mathcal{F}$ suivante. $E \in \mathcal{F}$ si et seulement si il existe i tel que pour tout j, on a

$$\begin{aligned} E \subset X_{i,j}^{-1}(0) \quad &\text{si} \quad X_{i,j} \in \mathcal{C}(S) \\ E \cap X_{i,j}^{-1}(2) \neq \emptyset \quad &\text{si} \quad X_{i,j} \in \mathcal{B}(S) \end{aligned}$$

□

1.3 Jeu, partie et stratégie

Une des façons les plus simples de formaliser la notion de calcul et d'acceptation d'un automate est d'introduire la notion de jeu que nous définissons dans cette section.

Définition 1.3.1 Un jeu G est un tuple $\langle V_0, V_1, Edge, Acc\rangle$ où $Edge \subset (V_0 \cup V_1)^2$ et $Acc \subset (V_0 \cup V_1)^\omega$ est l'ensemble définissant la condition de gain. $\langle V_0 \cup V_1, Edge\rangle$ est un graphe dont les sommets sont partitionnés entre ceux du joueur 0 et ceux de joueur 1. On dit qu'un sommet v' est successeur de v si $(v, v') \in Edge$.

Définition 1.3.2 Une partie entre les joueur 0 et 1 depuis un sommet $v \in V = V_0 \cup V_1$ est définit comme suit : si $v \in V_i$ alors le joueur i choisi un successeur et la même règle s'applique au successeur de v définissant ainsi une partie qui s'arrête seulement si l'un des deux joueurs ne peut choisir un successeur. Le joueur qui ne peut pas choisir un successeur a perdu. Le résultat d'une partie infinie est un chemin infini $v_0v_1v_2\cdots \in V^\omega$. Cette partie est gagnante pour le joueur 0 si $v_0v_1v_2\cdots \in Acc$ et gagnante pour le joueur 1 sinon.

Définition 1.3.3 Une *stratégie* v pour le joueur i est une fonction qui associe à toute suite finie de sommets $\vec{v}v \in V^+$ finissant dans un sommet $v \in V_i$ un successeur $s(\vec{v}v)$. Un stratégie s est dite *positionnelle* si, pour tout $\vec{v}, \vec{w} \in V^*$ et $v \in V_i$, on a $s(\vec{v}v) = s(\vec{w}v)$. Une stratégie positionnelle est alors identifiée à une fonction $s : V_i \to V$.

Une partie *compatible* avec une stratégie s pour le joueur i est une suite $v_0v_1v_2\cdots$ tel que, pour tout $v_j \in V_i$, $v_{j+1} = s(v_j)$. La stratégie s est *gagnante* pour le joueur i depuis un sommet v si toutes les parties depuis v compatibles avec s sont gagnantes pour le joueur i. Un sommet v est gagnant si il existe une stratégie gagnante depuis v.

Définition 1.3.4 Une stratégie avec l'histoire H pour le joueur i est un triplet $\langle \sigma, hist, h_0\rangle$ tel que $\sigma : V_i \times H \to V$, $hist : H \times V \to H$ et $h_0 \in H$. On note $hist^* : H \times V^* \to H$ l'extension de $hist$ tel que, pour tout $h \in H$, $\vec{v} \in V^*$ et $v \in V$,

$$hist^*(h, \varepsilon) = h \quad \text{et} \quad hist^*(h, \vec{v}v) = hist^*(hist(h, \vec{v}), v)$$

On dit qu'une statégie avec l'histoire H est à mémoire finie si H est un ensemble fini.

Le rôle de h_0 et de *hist* est d'abstraire certaines informations tel qu'à chaque suite $\vec{v} \in V^*$ on associe la mémoire $hist^*(h_0, \vec{v})$. La fonction de choix σ définit alors une stratégie $s(\vec{v}v) = \sigma(v, hist^*(h_0, \vec{v}))$. Si il n'y a pas d'ambiguïté, par abus de langage, on appellera σ la stratégie. Nous récapitulons maintenant un résultat classique de la théorie des jeux (voir, par exemple, [Buc83, GH82, Tho97, EJ91, AN01]).

Théoreme 1.3.5 Soit un jeu G avec la condition de gain *Acc*.

Si *Acc* est une condition de Muller (voir définition 1.2.1), alors G est déterminé, i.e., tout sommet est gagnant pour l'un des deux joueurs. De plus, il est décidable de savoir pour quel joueur un sommet est gagnant.

Si *Acc* est une condition de parité, alors il existe une stratégie gagnante positionnelle.

1.4 Jeu d'acceptation d'un automate simple

Définition 1.4.1 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate simple et $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus.

Le *jeu d'acceptation* $G(\mathcal{A}, P) = \langle V_0, V_1, Edge, Acc_G\rangle$ est un jeu tel que :

- L'ensemble V_0 de sommets pour le joueur 0 est $(Q^\exists \times S) \cup \{\bot\}$.
- L'ensemble V_1 de sommets pour le joueur 1 est $(Q^\forall \times S) \cup \{\top\}$.
- Depuis chaque sommet (q, s), et pour tout $(a, q') \in \delta(q, \lambda(s))$ il y a une arête dans *Edge* vers (q', s) si $a = \varepsilon$ et vers $(q', e(s, a))$ si $a \in A$ et $e(s, a)$ est défini.
- Depuis chaque sommet (q, s) et pour tout $\rightarrow_a \in \delta(q, \lambda(s))$ il y a une arête dans *Edge* vers $\bot$ si $e(s, a)$ n'est pas défini et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\nrightarrow_a \in \delta(q, \lambda(s))$, il y a une arête dans *Edge* vers $\bot$ si $e(s, a)$ est défini et vers $\top$ sinon.
- La condition de gain Acc_G est l'ensemble des suites infinies d'arêtes de la forme $(q_0, s_0)(q_1, s_1)\ldots$ tel que $q_0q_1\ldots$ est dans *Acc*, i.e., appartient à la condition d'acceptation de l'automate.

Intuitivement, le jeu d'acceptation $G(\mathcal{A}, P)$ représente tous les calculs possibles de l'automate $\mathcal{A}$ sur le processus P. Il existe par ailleurs, des arêtes spéciales vers la position perdante $\bot$ et vers la position gagnante $\top$, qui permettent de vérifier l'existence de certaines transitions dans le processus P.

Définition 1.4.2 On appelle *position initiale* du jeu $G(\mathcal{A}, P)$ la position (q_0, s_0), noté v_0.

On dit que P est un *modèle* de $\mathcal{A}$ s'il existe une stratégie gagnante depuis la position initiale du jeu $G(\mathcal{A}, P)$.

Le *langage reconnu* par $\mathcal{A}$, noté $Mod(\mathcal{A})$, est l'ensemble des modèles de $\mathcal{A}$. Par la suite, nous noterons souvent $P \vDash \mathcal{A}$ au lieu de $P \in Mod(\mathcal{A})$.

1.5 Propriétés des automates simples

Dans cette partie nous rappelons certaines propriétés des automates simples qui nous seront utiles.

1.5.1 Théorème de simulation et satisfiabilité

Définition 1.5.1 Un automate simple $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ est *biparti* si, pour tout $E \subseteq \Lambda$, on a :
- Pour tout $q \in Q^{\exists}$, $\delta(q, E) \subseteq \{\varepsilon\} \times Q^{\forall}$.
- Pour tout $q \in Q^{\forall}$, $\delta(q, E) \subseteq A \times Q^{\exists} \cup A \times \{\rightarrow, \nrightarrow\}$
- De plus, on suppose que l'état initial est existentiel (i.e. $q_0 \in Q^{\exists}$).

Définition 1.5.2 Un automate simple $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ est *nondéterministe* si il est biparti et pour tout $E \subset \Lambda$, on a :

$$\text{Si } (a, q_1), (a, q_2) \in \delta(q, E) \text{ alors } q_1 = q_2.$$

Intuitivement, une transition depuis un état existentiel est une disjonction d'états universels qui représentent les "règles" possibles. Une transition depuis un état universel est une conjonction de contraintes locales (qui doivent être satisfaites) et de transitions de la forme (a, q) (où q est un état existentiel). Une transition (a, q) revient à spécifier "qu'après une $a - transition$ le processus doit satisfaire l'une des règles contenu dans $\delta(q)$". Le théorème suivant, appelé parfois théorème de simulation, est l'un des résultats majeurs de la théorie des automates et μ-calcul modal (voir, par exemple, [AN01, Tho97, MS95]).

Théoreme 1.5.3 Tout automate simple est équivalent à un automate nondéterministe à parité que l'on peut effectivement construire.

Ce résultat permet, comme il est prouvé dans [AVW03], d'établir le théorème de simulation suivant :

Théoreme 1.5.4 Pour tout automate simple $\mathcal{A}$, on peut construire un jeu $G(\mathcal{A})$ avec une position initiale v_0 tel que :

- A chaque stratégie gagnante à mémoire finie depuis v_0 dans $G(\mathcal{A})$ on peut faire correspondre un modèle de $\mathcal{A}$.
- De plus, tous les modèles de $\mathcal{A}$, à bisimulation près (voir définition 0.4.2), peuvent être obtenus de cette manière.

La section 3.5.2 propose un théorème de simulation et une preuve très similaire à celle de [AVW03] mais pour une définition d'automate plus générale. Ce théorème a pour corollaire la décidabilité de la satisfiabilité des automates simples (définie ci-dessous).

Définition 1.5.5
- Soit $\mathcal{A}$ un automate simple. On dit que $\mathcal{A}$ est *satisfiable* si $\mathcal{A}$ a un modèle.
- On appelle *problème de satisfiabilité des automates simples* le problème suivant : Soit $\mathcal{A}$ un automate simple. $\mathcal{A}$ est-il satisfiable ?

Corollaire 1.5.6 Le problème de satisfiabilité des automates simples est décidable. De plus, si un automate est satisfiable on peut effectivement construire un modèle de l'automate.

Preuve : Puisque l'existence d'une stratégie gagnante dans un jeu est décidable (voir théorème 1.3.5) et que cette stratégie est calculable, il est décidable de savoir si un automate simple nondéterministe $\mathcal{A}$ est satisfiable. De plus, si c'est le cas, on peut effectivement trouver un modèle P de $\mathcal{A}$. □

1.5.2 Expressivité et opérations des automates simples

Proposition 1.5.7 Les automates simple de Rabin, de Streett, à parité, de Muller ont les mêmes familles de modèles.

Preuve : On peut facilement vérifier que les conditions de parité sont des cas particuliers de conditions de Rabin et de Streett et que celles-ci sont elles-même des conditions particulières de Muller. La transformation d'un automate simple de Muller en un automate simple à parité utilise la technique LAR (last appearance record) (voir par exemple [Tho96]). □

Remarque 1.5.8 On peut identifier les automates d'arbres binaires infinis Σ-étiqueté à la sous-famille d'automates simples $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ tel que $\mathcal{A}$ est nondéterministe, $A = \{0, 1\}$, $\Sigma = 2^{\Lambda}$ et pour tout $q \in Q^{\forall}$ et $E \subseteq \Lambda$, on a $\rightarrow_0, \rightarrow_1 \in \delta(q, E)$.

Remarque 1.5.9 On peut aussi identifier les automates de mots infinis sur l'alphabet A à la sous-famille d'automates simples, noté $AUTO_{mots}(A, \emptyset)$, $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$ tel que $\mathcal{A}$ est nondéterministe, $\Lambda = \emptyset$ et pour tout $q \in Q^{\forall}$, il existe $a \in A$ tel que $\{\twoheadrightarrow_x : x \in A - \{a\}\} \in \delta(q)$.

On peut donc définir les sous-ensembles *ω-réguliers*. Ainsi $L \subset A^{\omega}$ est ω-régulier si il existe un automate simple de Büchi $\mathcal{A} \in AUTO_{mots}(A, \emptyset)$ tel que

$$L = \{w : \mathcal{L}^{\omega}(P) = \{w\} \text{ et } P \vDash \mathcal{A}\}$$

(Un résultat classique est que les automates (nondéterministes) de mots infinis avec condition de Büchi, Rabin, Streett, Mostowski et Muller définissent les mêmes langages, i.e. les langages ω-réguliers.)

Définition 1.5.10

- On dit que l'ensemble $AUTO(A, \Lambda)$ est *clos par union* si pour tous $\mathcal{A}, \mathcal{B} \in AUTO(A, \Lambda)$, il existe un automate simple dont l'ensemble des modèles est $Mod(\mathcal{A}) \cup Mod(\mathcal{B})$.
- On dit que $AUTO(A, \Lambda)$ est *clos par intersection* si pour tous $\mathcal{A}, \mathcal{B} \in AUTO(A, \Lambda)$, il existe un automate simple dont l'ensemble des modèles est $Mod(\mathcal{A}) \cap Mod(\mathcal{B})$.
- On dit que $AUTO(A, \Lambda)$ est *clos par complémentation* si pour tout $\mathcal{A} \in AUTO(A, \Lambda)$, il existe un automate simple dont l'ensemble des modèles est $PROC(A, \Lambda) - Mod(\mathcal{A})$.

Théoreme 1.5.11 L'ensemble des automates simples est clos par union, intersection et complémentation.

La preuve du théorème 1.5.11 est très similaire à celle du théorème 3.8.13 qui présentent les mêmes propriétés pour une définition d'automate plus générale.

Notation 1.5.12 Soit $\mathcal{A}, \mathcal{B} \in AUTO(A, \Lambda)$. On note $\mathcal{A} \vee \mathcal{B}$, $\mathcal{A} \wedge \mathcal{B}$ et $\mathcal{A}^C$ des automates simples quelconques dont les modèles sont respectivement l'union des modèles de $\mathcal{A}$ et $\mathcal{B}$, leurs intersection et le complément de $Mod(\mathcal{A})$ dans $PROC(A, \Lambda)$.

Nous finissons cette partie par rappeler un résultat d'expressivité de ces automates (voir [AN01] et les références associées).

Lemme 1.5.13 Soit $\mathcal{A}$ un automate simple et P et P' deux processus bisimilaires. $P \vDash \mathcal{A}$ si et seulement si $P' \vDash \mathcal{A}$.

Preuve : Soient $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$, $P' = \langle A, \Lambda, S', s'_0, e', \lambda\rangle$ et $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$. Supposons que $P \vDash \mathcal{A}$. Le cas $P' \vDash \mathcal{A}$ est identique puisque la bisimulation est une propriété symétrique. On suppose (voir théorème 1.5.3) que $\mathcal{A}$ est nondéterministe à parité. De plus, on suppose que pour tout $q \in Q^\forall$ et $E \subset \Lambda$ on a :

$$(a, q_1), (b, q_2) \in \delta(q, E) \text{ implique } q_1 \neq q_2.$$

Il suffit pour cela de dupliquer certains états de $\mathcal{A}$.

Puisque $\mathcal{A}$ est à parité, il existe une stratégie positionnelle gagnante $\sigma : Q^\exists \times S \to Q^\forall \times S$ dans le jeu $G(\mathcal{A}, P)$.

Construisons une stratégie gagnante dans le jeu $G(\mathcal{A}, P')$. Soit la stratégie (voir définition 1.3.4) $\sigma' : (Q^\exists \times S') \times H \to Q^\forall \times S'$ où $H = S$ avec la fonction d'histoire $hist : H \times (Q \times S') \to H$ et $h_0 = s_0$ définit comme suit.

Soit une position $(q, s) \in Q^\exists \times S$ où $\sigma(q, s)$ est défini et égale à (q_1, s) et tel qu'il existe $u \in A^*$ tel que $s = e(s_0, u)$. Alors, pour $s' = e'(s'_0, u)$, on a :

- $\sigma'((q, s'), s)$ est défini et égale à (q_1, s') et $hist(s, (q_1, s')) = s$
- Si il existe une arête de (q_1, s') vers (q_2, s'_2) dans $G(\mathcal{A}, P')$, alors $hist(s, (q_2, s'_2)) = e(s, a)$ où $(a, q_2) \in \delta(q_1, \lambda(s'))$.

Remarquons d'abord que, comme $P \leq P'$ (voir lemme 0.4.5), si $e(s_0, u)$ est défini, alors $e'(s'_0, u)$ est défini. De plus, comme $P' \leq P$, si $e'(s', a) = e(s'_0, ua) = s'_2$ alors $e(s, a) = e(s_0, ua)$ est défini. Donc σ' est bien défini.

On peut alors vérifier que la projection sur Q d'une partie compatible (voir définition 1.3.3) avec σ' et $hist$ dans le jeu $G(\mathcal{A}, P')$ coïncide avec la projection sur Q d'une partie compatible avec σ. Donc $P' \vDash \mathcal{A}$. □

1.6 Conclusion

Pour conclure ce chapitre, les automates simples permettent de décrire de nombreuses classes de processus comme ceux décrits par les automates de mots finis et infinis ainsi que les automates d'arbres. Ils permettent plus particulièrement de spécifier, localement, l'existence de certaines transitions des processus (pouvant dépendre de comportements antérieurs finis) et, globalement, des comportements infinis (sur les mots et les sous-arbres). Le nondéterminisme de ces automates nous permet, entre autre, de spécifier des unions de comportements disjoints. Par exemple l'union de l'ensemble des processus ayant des transitions par un événement a mais pas par un événement b et son "opposé", celui ayant des transitions par b mais pas par a.

L'expressivité générale de ces automates dépend de la condition d'acceptation qui induit une hiérarchie infinie stricte (voir [N97] et aussi [A99],

[Wal01]). Cependant, dans les exemples présentés dans ce travail, les automates ne dépasserons pas l'expressivité des automates à condition de parité dans $\{0, 1, 2\}$.

Nous allons fréquemment utiliser le théorème de simulation 1.5.3. C'est un outil essentiel pour décider la satisfiabilité des automates simples mais aussi pour simplifier les constructions d'automates. Un automate alternant ayant n états peut se transformer en automate nondéterministe ayant $2^{\mathcal{O}(nlogn)}$ états (l'augmentation du nombre de parités est linéaire) voir [EJ88], [MS95] et [Tho97].

C'est le théorème 1.3.5, permettant de décider s'il existe une stratégie gagnante dans un jeu et, ainsi, de décider la satisfiabilité des automates simples, qui sera ici l'outil principal pour établir des résultats de décidabilité. Dans le cas des automates nondéterministes à parité, trouver une stratégie gagnante est un problème dans $NP \cap co - NP$. Une borne supérieure est $\mathcal{O}(n^{(k/2)+1})$ où n est le nombre d'états de l'automate et k la plus grande parité (voir [AN01] et [VJ00]). Un algorithme *probabiliste* proposé dans [BSV03] donne une complexité $n^{\mathcal{O}(\sqrt{n/\log n})}$. Plus récemment, dans [JPZ06] un algorithme *déterministe*, basé sur une méthode très différente, obtient une complexité sous-exponentielle $n^{\mathcal{O}(\sqrt{n})}$ (La complexité est $n^{\mathcal{O}(\sqrt{n/\log n})}$ si le degré sortant de tous les sommets du graphe est borné).

Chapitre 2

Décidabilité du contrôle

Grâce aux automates simples décris au chapitre précédent, nous reformulons les problèmes de contrôle définis dans la partie I. Nous rappelons ici les opérations de *quotients* introduits dans [AVW03] qui permettent la décidabilité du contrôle avec information totale. Ce chapitre se termine par l'exposition de certains problèmes de contrôle plus spécifiques.

2.1 Définition des problèmes de contrôle

Nous formulons le problème de contrôle *décentralisé* avec information totale, noté **PCD**, comme suit :

Soit un processus P et des automates simples $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_n$ et $\mathcal{A}$. Existe-t-il des contrôleurs $Q_1, \ldots, Q_n$ tels que, $Q_i \vDash \mathcal{B}_i$ et $P \times Q_1 \times \cdots \times Q_n \vDash \mathcal{A}$? Et le problème de la synthèse est de construire les contrôleurs, si ils existent.

Le problème de contrôle *centralisé* avec information totale, noté **PCC**, est le cas particulier du **PCD** où $n = 1$.

2.2 Décidabilité : opérations de quotient

Les problèmes **PCC** et **PCD** sont décidables. Ce sont des conséquences des deux théorème suivants. Nous donnons des preuves de ces théorèmes pour une définition d'automate plus générale dans les parties 3.6 et 6.3.

Théoreme 2.2.1 Soit $\mathcal{A}$ un automate simple et P un processus. Il existe un automate simple $\mathcal{A}/P$ tel que :

$$Q \vDash \mathcal{A}/P \text{ si et seulement si } P \times Q \vDash \mathcal{A}.$$

On appelle quotient d'un automate simple par un processus l'opération qui à $\mathcal{A}$ et P associe l'automate $\mathcal{A}/P$. Le problème de contrôle centralisé est donc réduit au problème de la satisfiabilité d'un automate $\mathcal{A}/P \wedge \mathcal{B}$. De plus, les modèles de $\mathcal{A}/P \wedge \mathcal{B}$ sont exactement les solutions du problème **CCP** .

Théoreme 2.2.2 Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates simples. On peut construire un automate simple $\mathcal{A}/\mathcal{B}$ tel que :

$P \vDash \mathcal{A}/\mathcal{B}$ si et seulement si il existe un processus $Q \vDash \mathcal{B}$ tel que $P \times Q \vDash \mathcal{A}$.

On appelle quotient d'automates simples l'opération qui à $\mathcal{A}$ et $\mathcal{B}$ associe l'automate $\mathcal{A}/\mathcal{B}$. Comme le démontre les auteurs dans [AVW03], le problème **PCD** est réduit au problème de la satisfiabilité d'un automate simple résumé dans le corollaire suivant :

Corollaire 2.2.3 Le problème **PCD** a une solution si et seulement si l'automate $\left(\mathcal{A}/\mathcal{B}_n/\mathcal{B}_{n-1}/\cdots\mathcal{B}_2/P\right) \wedge \mathcal{B}_1$ a un modèle.

Preuve : En effet :

$Q_k \vDash \mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{k+1}/(P \times Q_1 \times \cdots \times Q_{k-1})$ et pour tout $i \leq k$, $Q_i \vDash \mathcal{B}_i$

si et seulement si

$$P \times Q_1 \times \cdots \times Q_k \vDash \mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{k+1} \text{ et pour tout } i \leq k,\ Q_i \vDash \mathcal{B}_i$$

si et seulement si il existe $Q_{k+1} \vDash \mathcal{B}_{k+1}$ tel que :

$P \times Q_1 \times \cdots \times Q_k \times Q_{k+1} \vDash \mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{k+2}$ et pour tout $i \leq k$, $Q_i \vDash \mathcal{B}_i$

si et seulement si il existe Q_{k+1} tel que :

$Q_{k+1} \vDash \mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{k+2}/(P \times Q_1 \times \cdots \times Q_k)$ et pour tout $i \leq k+1$, $Q_i \vDash \mathcal{B}_i$

On montre alors par induction que $Q_1 \vDash \left(\mathcal{A}/\mathcal{B}_n/\mathcal{B}_{n-1}/\cdots\mathcal{B}_2/P\right) \wedge \mathcal{B}_1$ si et seulement il existe $Q_2, Q_3, \ldots, Q_n$ tel que $Q_k \vDash \mathcal{B}_k$ et $P \times Q_1 \times \cdots \times Q_n \vDash \mathcal{A}$ □

2.3 Spécifications de contrôlabilité

Nous avons vu au chapitre 1 qu'un automate simple peut forcer l'existence de transition. C'est le principal point qui nous permet d'exprimer les contraintes de contrôlabilité avec des automates simples. Nous illustrons ceci avec les exemples suivants.

Une très grande partie des travaux sur le contrôle stipule que les évènements incontrôlables sont fixés comme le traduit la spécification suivante.

Exemple 2.3.1 Supposons que l'ensemble des évènements A est partitionné en l'ensemble des évènements contrôlables A_C et l'ensemble des évènements incontrôlables A_U. Soit $\mathcal{A}(A_U)$ un automate simple qui a seulement un état universel q de parité nulle et la fonction de transition :

$$\delta(q) = \{(a, q) : a \in A\} \cup \{\rightarrow_a : a \in A_U\}$$

Un contrôleur Q satisfait $\mathcal{A}(A_U)$ si les transitions étiquetées par un événement incontrôlable existent toujours. Donc Q ne peut pas interdire les évènements incontrôlables.

Nous pouvons considérer aussi le cas où le contrôle de certains évènements ne peut pas être simultané.

Exemple 2.3.2 Soit $a, b \in A$. Soit $\mathcal{A}_{a,b} = \langle A, Q, q_0, \{q_a, q_b\}, q_0, \delta, Q^\omega \rangle$ un automate simple avec :
- $\delta(q_0) = \{(\varepsilon, q_a), (\varepsilon, q_b)\}$
- $\delta(q_a) = \{(c, q_0) : c \in A\} \cup \{\rightarrow_a\}$
- $\delta(q_b) = \{(c, q_0) : c \in A\} \cup \{\rightarrow_b\}$

Alors un modèle de $\mathcal{A}_{a,b}$ a toujours soit une a-transition ou soit une b-transition. Donc le contrôleur ne peut pas interdire les évènements a et b en même temps. En d'autres termes, il peut bloquer au plus un événement parmi a et b.

Le cas où la contrôlabilité d'un événement dépend du comportement du SED est illustré dans l'exemple suivant.

Exemple 2.3.3 Soit $f, c \in A$ et $Q = \{q_f, q_0\}$.
Soit $\mathcal{A}_{f,c} = \langle A, Q, \emptyset, Q, q_0, \delta, Q^\omega \rangle$ un automate simple avec :
- $\delta(q_0) = \{(a, q_0) : a \neq f\} \cup \{(f, q_f)\}$
- $\delta(q_f) = \{(a, q_f) : a \in A\} \cup \{\rightarrow_c\}$

Un contrôleur est un modèle de $\mathcal{A}_{f,c}$ si il y a toujours une c-transition après l'occurrence de l'événement f. Ainsi, après une "erreur" f, c devient incontrolable.

Dans l'exemple suivant la contrôlabilité dépend de critères infinis sur les comportements.

Exemple 2.3.4 Soit $c \in A$. Soit $\mathcal{A}_c = \langle A, Q, q_0, \{q_c, q_{uc}\}, q_0, \delta, r \rangle$ un automate simple à parité avec :
- $\delta(q_0) = \{(\varepsilon, q_c), (\varepsilon, q_{uc})\}$
- $\delta(q_c) = \{(a, q_0) : a \in A\}$
- $\delta(q_{uc}) = \{(a, q_{uc}) : a \in A\} \cup \{\rightarrow_c\}$

– $r(q_0) = r(q_c) = 1$ et $r(q_{uc}) = 2$

Soit Q un contrôleur. Une partie infinie dans le jeu $G(\mathcal{A}_c, Q)$ est gagnante si et seulement si elle atteint l'état q_{uc}. De plus, quand une stratégie atteint l'état q_{uc}, le contrôleur doit toujours avoir une c-transition. En d'autres termes, l'évènement c doit devenir incontrôlable sur les comportements infinis du contrôleur.

2.4 Comportement admissible et comportement requis

Exemple 2.4.1 De nombreuses études de la théorie du contrôle supervisé ont pour but de réussir à restreindre les comportements du SED à des *comportements admissibles*. Supposons que ces comportements admissibles soient décrits par un processus $Ad = \langle A, S_a, s_0^a, e_a \rangle$. Soit $\mathcal{A}(Ad) = \langle A, S_a, \emptyset, S_a, s_0^a, \delta_a, r_a \rangle$ l'automate simple à parité avec, pour tout $s \in S_a$, on a $r_a(s) = 0$ et :

$$\delta_a(s) = \{(a, e_a(s,a)) : e_a(s,a) \text{ est défini}\} \cup \{\nrightarrow_a : e_a(s,a) \text{ est défini}\}$$

Intuitivement, la propriété $r_a(s) = 0$ (équivalente à $Acc = Q^\omega$) implique qu'il n'y a pas de contraintes sur les comportements infinis. La fonction de transition indique que les modèles de $\mathcal{A}(Ad)$ doivent avoir uniquement des transitions qui existent dans le processus Ad. Ainsi, le système supervisé SVS satisfait $\mathcal{A}(Ad)$ si et seulement si $SVS \leq Ad$ (voir Définition 0.4.2).

Exemple 2.4.2 On peut aussi vouloir que le SED garde certains comportements, i.e., le système supervisé réalise un *comportement requis*. Supposons que le comportement requis soit défini par le processus $Re = \langle A, S_r, s_0^r, e_r \rangle$. Soit $\mathcal{A}(Re) = \langle A, S_r, \emptyset, S_r, s_0^r, \delta_r, Q^\omega \rangle$ un automate simple avec, pour tout $s \in S_a$,

$$\delta_a(s) = \{(a, e_r(s,a)) : e_r(s,a) \text{ est défini}\} \cup \{\rightarrow_a : e_r(s,a) \text{ n'est pas défini}\}$$

La fonction de transition indique que les modèles de $\mathcal{A}(Re)$ doivent avoir toutes les transitions qui existent dans le processus Re. Donc, un système supervisé SVS est un modèle de $\mathcal{A}(Re)$ si et seulement si $Re \leq SVS$.

2.5 Comportement non bloquant

Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un processus et $E \subseteq \Lambda$ un sous-ensemble de propriétés d'états. Nous avons vu dans la partie que l'on peut associer à P un langage marqué $\mathcal{L}_E(P)$ (voir Remarque 0.3.3).

Exemple 2.5.1 Avec les automates simples, on peut spécifier que le langage marqué d'un système supervisé soit exactement un langage marqué donné (ici $\mathcal{L}_E(P)$). Soit $\mathcal{A}_P^E = \langle A, \Lambda, S \cup \{q_\perp\}, \emptyset, S \cup \{q_\perp\}, s_0, \delta, Q^\omega \rangle$ un automate simple avec, pour tout $s \in S$ et pour tout $X \subseteq \Lambda$, on a

- Si $\lambda(s) = E$ et $X \neq E$, alors :

$$\delta(s, X) = \{(\varepsilon, q_\perp)\} \text{ et } \delta(q_\perp, X) = \emptyset = \delta(q_\perp, E)$$

- Sinon

$$\delta(s, X) = \{\rightarrow_a, (a, e(s,a)) : e(s,a) \text{ est défini}\} \cup \{\nrightarrow_a : e(s,a) \text{ n'est pas défini}\}$$

Soit $Q = \langle A, S', s'_0, e' \rangle$ un contrôleur sans propriété locale et R un SED. On peut vérifier que $Q \times R \vDash \mathcal{A}_P^E$ si et seulement si $\mathcal{L}_E(Q \times R) = \mathcal{L}_E(P)$.

Exemple 2.5.2 Soit un contrôleur Q sans propriété locale. On peut aussi vouloir que le système supervisé $P \times Q$ soit nonbloquant par rapport à E, i.e., tous ses comportements finis peuvent atteindre un état ayant la propriété E (voir remarque 0.3.3). En d'autres termes :

$$\mathcal{L}(P \times Q) \subset Pref(\mathcal{L}_E(P \times Q))$$

Définition 2.5.3 Soit $\mathcal{A}_{NB}(P, E) = \langle A, Q, Q^\exists, Q^\forall, s_0, \delta, r \rangle$ un automate simple *alternant* avec :

- $Q^\exists = \{q_s : s \in S\}$
- $Q^\forall = S \cup \{q_s^a : e(s,a) \text{ est défini}\} \cup \{q_\top\}$
- $r(q) = 0$ pour tout $q \in Q^\forall$ et $r(q) = 1$ pour tout $q \in Q^\exists$
- Pour tout $s \in S$ et $a \in A$ on a :
  - $\delta(s) = \{(a, e(s,a)) : e(s,a) \text{ est défini}\} \cup \{(\epsilon, q_s)\}$
  - Si $s \in S_E$, alors $\delta(q_s) = \{(\epsilon, q_\top)\}$ et $\delta(q_\top) = \emptyset$
  - Si $s \notin S_E$, alors $\delta(q_s) = \{(\epsilon, q_s^a) : e(s,a) \text{ est défini}\}$
  - $\delta(q_s^a) = \{\rightarrow_a\} \cup \{(a, q_{e(s,a)})\}$

Proposition 2.5.4 Soit $Q = \langle A, S', s'_0, e' \rangle$ un processus sans propriétés locales. $P \times Q \vDash \mathcal{A}_{NB}(P, E)$ si et seulement si

$$\mathcal{L}(P \times Q) \subset Pref(\mathcal{L}_E(P \times Q))$$

Preuve : Soit $u \in A^*$ tel que $e \times e'((s_0, s'_0), u)$ est défini et égale à (s, s'). Notons que, dans le jeu $G(\mathcal{A}_{NB}(P, E), P \times Q)$, tous les chemins, correspondant à un mot u et qui peuvent être perdant, finissent dans la position $(q_s, (s, s'))$.

Supposons que $\mathcal{L}(P \times Q) \subset Pref(\mathcal{L}_E(P \times Q))$. Dans $G(\mathcal{A}_{NB}(P, E), P \times Q)$, on définit la stratégie σ comme suit. Soit $(q_s, (s, s'))$ une position accessible par σ. Si $s \in S_E$ alors $\sigma(q_s, (s, s')) = q_\top$. Si $s \notin S_E$, alors il

existe au moins un mot av de longueur minimal où $a \in A$, $v \in A^*$ tel que $e \times e'((s,s'),av)$ est défini et $e(s,av) \in S_E$. Dans ce cas, $\sigma(q_s,(s,s')) = q_s^a$. On peut vérifier que σ est une stratégie gagnante. Donc $P \times Q$ est un modèle de $\mathcal{A}_{NB}(P,E)$.

Supposons que $\mathcal{L}(P \times Q) \not\subset Pref(\mathcal{L}_E(P \times Q))$. Soit $u \notin Pref(\mathcal{L}_E(P \times Q))$ tel que $e \times e'((s_0,s_0'),u) = (s,s')$. Notons que tous les chemins infinis depuis $(q_s,(s,s'))$ sont perdants à cause de la parité des états. Soit $v \in A^*$ tel que $e \times e'((s_0,s_0'),uv) = (s_v,s_v')$ et $\{e \times e'((s_v,s_v'),a) : a \in A\} = \emptyset$. Depuis la position $(q_{s_v},(s_v,s_v'))$, il n'y a pas d'arête vers $(q_\top,(s,s'))$ car $s_v \notin S_E$. De plus, soit $\delta(q_{s_v}) = \emptyset$ ou soit, pour tout a tel que $e(s,a)$ est défini, il y a une arête depuis $(q_{s_v}^a,(s_v,s_v'))$ vers $\bot$ car $\rightarrow_a \in \delta(q_{s_v}^a)$. Ainsi, il n'y a pas de stratégie gagnante depuis $(q_s,(s,s'))$. Donc, $P \times Q$ n'est pas un modèle de $\mathcal{A}_{NB}(P,E)$. □

2.6 SED avec condition infinitaire

Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un SED. On peut supposer que les comportements infinis de P ne sont pas tous souhaitables. Plus formellement, supposons que soit défini une condition $Acc(P) \subset S^\omega$, d'une des formes introduites dans la partie 1.2, représentant les suites d'états infinis admissibles.

Définissons maintenant une spécification correspondant à cet objectif de contrôle. Soit $\mathcal{A}(P,Acc(P)) = \langle A, S, \emptyset, S, s_0, \delta, Acc(P) \rangle$ l'automate simple tel que pour tout $s \in S$, $\delta(s) = \{(a,e(s,a)) : e(s,a) \text{ est défini}\}$.

Soit P' un contrôleur. Dans le jeu $G(\mathcal{A}(P,Acc(P)), P \times P')$ tous les chemins finis sont gagnants puisque qu'il n'y a pas d'arête vers $\bot$ et que toutes les positions sont universelles. De plus un chemin infini est gagnant si et seulement si sa projection sur S est dans $Acc(P)$. Donc

$$P \times P' \vDash \mathcal{A}(P,Acc(P)) \text{ si et seulement si } \mathcal{L}^\omega(P \times P') \subset Acc(P,A)$$

où $Acc(P,A)$ est l'ensemble des comportements infinis $a_1a_2 \cdots a_n \cdots \in A^\omega$ tel que la suite associée d'états $s_0e(s_0,a_1)e(s_0,a_1a_2) \cdots e(s_0,a_1 \cdots a_n) \cdots$ appartient à l'ensemble $Acc(P)$.

2.7 Contrôle avec évènements forcés

On envisage dans cette partie le cas des événements forcés introduit dans [GR87] (voir aussi [H90] et [KS95]) où non seulement le contrôleur bloque certaines transitions du SED mais aussi force l'occurrence de certains évènements. Plus précisément, si le SED peut effectuer une transition par un

événement forcé f, alors le contrôleur doit n'autoriser qu'une unique transition par f, les autres évènements étant tous bloqués.

Dans ce contexte, le contrôleur doit donc savoir si le SED peut réaliser une transition forcée. Généralement, ceci n'est pas possible car le contrôleur ne peut préjuger des comportements futurs du SED. On peut tout de même répondre à ce type de problème si l'on permet au SED de communiquer avec le contrôleur des informations par les propriétés locales de ses états.

Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un SED. Supposons que l'ensemble Λ contienne un symbole spécial F spécifiant que l'événement f peut être forcé. Plus précisément, si un état $s \in S$ est étiqueté par F (c'est à dire $F \in \lambda(s)$), alors il y a depuis s une f-transition ($f \in D_P(s)$).

On suppose qu'un contrôleur $Q = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ ne peut être étiqueté par F, ce qui peut être spécifié par l'automate simple $\mathcal{A}_{NotF}$ ayant un état universel q et un état perdant existentiel $q_\perp$ tous deux de parité nulle et avec la fonction de transition δ définie comme suit. Pour $E \subseteq \Lambda$, $\delta(q_\perp, E) = \emptyset$ et

$$\delta(q, E) = \begin{cases} \{(\varepsilon, q_\perp)\} \text{ si } F \in E \\ \{(a, q) : a \in A\} \text{ sinon} \end{cases}$$

On définit enfin une spécification $FORCE(f)$ pour le système supervisé $P \times Q$ qui force la transition f comme suit. $FORCE(f)$ est un automate ayant un seul état q et une fonction de transition δ tel que, pour tout $E \subseteq \Lambda$, on a

$$\delta(q, E) = \begin{cases} \{\rightarrow_f, (f, q)\} \cup \{\nrightarrow_a : a \neq f\} \text{ si } F \in E \\ \{\nrightarrow_f\} \cup \{(a, q) : a \in A\} \text{ sinon} \end{cases}$$

Supposons que $Q \vDash \mathcal{A}_{NotF}$, c'est à dire, Q n'a pas d'étiquettes contenant le symbole F. $P \times Q \vDash FORCE(f)$ si et seulement si, pour tout état (s, s') de $P \times Q$, on a :

- Soit $F \notin \lambda \times \lambda'_{P\times Q}(s, s')$ et $f \notin D_{P\times Q}(s, s')$.
- Ou soit $F \in \lambda \times \lambda'_{P\times Q}(s, s')$ et $D_{P\times Q}(s, s') = \{f\}$.

Mais $F \in \lambda \times \lambda'_{P\times Q}(s, s') = \lambda_P(s) \cup \lambda'_Q(s')$ seulement si $F \in \lambda_P(s)$ puisque Q n'a pas d'étiquettes contenant le symbole F. C'est à dire, uniquement si P a une f-transition : $f \in D_P(s)$.

Donc $P \times Q \vDash FORCE(f)$ si et seulement si, depuis chaque état (s, s') du système supervisé, si P a une f-transition, alors $P \times Q$ a une unique f-transition : $D_{P\times Q}(s, s') = \{f\}$.

Donc l'ensemble des contrôleurs $Q \vDash \mathcal{A}_{NotF}$ tel que $P \times Q \vDash FORCE(f)$ est l'ensemble des contrôleurs qui ne sont pas étiquetés par F et qui forcent

les transitions f du SED P.

On peut bien sûr ajouter d'autres contraintes de contrôlabilité pour le contrôleur grâce à une spécification que celui-ci doit satisfaire. Par exemple, on peut spécifier qu'un contrôleur ne peut que forcer l'événement f, c'est à dire soit il force f ou soit il ne bloque aucun événements.

Soit l'automate simple $FORCABLE(f)$ avec un état initial existentiel q_0 et deux états universels q_f et $q_{\text{not}\,f}$, tous de parités nulles, tel que, pour la fonction de transition δ, on a :

- $\delta(q_0) = \{(\varepsilon, q_f), (\varepsilon, q_{\text{not}\,f})\}$
- $\delta(q_f) = \{\rightarrow_f, (f, q_0)\} \cup \{\not\rightarrow_a : a \neq f\}$
- $\delta(q_{\text{not}\,f}) = \{\rightarrow_a, (a, q_0) : a \in A\}$

Ainsi, si $Q \vDash \mathcal{A}_{NotF} \wedge FORCABLE(f)$ et $P \times Q \vDash FORCE(f)$, alors non seulement Q force l'événement f si il existe, mais aussi ne bloque aucun événement si la transition f n'existe pas.

2.8 Conclusion

Le contrôle centralisé et décentralisé avec observation totale est donc décidable si l'on définit les contraintes de contrôlabilité ainsi que les spécifications du système supervisé par des automates simples.

Les opérations de quotient d'un automate par un processus et de quotient de deux automates, introduit dans [AVW03], sont les principaux outils qui nous permettent de transformer les problèmes de contrôles en problèmes de satisfiabilités d'automates simples. La synthèse de contrôleurs est donc ici rendu possible par le calcul de stratégie gagnante dans des jeux présentés aux parties 1.3 et 1.4. Nous essayerons d'étendre ces types d'opérations dans des chapitres ultérieurs pour le cas du contrôle avec information partielle.

Le cadre défini dans [AVW03], et repris ici, est une extension importante de la problèmatique de la synthèse de contrôleurs. Il permet, comme dans la plupart des travaux, de fixer un ensemble d'événements incontrôlables mais aussi de définir la contrôlabilité de façon dynamique, en fonction du comportement passé du SED. Concernant le système supervisé, on peut exprimer des spécifications régulières classiques : comportement requis, admissible ou nonbloquant. Il est aussi possible de prendre en compte les comportements infinis du SED. Le problème de permissivité maximale, c'est à dire trouver, quand ils existent, les contrôleurs qui autorisent "le plus" de comportements du SED, est aussi un problème spécifiable avec des automates simples (voir [PR05] et [Rie03]).

De manière beaucoup moins formelle, on peut considérer que cette approche est très générique puisque elle ne différencie pas les contrôleurs des processus (les contrôleurs peuvent être eux mêmes contrôlés), et pas non plus les contraintes de contrôlabilité des spécifations du système supervisé.

Troisième partie

Contrôle centralisé avec information partielle

Chapitre 3

Information partielle sur les évènements

Nous allons dans cette partie étendre le problème de contrôle d'un SED au cas où le contrôleur n'a qu'une connaissance partielle des événements auxquels le SED réagit. Nous prouvons ici que le contrôle centralisé reste décidable dans ce cas.

3.1 Événements inobservables et indiscernables

La notion d'observation partielle introduite dans [LW88] et dans [CDFV88] est formalisée par la notion de *masque* $M : A \to \Delta \cup \{\varepsilon\}$ (voir aussi [CL99]) qui associe à un évènement, soit le mot vide ε, soit un symbole dans l'ensemble Δ. Ainsi, on considère que le contrôleur observe uniquement les évènements filtré par le masque. Si, pour $a \in A$, $M(a) = \varepsilon$ on parle alors d'un évènement *inobservable* et si pour $a, b \in A$, on a $M(a) = M(b)$, on dit que les évènements a et b sont *indiscernables*.

Un contrôleur n'observant que les évènements d'un SED filtré par le masque doit exercer le même contrôle sur le SED après des évènements indiscernables. De même, un contrôleur doit se comporter de la même façon avant et après un évènement inobservable.

Les contraintes d'observabilité peuvent se formaliser, comme dans [AVW03] de la façon suivante. Soit $Q = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un contrôleur. Pour tout $s \in S$, si l'évènement $a \in A$ est inobservable, alors $e(s, a) = s$. Les contraintes d'indiscernabilité peuvent se formuler de la même manière. Si les évènements $a, b \in A$ sont indiscernables, alors $e(s, a) = e(s, b)$.

Remarquons que même si des évènements sont indiscernables, le contrôleur peut détecter que l'un d'eux s'est produit ; ce qui n'est pas la cas pour les évènements inobservables. En particulier, un contrôleur peut se comporter différement avant et après des évènements indiscernables. Il peut, par exemple, bloquer un événement a uniquement s'il n'y a pas eu d'occurence d'un des événements indiscernables f_1 ou f_2.

On peut aussi avoir différentes classes d'évènements indiscernables alors qu'il y a au plus un ensemble d'évènements inobservables (voir Figure 3.1).

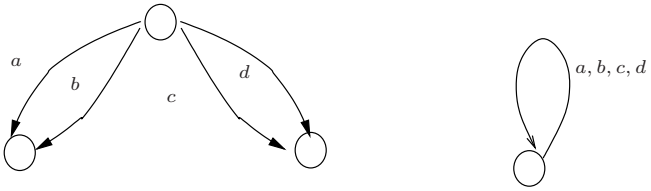


FIG. 3.1 –

Ces contraintes sur les contrôleurs ne sont pas expressibles par des automates simples car ceux-ci ne distinguent pas les processus bisimilaires (voir lemme 1.5.13).

Nous introduisons dans la partie 3.2 suivante, les automates étendus. C'est une extension des automates simples permettant de tester si une transition est une boucle ou si deux transitions atteignent le même état. Des exemples de spécifications pour les contrôleurs sont présentés dans la partie 3.3.

Nous verrons dans la partie 3.5 que la satisfiabilité de ces automates est décidable après avoir établi, dans une partie 3.4 très technique, un théorème de simulation pour les automates étendus.

Dans la partie 3.6, nous prouvons que le contrôle centralisé avec évènements inobservables et indiscernables est décidable.

Nous prouvons ensuite certaines propriétés de ces automates dans la partie 3.8 et les comparons avec les automates simples dans la partie 3.7.

3.2 Extension du μ-calcul : automate étendu

3.2.1 Syntaxe d'un automate étendu

Définition 3.2.1 Un *automate étendu* $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$ est un automate simple dont on modifie la fonction de transition δ. En effet,

celle-ci définit de nouveaux types de transitions comme suit :

$$\delta : Q \times 2^{\Lambda} \to \mathcal{P}\Big(\big((A \cup \{\varepsilon\}) \times Q\big) \cup \big(A \times \{\to, \twoheadrightarrow, \circlearrowleft, \not\circlearrowleft\}\big) \cup \big(A \times A \times \{\downdownarrows, \not\downdownarrows\}\big)\Big)$$

Notation 3.2.2 Pour simplifier les écritures, nous noterons très souvent $\circlearrowleft_a$, $\not\circlearrowleft_a$, $\downdownarrows_{a,b}$ et $\not\downdownarrows_{a,b}$ au lieu, respectivement, de $(a, \circlearrowleft)$, $(a, \not\circlearrowleft)$, $(a, b, \downdownarrows)$ et $(a, b, \not\downdownarrows)$.

De façon informelle, $\circlearrowleft_a$ spécifie que si un processus, depuis un état, a une a-transition alors ce doit être une boucle. La partie suivante formalise cette idée ainsi que la sémantique pour $\not\circlearrowleft_a$, $\downdownarrows_{a,b}$ et $\not\downdownarrows_{a,b}$.

3.2.2 Sémantique d'un automate étendu

La notion de calcul et d'acceptation d'un automate étendu est aussi décrite par un jeu comme suit.

Définition 3.2.3 Soit $\mathcal{A} = \langle A, \lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ un automate étendu et $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus. Le *jeu d'acceptation* $G(\mathcal{A}, P)$ est un graphe $\langle V_0, V_1, v_0, Edge, Acc_G\rangle$ définit comme pour un automate simple, auquel on ajoute de nouvelles arêtes :

- Depuis chaque sommet (q, s) et pour tout $\circlearrowleft_a \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, a)$ est défini mais différent de s, et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\not\circlearrowleft_a \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, a)$ est défini et égale à s, et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\downdownarrows_{a,b} \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, a)$ et $e(s, b)$ sont définis mais pas égaux, et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\not\downdownarrows_{a,b} \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, a)$ et $e(s, b)$ sont définis et égaux, et vers $\top$ sinon.

Notation 3.2.4 On note $AUTO(A, \Lambda, \circlearrowleft, \not\circlearrowleft, \downdownarrows, \not\downdownarrows)$ l'ensemble des automates étendus sur l'alphabet A avec les propriétés locales Λ. On appelle automate avec évènements inobservables et indiscernables ces automates.

On note $AUTO(A, \Lambda, \circlearrowleft, \not\circlearrowleft)$ l'ensemble des automates étendus sur l'alphabet A avec les propriétés locales Λ n'ayant aucune transition vers $A \times A \times \{\downdownarrows, \not\downdownarrows\}$. On appelle automate avec évènements inobservables ces automates.

On note $AUTO(A, \Lambda, \downdownarrows, \not\downarrow)$ l'ensemble des automates étendus sur l'alphabet A avec les propriétés locales Λ n'ayant aucune transition vers $A \times \{\circlearrowleft, \varnothing\}$. On appelle automate avec évènements indiscernables ces automates.

Les ensembles $AUTO(A, \Lambda, \circlearrowleft)$, $AUTO(A, \Lambda, \not\circlearrowleft)$, $AUTO(A, \Lambda, \downdownarrows)$, $AUTO(A, \Lambda, \not\downarrow)$, $AUTO(A, \Lambda, \varnothing, \not\downarrow)$, etc... sont définis de façon similaire.

3.2.3 Automate étendu nondéterministe

Définition 3.2.5 Un automate étendu $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ est *biparti* si, pour tout $q \in Q$ et $E \subseteq \Lambda$, on a :
- Si $q \in Q^\exists$ alors $\delta(q, E) \subset \{\varepsilon\} \times Q$.
- Si $q \in Q^\forall$ alors $\delta(q, E) \cap \{\varepsilon\} \times Q = \emptyset$.

Définition 3.2.6 Un automate étendu $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ est *nondéterministe* si il est biparti et pour tout $a, b \in A$, $q, q_1, q_2 \in Q$ et $E \subseteq \Lambda$, on a :
- Si $\circlearrowleft_a \in \delta(q, E)$, alors $\delta(q, E) \cap (\{a\} \times Q) = \emptyset$.
- Si $(a, q_1), (a, q_2) \in \delta(q, E)$, alors $q_1 = q_2$.
- Si $(a, q_1), (b, q_2), \downdownarrows_{a,b} \in \delta(q, E)$, alors $q_1 = q_2$.

Nous introduisons maintenant une réécriture des automates étendus qui nous sera utile dans la section suivante.

Définition 3.2.7 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate étendu nondéterministe. On dit que $\mathcal{A}$ est *complet* si pour tout $q \in Q^\forall$ et $E \subset \Lambda$, si $\twoheadrightarrow_a \notin \delta(q, E)$ et $\circlearrowleft_a \notin \delta(q, E)$ alors il existe $q' \in Q^\exists$ tel que $(a, q') \in \delta(q, E)$.

Proposition 3.2.8 Tout automate étendu nondéterministe est équivalent à un automate étendu nondéterministe complet.

Preuve : Définissons d'abord $complet(\mathcal{A})$ l'automate suivant $\langle A, \Lambda, Q, Q^\exists \cup \{{q_\top}^\exists\}, Q^\forall \cup \{{q_\top}^\forall\}, q_0, \delta, Acc'\rangle$ avec $Acc' = Acc \cup Q^*({q_\top}^\exists {q_\top}^\forall)^\omega$. C'est l'automate complet identique à $\mathcal{A}$ auquel on ajoute, pour tout q et $E \subset \Lambda$, $(a, {q_\top}^\exists) \in \delta(q, E)$ si $\twoheadrightarrow_a \notin \delta(q, E)$, $\circlearrowleft_a \notin \delta(q, E)$ et $(A \times Q) \cap \delta(q, E) = \emptyset$. De plus, pour tout $E \subset \Lambda$, $\delta({q_\top}^\exists, E) = \{(\varepsilon, {q_\top}^\forall)\}$ et $\delta({q_\top}^\forall, E) = \{(a, {q_\top}^\exists) : a \in A\}$ On peut vérifier que $\mathcal{A}$ et $complet(\mathcal{A})$ ont les mêmes modèles. □

3.3 Spécifications d'information partielle

Nous présentons ici quelque exemples de contraintes d'observabilité et d'indiscernabilité.

De nombreuses études considèrent que les évènements inobservables sont fixés comme le traduit la spécification suivante.

Exemple 3.3.1 Supposons que l'ensemble des évènements A est partitionné en l'ensemble des évènements observables A_O et l'ensemble des évènements inobservables A_{UO}. Soit $\mathcal{A}(A_O)$ un automate avec évènements inobservables ayant un seul état universel q de parité nulle et la fonction de transition :

$$\delta(q) = \{(a, q) : a \in A\} \cup \{\circlearrowleft_a : a \in A_O\}$$

Un contrôleur Q satisfait $\mathcal{A}(A_O)$ si les transitions étiquetées par un événement inobservable sont des boucles. Donc les comportements de Q sont identiques qu'un événement inobservable se produisent ou non.

On peut aussi considérer le cas où les évènements indiscernables sont fixés.

Exemple 3.3.2 Supposons que l'ensemble des évènements A est partitionné en classes d'événements indiscernables noté $\Pi = \{A_0, A_1, \ldots, A_n\}$. Soit $\mathcal{A}(\Pi)$ un automate avec évènements indiscernables ayant un seul état universel q de parité nulle et la fonction de transition :

$$\delta(q) = \{(a, q) : a \in A\} \cup \{\downdownarrows_{a,b} : \exists i. a, b \in A_i\}$$

Un contrôleur Q satisfait $\mathcal{A}(\Pi)$ si les transitions appartenant au même élément de Π atteignent le même état. Donc les comportements de Q sont identiques après deux évènements indiscernables.

On peut aussi spécifier un lien entre la contrôlabilité et l'observabilité, comme le fait que les évènements inobservables sont incontrôlable.

Exemple 3.3.3 Soit $\mathcal{A}_{uo \Rightarrow uc} = \langle A, Q, q_0, 2^A, q_0, \delta, Q^\omega \rangle$ l'automate simple avec :
- $\delta(q_0) = \{(\varepsilon, B), B \subseteq A\}$
- $\delta(B) = \{(a, q_0) : a \in A\} \cup \{\circlearrowleft_a, \rightarrow_a : a \in B\} \cup \{\not\circlearrowleft_a : a \notin B\}$

Un modèle de cet automate doit avoir une a-transition quand il a une a-boucle. Donc il ne peut pas interdire les évènements inobservables.

Le cas où l'observabilité d'un événement dépend du comportement du SED est illustré dans l'exemple suivant. On peut adapter facilement cet exemple pour exprimer des contraintes dynamiques (dépendant du SED) pour l'indiscernabilité.

Exemple 3.3.4 Soit $f, b \in A$ et $Q = \{q_f, q_0\}$.
Soit $\mathcal{A}_{f,b} = \langle A, Q, \emptyset, Q, q_0, \delta, Q^\omega \rangle$ un automate simple avec :

- $\delta(q_0) = \{(a, q_0) : a \neq f\} \cup \{(f, q_f)\}$
- $\delta(q_f) = \{(a, q_f) : a \in A\} \cup \{\circlearrowleft_b\}$

Un contrôleur est un modèle de $\mathcal{A}_{f,b}$ si toute b-transition après l'occurrence de l'événement f est une boucle. Ainsi, après une "erreur" f, b devient inobservable.

3.4 Théorème de simulation pour les automates étendus

Dans cette partie, nous montrons qu'il existe un théorème de simulation pour les automates étendus comme l'affirme le théorème 3.4.1.

Théoreme 3.4.1 Tout automate étendu est équivalent à un automate étendu nondéterministe.

3.4.1 Lien entre automates étendus et automates simples

Définition 3.4.2 Soit A un ensemble. On note Π_A l'ensemble des partitions de la forme $\{N, A_0, A_1, \ldots, A_p\}$ où uniquement N et A_0 peuvent être vides.

Soit n un entier. On note $[n]$ l'ensemble $\{0, 1, \ldots n\}$ et $]n]$ l'ensemble $\{1, \ldots n\}$.

Définition 3.4.3 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus sur Λ. On note, pour tout $s \in S$, $\Pi_P(s)$ la partition $\{N_P(s), L_P(s), A_1, \ldots, A_p\} \in \Pi_A$ telle que :

- $a \in N_P(s)$ si et seulement si $e(s, a)$ n'est pas défini.
- $a \in L_P(s)$ si et seulement si $e(s, a) = s$.
- Pour tout $i \in]p]$ et $a, b \in A_i$, on a $e(s, a) = e(s, b)$.
- Pour tout $i, j \in]p]$ tel que $i \neq j$, $a \in A_i$ et $b \in A_j$, on a $e(s, a) \neq e(s, b)$.

On note $\widehat{P}$ le processus $\langle A, \Lambda, S, s_0, e, \widehat{\lambda}\rangle$ tel que, pour tout $s \in S$, $\widehat{\lambda}(s) = (\lambda(s), \Pi_P(s))$.

Définition 3.4.4 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ un automate étendu. Soit $q \in Q$, $E \subset \Lambda$ et $\pi = \{N, A_0, A_1, A_2, \ldots, A_p\} \in \Pi_A$.

Si $q \in Q^{\forall}$, on note $C_{\mathcal{A}}(q, \pi, E)$ la condition $C_1 \wedge C_2 \wedge C_3 \wedge C_4$ où :

- $C_1 \equiv \{a : \circlearrowleft_a \in \delta(q, E)\} \subseteq (N \cup A_0)$
- $C_2 \equiv \{a : \not\oslash_a \in \delta(q, E)\} \cap A_0 = \emptyset$
- $C_3 \equiv$ pour tout a, b tel que $\downdownarrows_{a,b} \in \delta(q, E)$, il existe $i \in [p]$ tel que $\{a, b\} \subseteq N \cup A_i$

- $C_4 \equiv$ pour tout a, b tel que $\not\downarrow_{a,b} \in \delta(q, E)$, et pour tout $i \in [p]$, on a $\{a, b\} \not\subset A_i$

Si $q \in Q^{\exists}$, on note $D_{\mathcal{A}}(q, \pi, E)$ la condition $D_1 \vee D_2 \vee D_3 \vee D_4$ où :

- $D_1 \equiv \{a : \circlearrowleft_a \in \delta(q, E)\} \cap (N \cup A_0) \neq \emptyset$.
- $D_2 \equiv \{a : \varnothing_a \in \delta(q, E)\} \not\subset A_0$.
- $D_3 \equiv$ il existe $i \in [p]$ et $a, b \in A$ tel que $\downdownarrows_{a,b} \in \delta(q, E)$ et $\{a, b\} \subseteq N \cup A_i$.
- $D_4 \equiv$ il existe a, b tel que $\not\downarrow_{a,b} \in \delta(q, E)$ et pour tout $i \in [p]$, $\{a, b\} \not\subset A_i$.

Définition 3.4.5 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$ un automate étendu. On note $\widehat{\mathcal{A}}$ l'automate *simple* $\langle A, \Lambda \times \Pi_A, Q', Q^{\exists} \cup \{q_\top\}, Q^{\forall} \cup \{q_\bot\}, q_0, \widehat{\delta}, Acc \rangle$ où $Q' = Q^{\exists} \cup \{q_\top\} \cup Q^{\forall} \cup \{q_\bot\}$ et avec, pour tout $E \in 2^{\Lambda}$, $\pi \in \Pi_A$ et $q \in Q$, on a :

$$\widehat{\delta}(q_\bot, (E, \pi)) = \widehat{\delta}(q_\top, (E, \pi)) = \emptyset$$

et

$$\widehat{\delta}(q, (E, \pi)) = \begin{cases} \{(\varepsilon, q_\top)\} \text{ si } q \in Q^{\exists} \text{ et } D_{\mathcal{A}}(q, \pi, E) \text{ vrai} \\ \{(\varepsilon, q_\bot)\} \text{ si } q \in Q^{\forall} \text{ et } C_{\mathcal{A}}(q, \pi, E) \text{ faux} \\ \delta(q, E) - (A \times \{\circlearrowleft, \varnothing\} \cup A \times A \times \{\downdownarrows, \not\downarrow\}) \text{ sinon} \end{cases}$$

Théoreme 3.4.6 Soit $\mathcal{A}$ un automate étendu sur Λ et P un processus. $P \vDash \mathcal{A}$ si et seulement si $\widehat{P} \vDash \widehat{\mathcal{A}}$.

Preuve : Soit (q, s) une position des jeux $G(\mathcal{A}, P)$ et $G(\widehat{\mathcal{A}}, \widehat{P})$. Si $q \in Q^{\exists}$, on peut vérifier qu'il existe une arête vers $\top$ dans $G(\mathcal{A}, P)$ à cause des transitions dans $A \times \{\circlearrowleft, \varnothing\} \cup A \times A \times \{\downdownarrows, \not\downarrow\}$ si et seulement si $D_{\mathcal{A}}(q, \Pi_P(s), \lambda(s))$ est vrai. De plus, si $q \in Q^{\forall}$, on peut vérifier qu'il existe une arête vers $\bot$ dans $G(\mathcal{A}, P)$ à cause des transitions dans $A \times \{\circlearrowleft, \varnothing\} \cup A \times A \times \{\downdownarrows, \not\downarrow\}$ si et seulement si $C_{\mathcal{A}}(q, \Pi_P(s), \lambda(s))$ est faux. □

3.4.2 Automate simple réduit

Notation 3.4.7 On suppose qu'il existe un ordre totale, noté $\leq_A$, sur l'alphabet A. Pour tout $\pi = \{N, A_0, A_1, \ldots A_p\} \in \Pi_A$, on note $Min_{\leq_A}(\pi)$ l'ensemble

$$\bigcup_{i=1}^{p} min_{\leq_A}\{a \in A_i\}$$

et $Min^C_{\leq_A}(\pi)$ son complémentaire dans A.

Définition 3.4.8 Soit $\mathcal{A} = \langle A, \Lambda \times \Pi_A, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc \rangle$ un automate simple. On dit que $\mathcal{A}$ est réduit si pour tout $q \in Q$, $E \subseteq \Lambda$ et $\pi \in \Pi_A$ on a :

$$\delta(q, (E, \pi)) \cap Min^C_{\leq_A}(\pi) \times (Q \cup \{\rightarrow\}) = \emptyset$$

Définition 3.4.9 Soit $P = \langle A, \Lambda \times \Pi_A, S, s_0, e, \lambda \rangle$ un processus sur $\Lambda \times \Pi_A$. On appelle réduction du processus P, noté $R(P)$ le processus $\langle A, \Lambda \times \Pi_A, S, s_0, R(e), \lambda \rangle$ tel que, pour tout $s \in S$ avec $\lambda(s) = (E, \pi)$ on a :

$$D_{R(P)}(s) = Min_{\leq_A}(\pi)$$

et pour tout $a \in D_{R(P)}(s)$, on a $R(e)(s, a) = e(s, a)$

Proposition 3.4.10 Soit $\mathcal{A}$ un automate simple réduit et P un processus. $P \vDash \mathcal{A}$ si et seulement si $R(P) \vDash \mathcal{A}$.

Preuve : On peut vérifier que les jeux $G(\mathcal{A}, P)$ et $G(\mathcal{A}, R(P))$ sont identiques. □

Proposition 3.4.11 Soit $\mathcal{A}$ un automate simple réduit. Alors il existe un automate simple nondéterministe réduit équivalent à $\mathcal{A}$.

Preuve : Soit $\mathcal{B} = \langle A, \Lambda \times \Pi_A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ un automate nondéterministe équivalent à $\mathcal{A}$. Soit $\mathcal{B}'$ l'automate nondéterministe réduit $\langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta', Acc \rangle$ avec, pour tout $q \in Q^\forall$, $\delta'(q, \pi) = \delta(q, \pi) - Min^C_{\leq_A}(\pi) \times (Q \cup \{\rightarrow\})$. On peut vérifier que, pour tout processus $P = \langle A, \Lambda \times \Pi_A, S, s_0, e, \lambda \rangle$, les jeux $G(\mathcal{B}, R(P))$ et $G(\mathcal{B}', R(P))$ sont identiques puisque, pour tout $s \in S$ tel que $\lambda(s) = (E, \pi)$, $D_{R(P)}(s) \cap Min^C_{\leq_A}(\pi) = \emptyset$. Ainsi, avec la proposition 3.4.10, on a $P \vDash \mathcal{A}$ si et seulement si $R(P) \vDash \mathcal{A}$ si et seulement si $R(P) \vDash \mathcal{B}$ si et seulement si $R(P) \vDash \mathcal{B}'$ si et seulement si $P \vDash \mathcal{B}'$. □

3.4.3 Réduction d'automates simples

Définition 3.4.12 Soit $\mathcal{A} = \langle A, \Lambda \times \Pi_A, Q, Q^\exists, Q^\forall \cup \{q_\top\}, q_0, \delta, Acc \rangle$ un automate simple sur $\Lambda \times \Pi_A$. On appelle réduction de l'automate $\mathcal{A}$, noté $R(\mathcal{A})$ l'automate réduit $\langle A, \Lambda \times \Pi_A, Q, Q^\exists, Q^\forall, q_0, R(\delta), Acc \rangle$ tel que pour tout $q \in Q$, $E \subset \Lambda$ et $\pi = \{N, A_0, A_1, \ldots A_p\} \in \Pi_A$, on a :

- Si $q \in Q^\forall$, alors $R(\delta)(q, (E, \pi))$ contient :
  - $\delta(q, (E, \pi)) - (Min^C_{\leq_A}(\pi) \times (Q \cup \{\rightarrow\}))$
  - (ε, q') si il existe $a \in A_0$ tel que $(a, q') \in \delta(q, (E, \pi))$
  - (a, q') si il existe $i \in]p]$ tel que $a \in A_i \cap Min_{\leq_A}(\pi)$ et il existe $b \in A_i$ tel que $(b, q') \in \delta(q, (E, \pi))$.
- Si $q \in Q^\exists$, alors $R(\delta)(q, (E, \pi))$ contient :
  - $\delta(q, (E, \pi)) - (Min^C_{\leq_A}(\pi) \times (Q \cup \{\rightarrow\}))$
  - (ε, q') si il existe $a \in A_0$ tel que $(a, q') \in \delta(q, (E, \pi))$

- (a, q') si il existe $i \in]p]$ tel que $a \in A_i \cap Min_{\leq_A}(\pi)$ et il existe $b \in A_i$ tel que $(b, q') \in \delta(q, (E, \pi))$.
- $(\varepsilon, q_\top)$ si $\delta(q, (E, \pi)) \cap (Min^C_{\leq_A}(\pi) - N) \times \{\rightarrow\} \neq \emptyset$

– $\delta(q_\top, (E, \pi)) = \emptyset$

Théoreme 3.4.13 Soit $\mathcal{A}$ un automate simple sur $\Lambda \times \Pi_A$ et P un processus. $\widehat{P} \vDash \mathcal{A}$ si et seulement si $\widehat{P} \vDash R(\mathcal{A})$.

Preuve : On peut vérifier que les jeux $G(\mathcal{A}, \widehat{P})$ et $G(R(\mathcal{A}), \widehat{P})$ sont identiques à l'exception de certaines arêtes depuis un état existentielle vers $\top$ dans le jeu $G(\mathcal{A}, \widehat{P})$, remplacées par des arêtes vers la position gagnante $q_\top$ dans le jeu $G(R(\mathcal{A}), \widehat{P})$. □

3.4.4 Lien entre automate simple nondéterministe réduit et automate étendu nondéterministe

Définition 3.4.14 Soit $\mathcal{A} = \langle A, \Lambda \times \Pi_A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate *simple nondéterministe réduit* sur $\Lambda \times \Pi_A$. On note $\mathcal{A}^\vee$ l'automate *étendu* $\langle A, \Lambda, Q, Q^\exists, Q^\forall \times \Pi_A, q_0, \delta^\vee, Acc^\vee\rangle$ sur Λ, avec, pour tout $E \in 2^\Lambda$, $\pi \in \Pi_A$ et $q \in Q$, on a :

- Si $q \in Q^\exists$, alors $\delta^\vee(q, E) = \{(\varepsilon, (q', \pi)) : (\varepsilon, q') \in \delta(q, (E, \pi))\}$
- Si $q \in Q^\forall$ et $\pi = \{N, A_0, A_1, \ldots A_p\}$, alors $\delta^\vee((q, \pi), E)$ contient :
  - $\delta(q, (E, \pi))$
  - $\{\circlearrowleft_a : a \in A_0\} \cup \{\not\circlearrowleft_a : a \notin A_0\}$
  - $\downdownarrows_{a,b}$ si il existe $i \in [p]$ tel que $a, b \in A_i$
  - $\not\downdownarrows_{a,b}$ si $a \in A_i$, $b \in A_j$ tel que $i \neq j$
- $Acc^\vee$ est l'ensemble des suites infinies dans $Q^\exists \cup (Q^\forall \times \Pi_A)$ dont les projections sur $Q^\exists \cup Q^\forall$ sont dans Acc.

Lemme 3.4.15 $\mathcal{A}^\vee$ est un automate étendu nondéterministe.

Théoreme 3.4.16 Soit $\mathcal{A}$ un automate simple nondéterministe réduit sur $\Lambda \times \Pi_A$ et P un processus. $P \vDash \mathcal{A}$ si et seulement si $P \vDash \widehat{\mathcal{A}^\vee}$.

Preuve : On peut vérifier que pour tout $q \in Q^\forall$, $\pi, \pi' \in \Pi_A$ et $E \subset \Lambda$, on a :

$$\widehat{\delta^\vee}((q, \pi), (E, \pi')) = \begin{cases} \{(\varepsilon, q_\perp)\} \text{ si } \pi \neq \pi' \\ \delta(q, (E, \pi)) \text{ si } \pi = \pi' \end{cases}$$

Donc, la position (q, s) avec $q \in Q^\forall$ et $\lambda(s) = (E, \pi)$ du jeu $G(\mathcal{A}, P)$ a les mêmes arêtes sortantes que la position $((q, \pi), s)$ du jeu $G(\widehat{\mathcal{A}^\vee}, P)$.

De plus, si $q \in Q^{\exists}$, $\pi \in \Pi_A$ et $E \subset \Lambda$, on a $D_{\mathcal{A}^\vee}(q, \pi, E)$ est faux car $\delta^\vee(q,(E,\pi)) \cap (A \times \{\circlearrowleft, \not\oslash\}) \neq \emptyset$ et $\delta^\vee(q,(E,\pi)) \cap (A \times A \times \{\downdownarrows, \not\rightrightarrows\}) = \emptyset$. Donc

$$\widehat{\delta^\vee}(q,(E,\pi)) = \delta^\vee(q,E) = \{(\varepsilon,(q',\pi')) : (\varepsilon,q') \in \delta(q,(E,\pi'))\}$$

Soit $P = \langle A, \Lambda \times \Pi_A, S, s_0, e, \lambda\rangle$ un processus sur $\Lambda \times \Pi_A$ et une position (q,s) du jeu $G(\widehat{\mathcal{A}^\vee}, P)$ avec $q \in Q^{\exists}$ et $\lambda(s) = (E,\pi)$. D'après $\widehat{\delta^\vee}(q,(E,\pi))$, il y a pour tout $\pi' \in \Pi_A$, une arête depuis (q,s) vers $((q,\pi'),s)$. Or $((q,\pi'),s)$ est une position perdante si $\pi \neq \pi'$. On peut donc restreindre les transitions de $\widehat{\delta^\vee}(q,(E,\pi))$ comme suit :

$$\widehat{\delta^\vee}(q,(E,\pi)) = \{(\varepsilon,(q',\pi)) : (\varepsilon,q') \in \delta(q,(E,\pi))\}$$

Donc, chaque arête depuis la position (q,s) avec $q \in Q^{\exists}$ et $\lambda(s) = (E,\pi)$ vers la position (q',s) du jeu $G(\mathcal{A},P)$ correspond à une arête depuis la position $((q,\pi),s)$ vers la position $((q',\pi),s)$ du jeu $G(\widehat{\mathcal{A}^\vee}, P)$. □

3.4.5 Preuve de théorème de simulation 3.4.1

Soit $\mathcal{B}$ un automate nondéterministe réduit équivalent à $R(\widehat{\mathcal{A}})$. On a $P \vDash \mathcal{A}$ si et seulement si $\widehat{P} \vDash \widehat{\mathcal{A}}$ si et seulement si $\widehat{P} \vDash R(\widehat{\mathcal{A}})$ si et seulement si $\widehat{P} \vDash \mathcal{B}$ si et seulement si $\widehat{P} \vDash \widehat{\mathcal{B}^\vee}$ si et seulement si $P \vDash \mathcal{B}^\vee$. □

3.5 Satisfiabilité des automates étendus

Définition 3.5.1 Soit $\mathcal{A} = \langle A, \lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ un automate étendu nondéterministe complet (voir Section 3.2.3).
Soit $G(\mathcal{A})$ le jeu $\langle V_0, V_1, v_0, Edge, Acc_G\rangle$ avec :

- $V_0 = Q^{\exists}$ et $V_1 = Q^{\forall} \times \Lambda \times \Pi_A$ (voir Notation 3.4.2)
- Pour tout $q, q' \in Q$, $E \subset \Lambda$ et $\pi = \{N, A_0, A_1, A_2, \ldots, A_p\} \in \Pi_A$, on a une arête $(q,(q',E,\pi)) \in Edge \cap V_0 \times V_1$ si $(\varepsilon,q') \in \delta(q,E)$ et pour tout $a \in A$, on a :
  - Si $\rightarrow_a \in \delta(q,E)$ alors $a \notin N$.
  - Si $\twoheadrightarrow_a \in \delta(q,E)$ alors $a \in N$.
  - Si $\circlearrowleft_a \in \delta(q,E)$ alors $a \in A_0 \cup N$.
  - Si $\not\oslash_a \in \delta(q,E)$ alors $a \notin A_0$.
  - Si $\downdownarrows_{a,b} \in \delta(q,E)$ alors il existe $i \in [p]$ tel que $\{a,b\} \subset A_i \cup N$.
  - Si $\not\rightrightarrows_{a,b} \in \delta(q,E)$ alors pour tout $i \in [p]$, $\{a,b\} \not\subset A_i$.
- Pour tout $q, q' \in Q$, $E \subset \Lambda$ et $\pi = \{N, A_0, A_1, A_2, \ldots, A_p\} \in \Pi_A$, on a une arête $((q,E,\pi),q') \in Edge \cap V_1 \times V_0$ si et seulement si il existe $a \notin N$ tel que $(a,q') \in \delta(q,E)$.

– Acc_G est l'ensemble des éléments de $(V_0 \cup V_1)^\omega$ dont la projection sur Q est dans Acc.

Théoreme 3.5.2 A chaque stratégie gagnante à mémoire finie dans $G(\mathcal{A})$ on peut faire correspondre un modèle de $\mathcal{A}$.

De plus, tous les modèles de $\mathcal{A}$, à bisimulation près, peuvent être obtenu de cette manière.

Corollaire 3.5.3 Soit $\mathcal{A}$ un automate étendu nondéterministe complet. $\mathcal{A}$ a un modèle si et seulement si il existe une stratégie gagnante dans $G(\mathcal{A})$.

Preuve du théorème 3.5.2 : On suppose de plus que pour tout $a, b \in A$ et $q, q_1, q_2 \in Q$, on a :

$$\text{si } (a, q_1), (b, q_2) \in \delta(q) \text{ et } \downdownarrows_{a,b} \notin \delta(q) \text{ alors } q_1 \neq q_2 \tag{3.5.1}$$

Il suffit pour cela de dupliquer certains états de $\mathcal{A}$

Soit $P = \langle A, S, s_0, e\rangle$ un processus. Supposons que $P \vDash \mathcal{A}$. Soit $\sigma : Q^\exists \times S \times H \to Q^\forall \times S$ avec la fonction d'histoire $hist : H \times Q \times S \to H$, une stratégie gagnante dans le jeu $G(\mathcal{A}, P)$. Définissons maintenant une stratégie gagnante dans $G(\mathcal{A})$ $\sigma' : Q^\exists \times H' \to Q^\forall \times \Lambda \times \Pi_A$ avec la fonction d'histoire $hist' : H' \times (V_0 \cup V_1) \to H'$ où $H' = Q \times S \times H$. Soit $(q, s, h) \in Q^\exists \times S \times H$ tel que $\sigma(q, s, h)$ est défini et égale à (q_1, s). Alors $\sigma'(q, (q, s, h))$ est défini et égale à $(q_1, \lambda(s), \Pi_P(s))$ (voir définition 3.4.3) et $hist'((q, s, h), (q_1, \lambda(s), \Pi_P(s))) = (q_1, s, h_1)$ où $h_1 = hist(h, (q, s))$. De plus si il y a une arête depuis $(q_1, \lambda(s), \Pi_P(s))$ vers q_2 dans $G(\mathcal{A})$, alors il existe $a \notin N_P(s)$ tel que $(a, q_2) \in \delta(q_1, \lambda(s))$. alors $hist'((q_1, s, h_1), q_2) = (q_2, s_2, h_2)$ avec $s_2 = e(s, a)$ et $h_2 = hist(h_1, (q_2, s_2))$ ($hist'$ est bien définit grâce à la propriété 3.5.1).

Donc l'arête depuis $(q_1, \lambda(s), \Pi_P(s))$ avec l'histoire (q_1, s, h_1) vers q_2 avec l'histoire (q_2, s_2, h_2) dans $G(\mathcal{A})$ correspond à l'arête depuis (q_1, s) avec l'histoire h_1 vers (q_2, s_2) avec l'histoire h_2 dans le jeu $G(\mathcal{A}, P)$.

Réciproquement, supposons qu'il existe une stratégie gagnante $\sigma' : Q^\exists \times H' \to Q^\forall \times \Lambda \times \Pi_A$ avec la fonction d'histoire $hist' : H' \times (V_0 \cup V_1) \to H'$ et l'histoire initiale h'_0 dans le jeu $G(\mathcal{A})$. Soit $P = \langle A, S, s_0, e, \lambda\rangle$ le processus défini comme suit :

– $s_0 = (q_0, h'_0)$
– $S = \{(q, h') \in Q^\exists \times H' : \sigma'(q, h') \text{ est défini}\}$
– $\lambda(q, h') = E$ si $\sigma'(q, h') = (q_1, E, \pi)$

- Soit $(q, h') \in Q^{\exists} \times H'$ avec $\sigma'(q, h') = (q_1, E, \pi)$ et $\pi = \{N, L, A_1, A_2, \ldots, A_p\}$.
  - $D_P(q, h') = \{a \in A : a \notin N\}$
  - $e((q, h'), a) = (q, h')$ si $a \in L$
  - $e((q, h'), a) = (q_2, h_2)$ si il existe i tel que $a \in A_i$, $(a, q_2) \in \delta(q_1, \lambda(q, h'))$ et $h_2 = hist'(h_1, (q_1, E, \pi))$.

(Le fait que $\mathcal{A}$ soit complet nous assure que les transitions de P sont bien définies.) Soit $\sigma : Q^{\exists} \times (Q^{\exists} \times H') \to Q^{\forall}$ une stratégie dans le jeu $G(\mathcal{A}, P)$ telle que $\sigma(q, (q, h'))$ est définit et égale à q_1 si $\sigma'(q, h')$ est défini et égale à (q_1, E, π). On peut aisément vérifier que σ est gagnante. □

3.6 Décidabilité du contrôle centralisé

Nous formulons le problème de contrôle *centralisé* avec évènements inobservables et indiscernables, noté **PCCOI**, comme suit :

Soit un processus P, un automate simple $\mathcal{A}$ et un automate étendu $\mathcal{B}$. Existe-t-il un contrôleur Q tel que $Q \vDash \mathcal{B}$ et $P \times Q \vDash \mathcal{A}$?

Le problème **PCCOI** est décidable. C'est, comme pour le cas avec observation totale de la partie 2.2, la conséquence de l'existence d'un quotient d'automate simple par un processus $\mathcal{A}/P$ et de la satisfiabilité des automates étendus (ainsi que la clôture par intersection pour avoir les modèles de $\mathcal{A}/P \wedge \mathcal{B}$).

Nous avons prouvé que la satisfiabilité des automates étendus est décidable. Dans cette partie, nous prouvons le théorème 3.6.1 qui montre que cette opération de quotient existe aussi pour les automates étendus.

Théoreme 3.6.1 Soit $\mathcal{A}$ un automate étendu et P un processus. Il existe un automate étendu $\mathcal{A}/P$ tel que :

$$Q \vDash \mathcal{A}/P \text{ si et seulement si } P \times Q \vDash \mathcal{A}.$$

On appelle quotient d'un automate étendu par un processus l'opération qui à $\mathcal{A}$ et P associe l'automate $\mathcal{A}/P$. Nous allons maintenant démontrer le théorème 3.6.1

3.6.1 Définition de $\mathcal{A}/P$

Notation 3.6.2 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, r\rangle$ un automate étendu à parité et $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus.

Si $q \in Q^\forall$, $E \subset \Lambda$ et $s \in S$, on note $C(q,E,s) = C_1 \wedge C_2 \wedge C_3$ la condition où :
$C_1 \equiv$ pour tout $a \in A$, si $\rightarrow_a \in \delta(q, E \cup \lambda(s))$ alors $e(s,a)$ est défini.
$C_2 \equiv$ pour tout $a,b \in A$, si $\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ et $e(s,b)$ sont définis, alors $e(s,a) = e(s,b)$
$C_3 \equiv$ pour tout $a \in A$, $\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ est défini, alors $e(s,a) = s$.

Si $q \in Q^\exists$, $E \subset \Lambda$ et $s \in S$, on note $D(q,E,s) = D_1 \vee D_2 \vee D_3 \vee D_4 \vee D_5 \vee D_6 \vee D_7$ la condition où :
$D_1 \equiv$ il existe $a \in A$ tel que $e(s,a)$ n'est pas défini et $\twoheadrightarrow_a \in \delta(q, E \cup \lambda(s))$.
$D_2 \equiv$ il existe $a \in A$ tel que $e(s,a)$ n'est pas défini et $\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$.
$D_3 \equiv$ il existe $a \in A$ tel que $e(s,a)$ n'est pas défini et $\not\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$.
$D_4 \equiv$ il existe $a,b \in A$ tel que $e(s,a)$ et $e(s,b)$ sont définis mais pas égaux et $\not\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$.
$D_5 \equiv$ il existe $a \in A$ tel que $e(s,a)$ n'est pas défini et $\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$.
$D_6 \equiv$ il existe $a \in A$ tel que $e(s,a)$ n'est pas défini et $\not\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$.
$D_7 \equiv$ il existe $a \in A$ tel que $e(s,a)$ est défini mais pas égale à s et $\not\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$.

Définition 3.6.3 Soit $\mathcal{A}/P = \langle A, \Lambda, Q_/, Q_/^\exists, Q_/^\forall, (q_0,s_0), \delta_/, r_/ \rangle$ l'automate étendu défini comme suit :

- $Q_/^\exists = Q^\exists \times S \cup \{q_\perp\}$ et $Q_/^\forall = Q^\forall \times S \cup \{q_\top\}$.
- Pour tout $q \in Q$ et $s \in S$, $r_/(q,s) = r(q)$ et $r_/(q_\perp) = r_/(q_\top) = 0$
- Pour tout $E \subset \Lambda$, on a $\delta(q_\perp, E) = \delta(q_\top, E) = \emptyset$
- Si $E \subset \Lambda$, $(q,s) \in Q_/^\exists$ et $D(q,E,s)$ est vrai, alors $\delta((q,s),E) = \{(\epsilon, q_\top)\}$.
- Si $E \subset \Lambda$, $(q,s) \in Q_/^\forall$ et $C(q,E,s)$ est faux, alors $\delta((q,s),E) = \{(\epsilon, q_\perp)\}$.
- Si $E \subset \Lambda$, $(q,s) \in Q_/^\exists$ et $D(q,E,s)$ est faux, ou $(q,s) \in Q_/^\forall$ et $C(q,E,s)$ est vrai, alors $\delta_/((q,s),E)$ contient :
  - $\rightarrow_a$ si $\rightarrow_a \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ est défini.
  - $\twoheadrightarrow_a$ si $\twoheadrightarrow_a \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ est défini.
  - $\downdownarrows_{a,b}$ si $\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ et $e(s,b)$ sont définis et égaux.
  - $\not\downdownarrows_{a,b}$ si $\not\downdownarrows_{a,b} \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ et $e(s,b)$ sont définis et égaux.
  - $\circlearrowleft_a$ si $\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ est défini et égale à s.
  - $\not\circlearrowleft_a$ si $\not\circlearrowleft_a \in \delta(q, E \cup \lambda(s))$ et $e(s,a)$ est défini et égale à s.
  - $(a,(q_1,s_1))$ si $(a,q_1) \in \delta(q, E \cup \lambda(s))$ et $e(s,a) = s_1$.
  - $(\epsilon,(q_1,s))$ si $(\epsilon,q_1) \in \delta(q, E \cup \lambda(s))$.

3.6.2 Preuve : Si $Q \vDash \mathcal{A}/P$ alors $P \times Q \vDash \mathcal{A}$

Soit $Q = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ un processus.

Soit $\sigma_/ : Q^\exists \times S \times S' \to Q^\forall \times S \times S'$ une stratégie positionnelle gagnante dans le jeu $G(\mathcal{A}/P, Q)$. Nous assumons aussi que si il existe une arête depuis $((q,s),s')$ avec $q \in Q^\exists$ vers $((q_\top, s), s')$ ou vers $\top$, alors $\sigma_/((q,s),s')$ est égale à $((q_\top, s), s')$ ou à $\top$. Nous allons maintenant prouver qu'il existe une stratégie gagnante σ dans le jeu $G(\mathcal{A}, P \times Q)$.

Observons que si il existe une arête depuis $((q,s),s')$ vers $(q_\top, s')$ dans $G(\mathcal{A}/P, Q)$, alors $D(q, \lambda'(s'), s)$ est vrai et il y a une arête depuis $(q,(s,s'))$ vers $\top$ dans $G(\mathcal{A}, P \times Q)$. Supposons qu'il y a arête depuis $((q,s),s')$ vers $\top$ dans $G(\mathcal{A}/P, Q)$ et que $\sigma_/((q,s),s')$ est défini. Alors $D(q, \lambda'(s'), s)$ est faux et on a l'un des cas suivants :

- Soit $\rightarrow_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ est défini. Donc $\rightarrow_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $e(s,a)$ est défini.
- Soit $\nrightarrow_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ n'est pas défini. Donc $\nrightarrow_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $e(s,a)$ est défini.
- Soit $\downarrow\!\downarrow_{a,b} \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ ou $e'(s',b)$ n'est pas défini. Donc $\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$.
- Soit $\downarrow\!\downarrow_{a,b} \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ et $e'(s',b)$ sont définis et égaux. Donc $\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $e(s,a)$ et $e(s,b)$ sont définis et égaux.
- Soit $\not\downarrow\!\downarrow_{a,b} \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ ou $e'(s',b)$ n'est pas défini. Donc $\not\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$.
- Soit $\not\downarrow\!\downarrow_{a,b} \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ et $e'(s',b)$ sont définis mais pas égaux. Alors $\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$.
- Soit $\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ n'est pas défini. Donc $\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$.
- Soit $\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a) = s'$. Donc $\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $e(s,a) = s$.
- Soit $\not\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a)$ n'est pas défini. Donc $\not\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$.
- Soit $\not\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$ et $e'(s',a) \neq s'$. Donc $\not\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$.

Dans tous ces cas, on a une arête depuis $(q,(s,s'))$ vers $\top$ dans $G(\mathcal{A}, P \times Q)$. Donc la stratégie σ dans $G(\mathcal{A}, P \times Q)$ est identique à $\sigma_/$ excepté si $\sigma_/((q,s),s') = ((q_\top, s), s')$ alors $\sigma(q,(s,s')) = \top$.

Supposons qu'il existe une arête depuis $(q,(s,s'))$ avec $q \in Q^\forall$ vers $(q_1,(s_1,s'_1))$ dans $G(\mathcal{A}, P \times Q)$ tel que $((q,s),s')$ est accessible par $\sigma_/$ dans $G(\mathcal{A}/P, Q)$. $C(q, \lambda'(s'), s)$ doit être vrai car autrement, il y a une arête depuis $((q,s),s')$ vers la position perdante $(q_\perp, s')$ dans $G(\mathcal{A}/P, Q)$. Ainsi il y a deux cas ; soit $(a, q_1) \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $e(s,a) = s_1$ et $e'(s',a) = s'_1$ ou soit

$(\epsilon, q_1) \in \delta(q, \lambda(s) \cup \lambda'(s'))$ et $s = s_1$ et $s' = s'_1$. Mais dans les deux cas, il y a une arête depuis $((q,s),s')$ vers $((q_1,s_1),s'_1)$ dans $G(\mathcal{A}/P,Q)$. Il reste à vérifier qu'il n'y a pas d'arêtes depuis $(q,(s,s'))$ avec $q \in Q^\forall$ vers $\bot$ dans $G(\mathcal{A}, P \times Q)$.

- Si $\rightarrow_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$ alors $e(s,a)$ est défini à cause de la partie C_1 de la condition $C(q, \lambda'(s'), s)$. Donc $\rightarrow_a \in \delta_/((q,s), \lambda'(s'))$ et alors $e'(s',a)$ doit être défini car $((q,s),s')$ est une position gagnante dans $G(\mathcal{A}/P,Q)$.
- Supposons que $\nrightarrow_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$. Si $e(s,a)$ n'est pas défini alors $e \times e'((s,s'),a)$ n'est pas définit. Si $e(s,a)$ est défini alors $\nrightarrow_a \in \delta_/((q,s), \lambda'(s'))$. Puisque $((q,s),s')$ est une position gagnante dans $G(\mathcal{A}/P,Q)$, $e'(s',a)$ doit être défini.
- Supposons que $\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$. Soit $e(s,a)$ ou $e(s,a)$ ne sont pas défini, soit ils sont définis et la partie C_2 de la condition $C(q, \lambda'(s'), s)$ implique qu'ils sont égaux. dans ce dernier cas, $\downarrow\!\downarrow_{a,b} \in \delta_/((q,s), \lambda'(s'))$. Puisque $((q,s),s')$ est une position gagnante dans $G(\mathcal{A}/P,Q)$, si $e'(s',a)$ et $e'(s',b)$ sont définis, alors ils doivent être égaux.
- Supposons que $\not\downarrow\!\downarrow_{a,b} \in \delta(q, \lambda(s) \cup \lambda'(s'))$. Si $e(s,a)$ et $e(s,b)$ sont définis et égaux, alors $\not\downarrow\!\downarrow_{a,b} \in \delta((q,s), \lambda'(s'))$. Donc si $e'(s',a)$ et $e'(s',b)$ sont définis, alors ils sont égaux.
- Supposons que $\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$. Soit $e(s,a)$ n'est pas défini ou soit $e(s,a)$ est défini et la partie C_3 de la condition $C(q, \lambda'(s'), s)$ implique que $e(s,a) = s$. Dans ce dernier cas, $\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$. Puisque $((q,s),s')$ est une position gagnante dans $G(\mathcal{A}/P,Q)$, si $e'(s',a)$ est défini alors $e'(s',a) = s$.
- Supposons que $\not\circlearrowleft_a \in \delta(q, \lambda(s) \cup \lambda'(s'))$. Si $e(s,a)$ est défini et égale à s, alors $\not\circlearrowleft_a \in \delta_/((q,s), \lambda'(s'))$. Donc si $e'(s',a)$ est défini, alors $e'(s',a) \neq s$.

□

3.6.3 Preuve : Si $P \times Q \vDash \mathcal{A}$ alors $Q \vDash \mathcal{A}/P$

Soit $\sigma : Q^\exists \times S \times S' \rightarrow Q^\forall \times S \times S'$ une stratégie gagnante positionnelle dans le jeu $G(\mathcal{A}, P \times Q)$. Nous assumons aussi que si il y a une arête depuis $(q,(s,s'))$ avec $q \in Q^\exists$ vers $\top$, alors $\sigma(q,(s,s'))$ est égale à $\top$.

Nous allons maintenant prouver qu'il existe une stratégie gagnante $\sigma_/$ dans le jeu $G(\mathcal{A}/P,Q)$. Supposons que il y a une arête depuis $(q,(s,s'))$ avec $q \in Q^\exists$ vers $\top$ dans $G(\mathcal{A}, P \times Q)$. Alors on peut prouver que soit il y a une arête depuis $((q,s),s')$ vers $\top$ ou soit il y a arête depuis $((q,s),s')$ vers $(q_\top, s')$ dans $G(\mathcal{A}/P,Q)$. Ainsi, nous définissons $\sigma_/$ comme étant égale à σ excepté si il y a une arête depuis $((q,s),s')$ vers $(q_\top, s')$ alors $\sigma_/((q,s),s') = ((q_\top, s), s')$.

Supposons qu'il y a une arête depuis $((q,s),s')$ avec $q \in Q^\forall$ vers $((q_1,s_1),s'_1)$

dans $G(\mathcal{A}/P,Q)$ tel que $(q,(s,s'))$ est accessible par σ dans $G(\mathcal{A},P\times Q)$. $D(q,\lambda'(s'),s)$ doit être faux car autrement il y a une arête depuis $(q,(s,s'))$ vers la position perdante $\perp$ dans $G(\mathcal{A},P\times Q)$. Ainsi, il y a deux cas. Soit $(a,(q_1,s_1))\in\delta_/((q,s),\lambda'(s'))$ et $e'(s',a)=s'_1$ et alors $(a,q_1)\in\delta(q)$ et $e(s,a)=s_1$. Ou soit $(\epsilon,(q_1,s))\in\delta_/((q,s),\lambda'(s'))$ et $s=s_1$, $s'=s'_1$ et alors $(\epsilon,q_1)\in\delta(q,\lambda'(s')\cup\lambda(s)))$ Dans ces deux cas, il y a une arête depuis $(q,(s,s'))$ vers $(q_1,(s_1,s'_1))$ dans $G(\mathcal{A},P\times Q)$.

Il reste à vérifier qu'il n'y a pas d'arêtes depuis $((q,s),s')$ avec $q\in Q^\forall$ vers $\perp$ ou vers $(q_\perp,s')$ dans $G(\mathcal{A}/P,Q)$.

- Si $\rightarrow_a\in\delta_/((q,s),\lambda'(s'))$ alors $\rightarrow_a\in\delta(q,\lambda'(s')\cup\lambda(s)))$ et $e(s,a)$ est défini. Donc la partie C_1 de la condition $C(q,\lambda'(s'),s)$ est vrai. Puisque $(q,(s,s'))$ est une position gagnante dans $G(\mathcal{A},P\times Q)$, $e'(s',a)$ doit être défini.
- Si $\nrightarrow_a\in\delta_/((q,s),\lambda'(s'))$ alors $e(s,a)$ est défini et $\nrightarrow_a\in\delta(q,\lambda'(s')\cup\lambda(s)))$, donc $e'(s',a)$ n'est pas défini.
- Supposons que $\downarrow\!\downarrow_{a,b}\in\delta_/((q,s),\lambda'(s'))$. Alors $\downarrow\!\downarrow_{a,b}\in\delta(q,\lambda'(s')\cup\lambda(s))$, $e(s,a)$ et $e(s,b)$ sont définis et égaux. Donc la partie C_2 de la condition $C(q,\lambda'(s'),s)$ est vrai. Puisque $(q,(s,s'))$ est une position gagnante dans $G(\mathcal{A},P\times Q)$, si $e'(s',a)$ et $e'(s',b)$ sont définis, alors ils sont égaux.
- Supposons que $\not\downarrow\!\downarrow_{a,b}\in\delta_/((q,s),\lambda'(s'))$. Alors $e(s,a)$ et $e(s,b)$ sont définis et égaux et $\not\downarrow\!\downarrow_{a,b}\in\delta(q,\lambda'(s')\cup\lambda(s))$. Donc si $e'(s',a)$ et $e'(s',b)$ sont définis, ils ne sont pas égaux.
- Supposons que $\circlearrowleft_a\in\delta_/((q,s),\lambda'(s'))$ alors $\circlearrowleft_a\in\delta(q,\lambda'(s')\cup\lambda(s))$ et $e(s,a)=s$. Donc la partie C_3 de la condition $C(q,\lambda'(s'),s)$ est vrai. Puisque $(q,(s,s'))$ est une position gagnante dans $G(\mathcal{A},P\times Q)$, si $e'(s',a)$ est défini, alors il est égale à s.
- Supposons $\not\circlearrowleft_a\in\delta_/((q,s),\lambda'(s'))$. Alors $e(s,a)=s$ et $\not\circlearrowleft_a\in\delta(q,\lambda'(s')\cup\lambda(s)))$. Donc si $e'(s',a)$ est défini alors $e'(s',a)\neq s$.
- Puisque les parties C_1, C_2 et C_3 de la condition $C(q,\lambda'(s'),s)$ sont vrais, $C(q,\lambda'(s'),s)$ est vrai et donc il n'y a pas d'arête depuis $((q,s),s')$ vers $(q_\perp,s')$ dans $G(\mathcal{A}/P,Q)$.

□

3.7 Comparaison des automates simples et étendus

Même si la définition d'un automate étendu est plus générale que celle d'un automate simple, on peut se demander si cette extension permet pour autant de définir des familles de processus différentes.

Les automates étendus permettent effectivement de définir des familles de processus qui ne sont pas définissables par des automates simples. En effet les automates simples définissent des classes de modèles invariantes par bisimulation (voir définition 0.4.4 et lemme 1.5.13) alors que ce n'est pas le cas des automates étendus comme le montre le lemme 3.7.2.

Définition 3.7.1 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ et $P' = \langle A, \Lambda, S', s'_0, e', \lambda'\rangle$ deux processus. On dit que P est *moins défini* que P' et l'on note $P \leq_d P'$, si $P \leq P'$ (voir Définition 0.4.2) et, pour tout état (s, s') du produit $P \times P'$, les propriétés suivantes sont vérifiées :

- Si $e(s, a)$ est défini et égale à s, alors $e'(s', a) = s'$.
- Si $e(s, a)$ et $e(s, b)$ sont définis et égaux, alors $e'(s', a) = e'(s', b)$.
- $\lambda_P(s) \supseteq \lambda_{P'}(s')$

On note $P =_d P'$ quand $P \leq_d P'$ et $P' \leq_d P$.

Lemme 3.7.2 Soit P et P' deux processus tels que $P =_d P'$ et $\mathcal{A}$ un automate étendu. Alors $P \vDash \mathcal{A}$ si et seulement si $P' \vDash \mathcal{A}$.

Preuve : La relation $P \leq_d P'$ décrite ci-dessus, stipule que les comportements de P sont inclus dans ceux de P' et que les évènements inobservables pour P sont inobservables pour P' et, de même, les évènements indiscernables pour P sont indiscernables pour P'. La propriété $\lambda_P(s) = \lambda_{P'}(s')$ nous assure que le processus P' a les mêmes propriétés locales que P. On peut alors vérifier que $P =_d P'$ implique que P et P' sont des modèles des mêmes automates étendus. □

Remarque 3.7.3 Cependant, même si les familles de processus spécifiées par les automates simples et étendus sont différentes, ces familles sont peut-être identiques à bisimulation près. La proposition suivante l'affirme pour les automates étendus dans $AUTO(A, \Lambda, \varnothing, \not\downarrow)$ (voir notation 3.2.4).

Proposition 3.7.4 Soit $\mathcal{A} \in AUTO(A, \Lambda, \varnothing, \not\downarrow)$ et P un processus. Alors il existe un automate simple $\mathcal{B}$ tel que $P \vDash \mathcal{B}$ si et seulement si il existe Q bisimilaire à P tel que $Q \vDash \mathcal{A}$.

Preuve : Soit $\mathcal{A} \in AUTO(A, \Lambda, \varnothing, \not\downarrow)$ et $\widetilde{\mathcal{A}}$ l'automate simple identique à $\mathcal{A}$ excepté le fait qu'on a enlevé toutes les transitions dans $A \times \{\varnothing\}$ et dans $A \times A \times \{\not\downarrow\}$. $P \vDash \widetilde{\mathcal{A}}$ si et seulement si il existe Q bisimilaire à P tel que $Q \vDash \mathcal{A}$.

On a, évidement, $Q \vDash \mathcal{A}$ implique $Q \vDash \widetilde{\mathcal{A}}$. Donc, si P est bisimilaire à Q, alors $P \vDash \widetilde{\mathcal{A}}$. Réciproquement, supposons que $P \vDash \widetilde{\mathcal{A}}$ avec $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$. Soit $Q = \langle A, \Lambda, S', s_0, e', \lambda'\rangle$ avec :

- $S' = (S \cup \{L_s : s \in S\} \times A) \cup \{s_0\}$
- $\lambda'(s_0) = \lambda(s_0)$
- Pour tout $s \in A$ et $a \in A$ on a $\lambda'(s,a) = \lambda'(L_s,a) = \lambda(s)$
- $D_Q(s_0) = D_P(s_0)$
- Pour tout $s \in A$ et $a \in A$ on a $D_Q(s,a) = D_Q(L_s,a) = D_P(s)$
- Pour tout $a \in D_P(s_0)$, on a :

$$e'(s_0,a) = \begin{cases} (L_{s_0},a) & \text{si } e(s_0,a) = s_0 \\ (e(s_0,a),a) & \text{si } e(s_0,a) \neq s_0 \end{cases}$$

- Pour tout $s \in S$ et $a,b \in D_P(s))$, on a $e'((L_s,a),b) = (e(s,b),b)$
- Pour tout $s \in S$ et $a,b \in D_P(s)$, on a :

$$e'((s,a),b) = \begin{cases} (L_s,b) & \text{si } a = b \text{ et } e(s,b) = s \\ (e(s,b),b) & \text{sinon} \end{cases}$$

On peut vérifier que Q est bisimilaire à P et que $Q \vDash \mathcal{A}$. □

La proposition 3.7.4 ne peut être étendue aux automates dans $AUTO(A,\Lambda,\circlearrowleft)$ ou dans $AUTO(A,\Lambda,\downdownarrows)$ comme le montre la proposition 3.7.5.

Proposition 3.7.5 Il existe des automates $\mathcal{A} \in AUTO(A,\Lambda,\circlearrowleft)$ et $\mathcal{B} \in AUTO(A,\Lambda,\downdownarrows)$ tel que les ensembles $\{P :$ il existe $Q \vDash \mathcal{A}$ tel que $P =_b Q\}$ et $\{P :$ il existe $Q \vDash \mathcal{B}$ tel que $P =_b Q\}$ ne sont pas définissables par des automates simples.

Preuve : C'est la conséquence de résultats que nous prouverons ultérieurement aux parties 5.1 et 5.3.

Tout d'abord, on peut vérifier que si deux processus R et R' sont bisimilaires, alors, pour tout processus R'', $R \times R''$ et $R' \times R''$ sont bisimilaires. Soit P un processus, $\mathcal{A} \in AUTO(A,\Lambda)$ et $\mathcal{B}_1, \mathcal{B}_2 \in AUTO(A,\Lambda,\circlearrowleft)$. Supposons qu'il existe des automates simples $\mathcal{B}_i^S \in AUTO(A,\Lambda)$, $i = 1,2$, tel que :

$$Mod(\mathcal{B}_i^S) = \{Q' :\text{ il existe } Q \vDash \mathcal{B}_i \text{ tel que } Q =_b Q'\}$$

Si il existe $Q_i \vDash \mathcal{B}_i$ tel que $P \times Q_1 \times Q_2 \vDash \mathcal{A}$ alors, évidement, $Q_i \vDash \mathcal{B}_i^S$. Réciproquement, supposons maintenant qu'il existe $Q_i' \vDash \mathcal{B}_i^S$ tel que $P \times Q_1' \times Q_2' \vDash \mathcal{A}$. Alors il existe $Q_i \vDash \mathcal{B}_i^S$ tel que Q_i et Q_i' sont bisimilaires. Donc $P \times Q_1 \times Q_2 \vDash \mathcal{A}$. Le cas où $\mathcal{B}_1 \in AUTO(A,\Lambda,\downdownarrows)$ est similaire.

Nous prouverons par la suite, que le problème de trouver des contrôleurs $Q_i \vDash \mathcal{B}_i$ tel que $P \times Q_1 \times Q_2 \vDash \mathcal{A}$ (i.e. le problème de contrôle décentralisé) est *décidable* si $\mathcal{B}_i \in AUTO(A,\Lambda)$ (voir section 6.3) et *indécidable* si $\mathcal{B}_i \in AUTO(A,\Lambda,\circlearrowleft)$ (voir partie 5.1) ou si $\mathcal{B}_i \in AUTO(A,\Lambda,\downdownarrows)$ (voir partie 5.3).

Or nous venons de prouver que, si la proposition 3.7.5 était fausse, le problème de contrôle décentralisé avec $\mathcal{B}_i \in AUTO(A, \Lambda, \circlearrowleft)$ ou $\mathcal{B}_i \in AUTO(A, \Lambda, \downdownarrows)$ se déduirait au problème où $\mathcal{B}_i$ les sont des automates simples.

Donc, si la proposition 3.7.5 était fausse le problème de contrôle décentralisé pour les automates simples serait indécidable. Contradiction. □

3.8 Propriétés des automates étendus

Dans cette partie nous présentons certaines propriétés des automates étendus qui peuvent être vues comme des extensions de résultats pour les automates simples et qui nous serons utiles dans les chapitres ultérieurs.

3.8.1 Automate étendu sans propriétés locales

Définition 3.8.1 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ un automate étendu tel que, pour tout $q \in Q$ et pour tout $E, E' \subset \Lambda$, $\delta(q, E) = \delta(q, E')$. On dit alors que $\mathcal{A}$ est sans propriétés locales et on note $\mathcal{A} = \langle A, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, Acc\rangle$ avec :

$$\delta : Q \to \mathcal{P}\Big(\big((A \cup \{\varepsilon\}) \times Q\big) \cup \big(A \times \{\to, \nrightarrow, \circlearrowleft, \varnothing\}\big) \cup \big(A \times A \times \{\downdownarrows, \not\downdownarrows\}\big)\Big)$$

Dans cette partie, nous présentons un moyen d'identifier les familles de processus sur l'alphabet A définies par des automates étendus, à des familles de processus *sur l'alphabet* $A \cup \Lambda$ définies par des automates étendus *sans propriétés locales.*

Définition 3.8.2 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus sur l'alphabet A. On note P_Λ le processus $\langle A \cup \Lambda, S \cup \Lambda, s_0, e_\Lambda\rangle$ sur l'alphabet $A \cup \Lambda$ avec $e_\Lambda : (S \cup \Lambda) \times (A \cup \Lambda) \to S \cup \Lambda$ et, pour tout $s \in S$, $a \in A$ et $l \in \Lambda$,

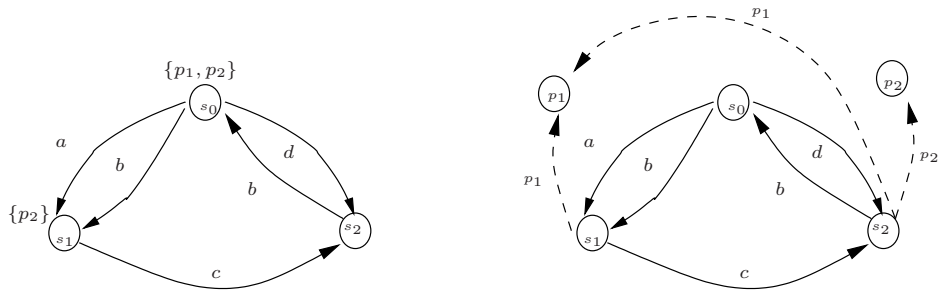
- $e_\Lambda(s, a)$ est défini et égale à $e(s, a)$ si $e(s, a)$ est défini.
- $e_\Lambda(s, l)$ est défini et égale à l si $l \notin \lambda(s)$
- $D_{P_\Lambda}(l) = \emptyset$ (voir Définition 0.3.2)

La figure 3.2 illustre la définition 3.8.2 avec $\Lambda = \{p_1, p_2\}$.

Proposition 3.8.3 Soit $P, Q \in PROC(A, \Lambda)$ (voir Définition 0.3.1). Alors

$$P_\Lambda \times Q_\Lambda = (P \times Q)_\Lambda$$

Preuve : Soit (s, s') un état de $P \times Q$. On a $D_{(P\times Q)_\Lambda}(s, s') = D_{(P\times Q)}(s, s') \cup (\Lambda - \lambda_{P\times Q}(s, s')) = (D_P(s) \cap D_Q(s')) \cup (\Lambda - (\lambda_P(s) \cup \lambda_Q(s')) = (D_P(s) \cup (\Lambda - \lambda_P(s))) \cap (D_Q(s') \cup (\Lambda - \lambda_Q(s'))) = D_{P_\Lambda}(s) \cap D_{Q_\Lambda}(s')$ □



Processus P Processus P_Λ

FIG. 3.2 –

Définition 3.8.4 Soit Q un processus sur l'alphabet $A \cup \Lambda$ sans propriété locale, c'est à dire $Q \in PROC(A \cup \Lambda, \emptyset)$ (voir Définition 0.3.1). On dit que Q est 2^Λ-*étiqueté* s'il existe un processus $P \in PROC(A, \Lambda)$ tel que $P_\Lambda = Q$.

Définition 3.8.5 Soit $LABEL_\Lambda = \langle A \cup \Lambda, Q, \emptyset, \{q_0, q_\emptyset\}, q_0, \delta, Q^\omega \rangle$ l'automate simple sans propriété locale tel que :

$$\delta(q_0) = \{(a, q_0) : a \in A\} \cup \{(l, q_\emptyset) : l \in \Lambda\} \text{ et } \delta(q_\emptyset) = \{\not\rightarrow_x \ : x \in A \cup \Lambda\}$$

Lemme 3.8.6 Soit $Q \in PROC(A \cup \Lambda, \emptyset)$ un processus. Q est 2^Λ-étiqueté si et seulement si $Q \vDash LABEL_\Lambda$.

Définition 3.8.7 Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ un automate étendu. Soit $\mathcal{A}_\Lambda = \langle A \cup \Lambda, \emptyset, Q_\Lambda, {Q_\Lambda}^\exists, {Q_\Lambda}^\forall, q_0, \delta_\Lambda, Acc_\Lambda \rangle$ l'automate sans propriété locale avec :

- ${Q^\exists}_\Lambda = Q \cup (Q^\exists \times \Lambda)$
- ${Q^\forall}_\Lambda = (Q^\forall \times \Lambda) \cup \{T_q^E : q \in Q^\exists \text{ et } E \subseteq \Lambda\} \cup \{q_\emptyset\}$
- Pour tout $q \in Q$ et $E \in 2^\Lambda$ on a :

$$\delta_\Lambda(q) = \{(\varepsilon, T_q^E) : E \subseteq \Lambda\}$$
$$\delta_\Lambda(T_q^E) = \{\rightarrow_l, (l, q_\emptyset) : l \in \Lambda - E\} \cup \{\not\rightarrow_l : l \in E\} \cup \{(\varepsilon, (q, E))\}$$
$$\delta_\Lambda(q, E) = \delta(q, E)$$

- $\delta_\Lambda(q_\emptyset) = \{\not\rightarrow_x \ : x \in A \cup \Lambda\}$
- Acc_Λ est l'ensemble des suites de la forme $q_0 \alpha q_1 \alpha \cdots$ où $\alpha \in (Q_\Lambda - Q)^*$ tel que $q_0 q_1 \cdots$ appartient à Acc.

Lemme 3.8.8 Soit $Q \in PROC(A \cup \Lambda, \emptyset)$ un processus. $Q \vDash \mathcal{A}_\Lambda$ si et seulement si il existe un processus $P \in PROC(A, \Lambda)$ tel que $P \vDash \mathcal{A}$ et $P_\Lambda = Q$.

Preuve : Soit $Q = \langle A \cup \Lambda, S, s_0, e\rangle$ un processus tel que $Q \vDash \mathcal{A}_\Lambda$ et σ_Λ une stratégie gagnante dans $G(\mathcal{A}_\Lambda, Q)$.

Soit (q, s) une position du jeu $G(\mathcal{A}, P)$ avec $q \in Q^\exists$. Soit P le processus $\langle A, \Lambda, S, s_0, e_P, \lambda_P\rangle$ tel que, pour tout $s \in S$, $D_P(s) = D_Q(s) \cap A$ et pour tout $a \in A$, $e_P(s, a) = e(s, a)$ et $\lambda_P(s) = \Lambda - (D_Q(s) \cap \Lambda)$.

Construisons une stratégie gagnante σ dans $G(\mathcal{A}, P)$. Soit (q, s) une position du jeu $G(\mathcal{A}, P)$ avec $q \in Q^\exists$. Dans le jeu $G(\mathcal{A}_\Lambda, Q)$, pour qu'une position de la forme (T_q^E, s) soit gagnante, il est nécessaire que $D_Q(s) \cap \Lambda = \Lambda - E$ et que pour tout $l \in \Lambda - E$, $D_Q(e(s, l)) = \emptyset$. Donc, il est nécessaire que $\lambda_P(s) = \Lambda - (D_Q(s) \cap \Lambda) = \Lambda - (\Lambda - E) = E$. Donc si $\sigma_\Lambda(q, s) = (T_q^E, s)$ et $\sigma_\Lambda((q, E), s) = (q', s')$ on définit $\sigma(q, s) = (q', s')$. On peut alors vérifier qu'une arête de (q, s) à (q', s') compatible avec σ dans $G(\mathcal{A}, P)$, correspond à la suite sommets $(q, s), (T_q^E, s), (q, E, s), (q', s')$ dans $G(\mathcal{A}_\Lambda, Q)$ avec $\lambda_P(s) = E$.

Réciproquement, soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus tel que $P \vDash \mathcal{A}$. On fait correspondre à l'arête de (q, s) à (q', s') dans $G(\mathcal{A}, P)$, la suite sommets $(q, s), (T_q^{\lambda(s)}, s), ((q, \lambda(s)), s), (q', s')$ dans $G(\mathcal{A}_\Lambda, P_\Lambda)$. On peut alors vérifier que $P_\Lambda \vDash \mathcal{A}_\Lambda$ □

3.8.2 Autres écritures des automates étendus

Définition 3.8.9 Soit $\mathcal{A} = \langle A, \lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate étendu nondéterministe. On dit que $\mathcal{A}$ est *spécifié* si pour tout $q \in Q^\forall$ et tout $E \subseteq \Lambda$ on a :

- $\{a :\rightarrow_a \in \delta(q, E)\} \cup \{a :\nrightarrow_a \in \delta(q, E)\} = A$,
- Si $(a, q') \in \delta(q, E)$ alors $\rightarrow_a \in \delta(q, E)$.

On dit que $\mathcal{A}$ est *complètement spécifié* si $\mathcal{A}$ est spécifié et si pour tout $q \in Q^\forall$ et tout $E \subseteq \Lambda$ on a :

- $\{a :\circlearrowleft_a \in \delta(q, E)\} \cup \{a : \not\circlearrowleft_a \in \delta(q, E)\} = A$,
- $\{(a, b) :\downdownarrows_{a,b} \in \delta(q, E)\} \cup \{(a, b) :\circlearrowleft_a, \circlearrowleft_b \in \delta(q, E)\} \cup \{(a, b) : \not\downdownarrows_{a,b} \in \delta(q, E)\} = A \times A$,
- Si $\circlearrowleft_a \in \delta(q, E)$ alors $\rightarrow_a \in \delta(q, E)$,
- Si $\downdownarrows_{a,b} \in \delta(q, E)$ alors $\rightarrow_a, \rightarrow_b \in \delta(q, E)$, $\circlearrowleft_a \notin \delta(q, E)$ et $\circlearrowleft_b \notin \delta(q, E)$.

Définition 3.8.10 Soit un automate étendu $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ nondéterministe et $q \in Q$.
On note $\mathcal{A}_q$ l'automate étendu $\langle A, \Lambda, Q, Q^\exists, Q^\forall, q, \delta, Acc\rangle$ identique à $\mathcal{A}$ excepté pour l'état initial changé en q. On dit que $\mathcal{A}$ est *élagué* si pour tout $q \in Q$, $\mathcal{A}_q$ a un modèle.

Théoreme 3.8.11 Tout automate étendu nondéterministe est équivalent à un automate étendu nondéterministe complet, complètement spécifié et élagué.

Preuve : Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate étendu nondéterministe.

Soit $specif(\mathcal{A})$ l'automate $\langle A, \Lambda, specif(Q), Q^\exists, Q^\forall \times \Pi_A, q_0, specif(\delta), Acc\rangle$ l'automate complètement spécifié définit comme suit (voir notation 3.4.2).

- Pour tout $q \in Q^\exists$ et tout $E \subseteq \Lambda$, $specif(\delta)(q, E)$ contient $(\varepsilon, (q', \pi))$ avec
  $\pi = \{N, A_0, A_1, \ldots A_p\} \in \Pi_A$ si :
  - $(\varepsilon, q') \in \delta(q)$
  - Si $\rightarrow_a \in \delta(q)$ alors $a \notin N$
  - Si $\nrightarrow_a \in \delta(q)$ alors $a \in N$
  - Si $\circlearrowleft_a \in \delta(q)$ alors $a \in L$
  - Si $\not\circlearrowleft_a \in \delta(q)$ alors $a \notin L$
  - Si $\downdownarrows_{a,b} \in \delta(q)$ alors il existe i tel que $a, b \in A_i$
  - Si $\not\downdownarrows_{a,b} \in \delta(q)$ alors pour tout i tel que $a, b \notin A_i$ et $a, b \notin L$
- Pour tout $q' \in Q^\forall$, $E \subseteq \Lambda$ et $\pi \in \Pi_A$, $specif(\delta)((q', \pi), E)$ contient :
  - $\delta(q')$
  - $\rightarrow_a \in \delta(q)$ si $a \notin N$
  - $\nrightarrow_a \in \delta(q)$ si $a \in N$
  - $\circlearrowleft_a \in \delta(q)$ si $a \in L$
  - $\not\circlearrowleft_a \in \delta(q)$ si $a \notin L$
  - $\downdownarrows_{a,b} \in \delta(q)$ si il existe i tel que $a, b \in A_i$
  - $\not\downdownarrows_{a,b} \in \delta(q)$ si pour tout i, $a, b \notin A_i$ et $a, b \notin L$

On peut alors vérifier que $\mathcal{A}$ et $specif(\mathcal{A})$ ont les mêmes modèles.

Soit $Q_\emptyset$ l'ensemble des états q tel que $\mathcal{A}_q$ n'a pas de modèle. Soit $elag(\mathcal{A})$ l'automate $\langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, elag(\delta), Acc\rangle$ définit comme suit :

- Pour tout $q \in Q^\exists$ et $E \subseteq \Lambda$, $elag(\delta)(q, E) = \delta(q, E) - \{(\varepsilon, q) : q \in Q_\emptyset\}$
- Pour tout $q \in Q^\forall - Q_\emptyset$ et $E \subseteq \Lambda$

$$elag(\delta)(q, E) = \delta(q, E) \cup \{\nrightarrow_a : \{a\} \times Q_\emptyset \cap \delta(q, E) \neq \emptyset\} - \{(a, q') : q' \in Q_\emptyset\}$$

On peut vérifier que $\mathcal{A}$ et $elag(\mathcal{A})$ ont les mêmes modèles.

On peut alors vérifier que l'automate $elague(specif(complet(\mathcal{A})))$ est un automate complet, complètement spécifié et élagué (voir la preuve de la proposition 3.2.8 pour la définition de $complet(\mathcal{A})$). □

3.8.3 Combinaison booléenne d'automates étendus

Définition 3.8.12

- On dit que l'ensemble des automates étendus est *clos par union* si pour tous automates étendus $\mathcal{A}, \mathcal{B}$, il existe un automate étendu dont l'ensemble des modèles est $Mod(\mathcal{A}) \cup Mod(\mathcal{B})$.
- On dit que l'ensemble des automates étendus est *clos par intersection* si pour tous automates étendus $\mathcal{A}, \mathcal{B}$, il existe un automate étendu dont l'ensemble des modèles est $Mod(\mathcal{A}) \cap Mod(\mathcal{B})$.
- On dit que l'ensemble des automates étendus est *clos par complémentation* si pour tout automate étendu $\mathcal{A}$, il existe un automate étendu dont l'ensemble des modèles est $AUTO(A, \Lambda) - Mod(\mathcal{A})$.

Théoreme 3.8.13 L'ensemble des automates étendus est clos par union, intersection et complémentation.

Notation 3.8.14 Soit $\mathcal{A}, \mathcal{B}$ deux automates étendus. On note $\mathcal{A} \vee \mathcal{B}$, $\mathcal{A} \wedge \mathcal{B}$ et $\mathcal{A}^C$ des automates étendus quelconques dont les modèles sont respectivement l'union des modèles de $\mathcal{A}$ et $\mathcal{B}$, leurs intersection et le complément de $Mod(\mathcal{A})$ dans $AUTO(A, \Lambda)$.

Preuve du théorème 3.8.13 : Soit $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ et $\mathcal{B} = \langle A, \Lambda, Q', Q^{\exists\prime}, Q^{\forall\prime}, q'_0, \delta', Acc' \rangle$ deux automates étendus. On suppose de plus que Q et Q' sont distincts.

Soit $\mathcal{A} \vee \mathcal{B}$ l'automate $\langle A, \Lambda, Q \cup Q', Q^\exists \cup Q^{\exists\prime} \cup \{p_0\}, Q^\forall \cup Q^{\forall\prime}, p_0, \delta_\vee, Acc \cup Acc' \rangle$ un automate étendu tel que $\delta_\vee(p_0) = \{(\varepsilon, q_0), (\varepsilon, q'_0)\}$ et

$$\delta_\vee(q, E) = \begin{cases} \delta(q, E) & \text{si } q \in Q \\ \delta'(q, E) & \text{si } q \in Q' \end{cases}$$

Soit P un processus. On peut vérifier que $P \vDash \mathcal{A} \vee \mathcal{B}$ si et seulement si $P \vDash \mathcal{A}$ ou $P \vDash \mathcal{B}$.

Soit $\mathcal{A} \wedge \mathcal{B}$ l'automate $\langle A, \Lambda, Q \cup Q', Q^\exists \cup Q^{\exists\prime}, Q^\forall \cup Q^{\forall\prime} \cup \{p_0\}, p_0, \delta_\vee, Acc \cup Acc' \rangle$ un automate étendu tel que $\delta_\wedge(p_0) = \{(\varepsilon, q_0), (\varepsilon, q'_0)\}$ et

$$\delta_\wedge(q, E) = \begin{cases} \delta(q, E) & \text{si } q \in Q \\ \delta'(q, E) & \text{si } q \in Q' \end{cases}$$

Soit P un processus. On peut vérifier que $P \vDash \mathcal{A} \wedge \mathcal{B}$ si et seulement si $P \vDash \mathcal{A}$ et $P \vDash \mathcal{B}$. (On peut évidement définir $\mathcal{A} \wedge \mathcal{B}$ par $(\mathcal{A}^C \vee \mathcal{B}^C)^C$.)

Montrons maintenant que $\mathcal{A}$ est clos par complémentation. Supposons d'abord que $\mathcal{A}$ est spécifié. Soit $\mathcal{A}^C$ l'automate $\langle A, \Lambda, Q, {Q^\exists}_C, {Q^\forall}_C, q_0, \delta_C, Acc_C\rangle$ tel que

$$ {Q^\exists}_C = Q^\forall, \quad {Q^\forall}_C = Q^\exists \text{ et } Acc_C = Q^\omega - Acc $$

Pour tout $q \in Q$ et $E \subseteq \Lambda$, $\delta_C(q, E)$ contient :

- (a, q') si $(a, q') \in \delta(q, E)$
- $\rightarrow_a$ si $\nrightarrow_a \in \delta(q, E)$
- $\nrightarrow_a$ si $\rightarrow_a \in \delta(q, E)$
- $\circlearrowleft_a$ si $\not\circlearrowleft_a \in \delta(q, E)$
- $\not\circlearrowleft_a$ si $\circlearrowleft_a \in \delta(q, E)$
- $\downdownarrows_{a,b}$ si $\not\rightrightarrows_{a,b} \in \delta(q, E)$
- $\not\rightrightarrows_{a,b}$ si $\downdownarrows_{a,b} \in \delta(q, E)$

Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus tel que $P \vDash \mathcal{A}$. Soit $\sigma : Q^\exists \times S \rightarrow Q$ une stratégie gagnante positionnelle dans le jeu $G(\mathcal{A}, P)$. Soit $\sigma_C : Q^\forall \times S \rightarrow Q$ une stratégie dans le jeu $G(\mathcal{A}^C, P)$. Soit p le chemin compatible avec σ et σ_C (voir définition 1.3.3). Si p est infini, alors la stratégie σ_C est perdante car $p \in Acc$ puisque σ est gagnante. Si p est fini, alors, puisque σ est gagnante, il existe une position $(q, s) \in Q^\forall \times S$ du chemin p dans le jeu $G(\mathcal{A}, P)$ tel que l'unique arête sortante depuis (q, s) va vers $\top$. Donc, d'après la fonction de transition δ_C, il y a depuis (q, s) une arête vers $\bot$. Donc $P \nvDash \mathcal{A}^C$. La réciproque se démontre de façon identique car $(\mathcal{A}^C)^C = \mathcal{A}$. □

3.9 Conclusion

Le contrôle centralisé avec événements indiscernables et événements inobservables est donc décidable si les spécifications sont exprimées par des automates étendus. Comme l'a présenté la partie 3.7, l'expressivité de ces automates est plus grande que celle du μ-calcul. Ils permettent de prendre en compte les spécifications classiques d'information partielle (voir [LW88, CDFV88]) mais aussi, comme pour la contrôlabilité, une approche dynamique des contraintes d'observations et d'indiscernabilités.

La décidabilité du contrôle centralisé est démontrée grâce à l'opération de quotient $\mathcal{A}/P$ d'un automate étendu $\mathcal{A}$ par un processus P et aussi grâce à l'opération d'intersection $\mathcal{A}/P \wedge \mathcal{B}$ de deux automates étendus $\mathcal{A}/P$ et $\mathcal{B}$. Le problème de contrôle est transformé en problème de la satisfiabilité d'un automate $\mathcal{A}/P \wedge \mathcal{B}$ dont la décidabilité est prouvé grâce à une extension du théorème de simulation et de la théorie des jeux.

La difficulté majeure a été d'établir un théorème de simulation pour les automates étendus. Ce théorème a déjà été établi pour le cas des automates avec événements inobservables ("loop automata") dans [AVW03].

La taille de l'automate $\mathcal{A}/P\wedge\mathcal{B}$ (son nombre d'états), noté $|\mathcal{A}/P\wedge\mathcal{B}|$, est $|\mathcal{A}|\cdot|P|+|\mathcal{B}|$ où $|P|$ est aussi le nombre d'états de P. L'automate quotient $\mathcal{A}/P$ présenté dans ce chapitre reste nondéterministe si $\mathcal{A}$ est nondéterministe. Cependant, l'opération d'intersection des automates $\mathcal{A}/P$ et $\mathcal{B}$ ne conserve pas le nondéterminisme. En général, il existe alors un automate nondéterministe équivalent à $\mathcal{A}/P\wedge\mathcal{B}$ de taille $2^{\mathcal{O}(n\log n)}$ où $n=|\mathcal{A}|\cdot|P|+|\mathcal{B}|$. Dans [AVW03], les auteurs font remarquer cette complexité peut être réduite à $|P|\cdot 2^{\mathcal{O}(m\log m)}$ où $m=|\mathcal{A}|+|\mathcal{B}|$.

Cet algorithme de synthèse de contrôleur est implémenté par G. Point dans le logiciel "synthesis" (voir [P05]). Ce logiciel traite aussi le cas du contrôle décentralisé (voir sections 2.2 et 6.3) avec information totale.

Chapitre 4

Information partielle sur les comportements

4.1 Comportements inobservables et indiscernables

Nous avons introduit dans la partie 3.1 du chapitre 3 la notion de masque $M : A \to \Delta \cup \{\varepsilon\}$ qui permet à la fois de représenter les évènements inobservables et indiscernables.

On peut très simplement considérer que l'opération de masque ne s'effectue pas sur les évènements mais sur les comportements : $M : A^* \to \Delta \cup \{\varepsilon\}$. Ainsi si $u \in A^*$ et $M(u) = \varepsilon$ le contrôleur ne peut observer le comportement u : il doit se comporter de la même manière avant et après u. De même, si $u, v \in A^*$ et $M(u) = M(v)$ le contrôleur ne peut distinguer les comportements u et v. On parle alors de comportements inobservables et indiscernables.

Remarquons d'abord que dire que deux comportements $wa, wb \in A^*$ avec $a, b \in A$ sont indiscernables revient à considérer qu'après le comportement w les évènements a et b sont indiscernables. Dans ce cas, les automates étendus définis au chapitre 1 suffisent pour spécifier ce type de contraintes.

Mais on peut aussi considérer le cas où le contrôleur détecte les évènements $a, b \in A$ mais ne peut savoir dans quel ordre ils se sont produits, c'est à dire $M(ab) = M(ba)$ (il existe un déficit de synchronisme). En d'autres termes, le contrôleur ne reçoit que le "paquet" $\{a, b\}$. Le contrôleur peut cependant discerner a et b, c'est à dire $M(a) \neq M(b)$. En effet, si de plus, il ne reçoit que le "paquet" $\{a, c\}$ où $c \in A$, c'est à dire $M(ac) = M(ca)$, mais différencie les deux "paquets" $\{a, b\}$ et $\{a, c\}$, alors nécessairement $M(b) \neq M(c)$. Donc, nécessairement $M(a) \neq M(b)$ ou $M(a) \neq M(c)$. Il existe donc bien

des cas où le contrôle avec comportements indiscernables ne peut se réduire au cas des évènements indiscernables.

Avec notre formalisme, si $Q = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ est un contrôleur et pour tout $s \in S$,

- Si le comportement $u \in A^*$ est inobservable et $e(s, u)$ est défini, alors $e(s, u) = s$.
- Si les comportements $u, v \in A$ sont indiscernables et $e(s, u)$ et $e(s, v)$ sont définis, alors $e(s, u) = e(s, v)$.

Nous prouvons dans ce chapitre qu'une extension du μ-calcul à ce type de contraintes rend la satisfiabilité indécidable. La partie 4.2 est dédiée au problème de Post et à sa caractérisation. Il s'agit de décomposer le problème de Post en termes de propriétés des automates afin d'identifier celles qui ne sont pas exprimables par le μ-calcul. La partie 4.3 prouve ensuite l'indécidabilité de ces spécifications même si nous restreignons le μ-calcul aux automates à condition d'accessibilité (voir définition 1.2.2). Seul le cas des boucles de longueur 2 reste ouvert.

4.2 Le problème de POST

4.2.1 Définition

Définition 4.2.1 Soit A un alphabet fini et n un entier. On appelle *Problème de Post* le problème suivant :

Soit $U = \{u_p\}_{1 \leq p \leq n}$ et $V = \{v_p\}_{1 \leq p \leq n}$ deux familles finies de mots de A^*. Existe-t-il un suite finie d'indices $\{p_k\}_{1 \leq k \leq s}$ tel que :

$$u_{p_1} u_{p_2} \cdots u_{p_s} = v_{p_1} v_{p_2} \cdots v_{p_s}$$

On appelle $\{(u_p, v_p) : p = 1, \ldots n\}$ un système de Post sur l'alphabet A.

L'indécidabilité de ce problème est affiné dans [MaSe05] comme suit.

Théoreme 4.2.2 Le problème de Post est indécidable même si on fixe $n = 7$ et $A = \{0, 1\}$.

4.2.2 Caractérisation du problème de Post

Dans cette section, nous présentons à la proposition 4.2.5 une caractérisation du problème de Post très similaire à celle présentée dans [AVW03]. Il s'agit d'une équivalence entre l'existence d'une solution au problème de

Post et l'existence de mots appartenant à deux langages. Nous explicitons à la fin de cette section et à la section 4.2.3 suivante, la "part" de ces langages expressible par un automate simple. Seule une propriété de "commutativité" de comportements (la propriété 4.2.5 de la proposition 4.2.9) n'est pas expressible.

C'est cette dernière propriété qui nous permettra de conclure, dans les parties 4.3.1 et 4.3.3, à l'indécidabilité de la satisfiabilité des automates prenant en compte des contraintes d'observabilité et d'indiscernabilité sur les comportements.

Définition 4.2.3 Soit $m \geq 1$ un entier. Soit X et Y deux alphabets et $Y_m = \{y^m : y \in Y\}$. Soit $x \in X^*$ et $y \in Y_m^*$. On note $\mathit{Shuffle}_m(x, y)$ l'ensemble des mots w de $\{X \cup Y_m\}^*$ tel que $\pi_X(w) = x$ et $\pi_Y(w) = y$. On peut représenter le shuffle comme l'ensemble des chemins d'une grille (voir Figure 4.1). On note $\mathit{Shuffle}(x, y)$ au lieu de $\mathit{Shuffle}_1(x, y)$.

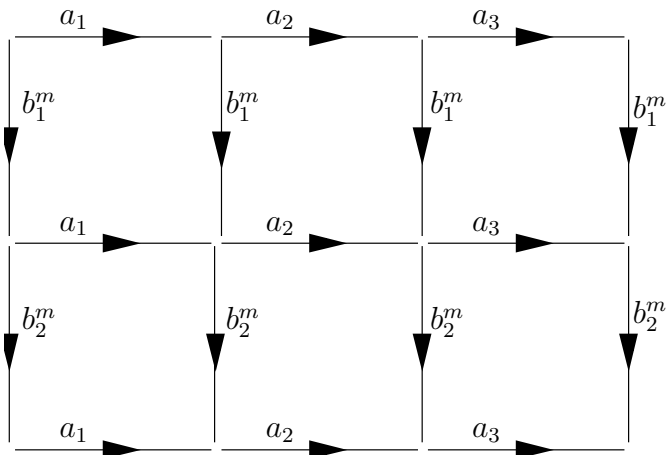


FIG. 4.1 – *Shuffle* de $x = a_1a_2a_3$ et $y = b_1^m b_2^m$

Définition 4.2.4 Soit $\{(u_p, v_p) : p = 1, \ldots n\}$ un système de Post sur A et B l'alphabet $\{\$_p\}_{1 \leq p \leq n}$. On note, pour $m > 0$, $L_1^m = \{\$_p^m u_p : p = 1, \ldots n\}^+$ et $L_2^m = \{\$_p^m v_p : i = 1, \ldots n\}^+$.

Proposition 4.2.5 Pour tout $m \geq 1$, le problème de Post admet une solution si et seulement si il existe $x \in A^*$ et $y \in B^*$ tel que :

$$\mathit{Shuffle}_m(x, y) \cap L_1^m \neq \emptyset \text{ et } \mathit{Shuffle}_m(x, y) \cap L_2^m \neq \emptyset$$

Preuve :
Soit $\{i_k\}_{1\leq k\leq p}$ une solution du problème de Post. Soit $x = u_{i_1}u_{i_2}\cdots u_{i_p} = v_{i_1}v_{i_2}\cdots v_{i_p}$ et $y = \$_{i_1}^m\$_{i_2}^m\cdots\$_{i_p}^m$. Alors $\$_{i_1}^m u_{i_1}\$_{i_1}^m u_{i_2}\cdots\$_{i_1}^m u_{i_p} \in L_1^m \cap Shuffle_m(x,y)$ et $\$_{i_1}^m v_{i_1}\$_{i_1}^m v_{i_2}\cdots\$_{i_1}^m v_{i_p} \in L_2^m \cap Shuffle_m(x,y)$.

Supposons maintenant qu'il existe $x \in A^*$ et $y \in B^*$ tel que $Shuffle_m(x,y) \cap L_1^m \neq \emptyset \neq Shuffle_m(x,y) \cap L_2^m$. Alors il existe $\$_{i_1}^m u_{i_1}\$_{i_2}^m u_{i_2}\cdots\$_{i_p}^m u_{i_p} \in L_1^m \cap Shuffle_m(x,y)$ et $\$_{j_1}^m v_{j_1}\$_{j_2}^m v_{j_2}\cdots\$_{j_q}^m v_{j_q} \in L_2^m \cap Shuffle_m(x,y)$. Donc, $p = q$ et pour tout $l = 1,2,\ldots,p$, on a $i_l = j_l$ car $\$_{i_1}^m\cdots\$_{i_p}^m = \$_{j_1}^m\cdots\$_{j_q}^m$. Donc $x = u_{i_1}u_{i_2}\cdots u_{i_p} = v_{i_1}v_{i_2}\cdots v_{i_p}$. □

Nous finissons cette section, en définissant des automates simples $\mathcal{A}^{\exists}(L_i^m)$ pour $i = 1,2$ et $m > 0$, qui vérifient le lemme 4.2.7 afin de prouver l'indécidabilité de problèmes grâce à la réduction du problème de Post (voir la proposition 4.2.8).

Définition 4.2.6 Notons pour $i = 1,2$ et pour tout $\$_p u_p \in L_i^m$ (voir définition 4.2.4) avec $p \in]n]$, $u_p = a_p^1 a_p^2 \cdots a_p^{j_p} \in A^{j_p}$. Soit Z un alphabet contenant $A \cup B$. Soit $\mathcal{A}^{\exists}(L_i^m) = \langle Z, Q, \{q_0, q_1\}, Q^{\forall}, q_0, \delta, \emptyset\rangle$ l'automate simple défini comme suit :

$$Q^{\forall} = \{q_{p,\$}^k, q_p^j : p = 1,\ldots,n, k = 1,\ldots,m \text{ et } j = 1,\ldots,j_p\} \cup \{q_\emptyset\}$$

- $\delta(q_0) = \{(\varepsilon, q_{p,\$}^1) : p = 1,\ldots,n\}$
- $\delta(q_{p,\$}^k) = \{\rightarrow_{\$_p}, (\$_p, q_{p,\$_p}^{k+1})\}$ pour $k = 1,\ldots,m-1$
- $\delta(q_{p,\$}^m) = \{\rightarrow_{\$_p}, (\$_p, q_p^1)\}$
- $\delta(q_p^j) = \{\rightarrow_{a_p^j}, (a_p^j, q_p^{j+1})\}$ pour $j = 1,\ldots,j_p - 1$
- $\delta(q_p^{j_p}) = \{\rightarrow_{a_p^{j_p}}, (a_p^{j_p}, q_1)\}$
- $\delta(q_1) = \{(\varepsilon, q_{p,\$}^1) : p = 1,\ldots,n\} \cup \{q_\emptyset\}$
- $\delta(q_\emptyset) = \{\nrightarrow_x : x \in A \cup B\}$

Lemme 4.2.7 Soit $P = \langle Z, S, s_0, e\rangle$ un processus avec $A \cup B \subseteq Z$. Alors $P \vDash \mathcal{A}^{\exists}(L_i^m)$ si et seulement si il existe $u \in L_i^m$ tel que $e(s_0,u)$ est défini et $D_P(e(s_0,u)) \cap (A \cup B)^* = \emptyset$.

Preuve : Si $P \vDash \mathcal{A}^{\exists}(L_i^m)$ alors à chaque stratégie gagnante, correspond un chemin u de P fini (car $Acc = \emptyset$) dans L_i^m atteignant la position $(q_\emptyset, e(s_0,u))$ du jeu $G(\mathcal{A}^{\exists}(L_i^m), P)$. La fonction de transition pour l'état $q_\emptyset$ nous assure que $D_P(e(s_0,u)) \cap (A \cup B)^* = \emptyset$.

Réciproquement, supposons qu'il existe $u \in L_i^m$ tel que $e(s_0,u)$ est défini et $D_P(e(s_0,u)) \cap (A \cup B)^* = \emptyset$ alors ont peut faire correspondre au chemin $u \in L_i^m$ une stratégie gagnante (la stratégie choisi le bon indice dans $]n]$). □

Proposition 4.2.8 Le problème de Post a une solution si et seulement si il existe un processus $P \vDash \mathcal{A}^{\exists}(L_1^m) \wedge \mathcal{A}^{\exists}(L_2^m)$ et il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = Pref(\mathit{Shuffle}_m(x, y))$

Preuve : Supposons que le problème de Post a une solution. D'après la proposition 4.2.5, il existe $x \in A^*$ et $y \in B^*$ tel que :

$$\mathit{Shuffle}_m(x, y) \cap L_1^m \neq \emptyset \text{ et } \mathit{Shuffle}_m(x, y) \cap L_2^m \neq \emptyset$$

On peut donc construire un processus $P = \langle Z, S, s_0, e\rangle$ à partir des mots x et y tel que $\mathcal{L}(P) = Pref(\mathit{Shuffle}_m(x, y))$. De plus, la proposition 4.2.5 nous assure qu'il existe deux chemins u et v de P, tous deux de longueur $|x| + |y|$, tel que $u \in L_1^m$ et $v \in L_2^m$. Comme $\mathcal{L}(P) = Pref(\mathit{Shuffle}_m(x, y))$, si $e(s_0, w)$ est défini et $|w| = |x| + |y|$, on a $D_P(e(s_0, w)) = \emptyset$. Donc, d'après le lemme 4.2.7, $P \vDash \mathcal{A}^{\exists}(L_1^m) \wedge \mathcal{A}^{\exists}(L_2^m)$.

Réciproquement, supposons maintenant qu'il existe un processus $P \vDash \mathcal{A}^{\exists}(L_1^m) \wedge \mathcal{A}^{\exists}(L_2^m)$ et il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = Pref(\mathit{Shuffle}_m(x, y))$. D'après le lemme 4.2.7, il existe $u \in L_1^m$ et $v \in L_2^m$ tel que $e(s_0, u)$ et $e(s_0, v)$ sont définis et $D_P(e(s_0, u)) \cap (A \cup B)^* = D_P(e(s_0, v)) \cap (A \cup B)^* = \emptyset$. Mais $u \in \mathit{Shuffle}_m(x, y)$ car, sinon, $D_P(e(s_0, u)) \cap (A \cup B)^* \neq \emptyset$. De même, $v \in \mathit{Shuffle}_m(x, y)$. Donc

$$\mathit{Shuffle}_m(x, y) \cap L_1^m \neq \emptyset \text{ et } \mathit{Shuffle}_m(x, y) \cap L_2^m \neq \emptyset$$

D'après la proposition 4.2.5, le problème de Post a une solution. □

4.2.3 Condition suffisante à l'existence du *Shuffle*

Afin que l'on puisse utiliser par la suite la caractérisation du problème de Post (proposition 4.2.8) présentée dans le paragraphe précédent, nous présentons ici des conditions suffisantes pour que le langage fini d'un processus soit le *Shuffle* de deux mots. Nous discutons aussi si ces conditions peuvent être ou non définissables par des automates simples.

Proposition 4.2.9 Soit A et B deux alphabets distincts et $B_m = \{b^m : b \in B\}$. Soit $P = \langle Z, S, s_0, e\rangle$ un processus avec $A \cup B \subset Z$. Il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = Pref(\mathit{Shuffle}_m(x, y))$ si les conditions suivantes sont vérifiées :

1.
$$\mathcal{L}(P) \cap (A \cup B)^* \subseteq (A \cup B_m)^* \tag{4.2.1}$$

2. Il existe $a \in A$ et $b \in B$, tel que :
$$D_P(s_0) \cap (A \cup B) = \{a, b\} \text{ ou } D_P(s_0) \cap (A \cup B) = \emptyset \tag{4.2.2}$$

3. Pour tout $s \in S$, il existe $a \in A$ et $b \in B$ tel que :

$$D_P(s) \cap (A \cup B) \subset \{a, b\} \tag{4.2.3}$$

4. Pour tout $s \in S$, si $D_P(s) \cap (A \cup B) = \{a, b\}$, alors

$$e(s, ab^m) \text{ et } e(s, b^m a) \text{ sont définis} \tag{4.2.4}$$

5. Pour tout $s \in S$, tel que $e(s, ab^m)$ et $e(s, b^m a)$ sont définis, alors :

$$e(s, ab^m) = e(s, b^m a) \tag{4.2.5}$$

6. Pour tout $s \in S$, si il existe $a \in A$ tel que $D_P(s) \cap (A \cup B) = \{a\}$, alors

$$D_P(e(s, a)) \cap (A \cup B) \subset A \tag{4.2.6}$$

7. Pour tout $s \in S$,
   - Si il existe $b, b' \in B$ tel que $e(s, b^m b'^m)$ est défini et $D_P(e(s, b^m)) \cap (A \cup B) = \{b'\}$, alors

$$D_P(e(s, b^m b'^m)) \cap (A \cup B) \subset B \tag{4.2.7}$$

   - Si il existe $a \in A$ et $b \in B$ tel que $e(s, a)$ est défini et $D_P(e(s, a)) \cap (A \cup B) = \{b'\}$, alors

$$D_P(e(s, ab'^m)) \cap (A \cup B) \subset B \tag{4.2.8}$$

Preuve : Soit X l'ensemble des états $s \in S$ tel qu'il existe $u \in (A \cup B_m)^*$ tel que $e(s_0, u) = s$. Pour $u, v \in (A \cup B)^*$, on note $u \equiv v$ si $\pi_A(u) = \pi_A(v)$ et $\pi_B(u) = \pi_B(v)$. Nous allons maintenant montrer les lemmes 4.2.10, 4.2.11 et 4.2.12 avant de revenir à la preuve principale.

Lemme 4.2.10 Soit $u, v \in (A \cup B_m)^*$ tel que $u \equiv v$. Alors pour tout $s \in X$, si $e(s, u)$ est défini, alors $e(s, v)$ est défini et égal à $e(s, u)$.

Preuve du lemme 4.2.10 : Supposons que le lemme est vrai pour tout $u, v \in (A \cup B_m)^*$ tel que $|u| = |v| < n$.
Soit $u, v \in (A \cup B_m)^*$ tel que $u \equiv v$ et $|u| = |v| = n$.
Supposons que $u = au'$ et $v = xv'$ avec $a \in A$ et $x \in A \cup B$. Si $x \in A$, alors, d'après la propriété (4.2.3), $x = a$. Donc d'après l'hypothèse d'induction, si $e(s, au') = e(e(s, a), u)$ est défini, alors $e(s, av') =$

$e(e(s,a),v)$ est défini et égale à $e(s,au')$. Supposons que $x = b \in B$. Si $e(s,b)$ n'est pas défini, d'après la propriété (4.2.6), $u \in A^*$ ce qui est contradictoire avec $u \equiv v$ puisque $\pi_B(u) = \pi_B(v) = \pi_B(bv') \neq \varepsilon$. Donc $e(s,b)$ est défini. D'après la propriété (4.2.1), on a $e(s,b^m)$ est défini et $v = b^m v''$.

De plus, on a :

$$\pi_A(v'') = \pi_A(b^m v'') = \pi_A(v) = \pi_A(u) = \pi_A(au') = a\pi_A(u')$$

$$\pi_B(u') = \pi_B(au') = \pi_B(u) = \pi_B(v) = \pi_B(b^m v'') = b^m \pi_B(v'')$$

Donc

$$u \equiv a\pi_A(u')\pi_B(u') \equiv a\pi_A(u')b^m\pi_B(v'') \equiv ab^m\pi_A(u')\pi_B(v'')$$

$$v \equiv b^m\pi_A(v'')\pi_B(v'') \equiv b^m\pi_A(u)\pi_B(v'') \equiv b^m a\pi_A(u')\pi_B(v'')$$

Donc, par induction, si $e(s,u) = e(s,au') = e(e(s,a),u')$ est défini, alors $e(s,ab^m\pi_A(u')\pi_B(v'')) = e(e(s,a),b^m\pi_A(u')\pi_B(v''))$ est défini et égale à $e(s,u)$. De même, d'après la propriété (4.2.4), $e(s,ab^m) = e(s,b^m a)$, donc si $e(s,ab^m\pi_A(u')\pi_B(v''))$ est défini, alors $e(s,b^m a\pi_A(u')\pi_B(v''))$ est défini et égale à $e(s,ab^m\pi_A(u')\pi_B(v''))$. De même, on montre que si $e(s,b^m a\pi_A(u')\pi_B(v''))$ est défini alors $e(s,v)$ est défini et égale à $e(s,b^m a\pi_A(u')\pi_B(v''))$. Donc $e(s,u) = e(s,v)$.

Supposons maintenant que $u = bu'$ et $v = xv'$ avec $b \in B$, $x \in A \cup B$ et $e(s,u)$ est défini. Si $x \in B$, alors, d'après la propriété (4.2.3), $x = b$. Donc d'après l'hypothèse d'induction, si $e(s,bu')$ est défini, alors $e(s,bv')$ est défini et égale à $e(s,bu')$. Supposons maintenant que $x \in A$. Si $e(s,a)$ n'est pas défini, d'après la propriété (4.2.2), $s \neq s_0$ et puisque $s \in X$, il existe $u \in \ \in (A \cup B_m)^+$ tel que $e(s_0,u) = s$. D'après les propriétés (4.2.7) et (4.2.8), $u \in B_m^+$. Ce qui est contradictoire avec $u \equiv v$. Donc $e(s,a)$ est défini. On démontre ensuite que $e(s,v)$ est défini et égale à $e(s,u)$ de façon similaire. □

Lemme 4.2.11 Pour tout $s \in X$ et $u \in A^*$ et $v \in B_m^*$, si $e(s,u)$ et $e(s,v)$ sont définis alors $e(s,uv)$ et $e(s,vu)$ sont définis et égaux.

Preuve du lemme 4.2.11 : La propriété est vrai si l'un des deux mots u ou v est vide. Supposons que la propriété est vrai pour tout $u \in A^*$ et $v \in B_m^*$ tel que $|u| + |v| < n$.

Soit $s \in X$, $u = au' \in A^+$ et $v = b^m v' \in B_m^+$ tel que $|u| + |v| = n$ et $e(s,u)$ et $e(s,v)$ sont définis. D'après les propriétés (4.2.4) et (4.2.5), $e(s, ab^m)$ et $e(s, b^m a)$ sont définis et égaux.

Comme $e(s, au')$ et $e(s, ab^m)$ sont définis, d'après l'hypothèse d'induction, $e(s, ab^m u')$ est défini. De même, comme $e(s, bv')$ et $e(s, b^m a)$ sont définis, d'après l'hypothèse d'induction, $e(s, b^m av')$ est défini.

Puisque $e(s, ab^m) = e(s, b^m a)$, d'après l'hypothèse d'induction et comme $e(s, ab^m u')$ et $e(s, b^m av')$ sont définis, $e(s, ab^m u'v')$ est défini.

Or, puisque $uv \equiv vu \equiv ab^m u'v'$, d'après le lemme 4.2.10, $e(s, uv)$ et $e(s, vu)$ sont définis et égaux. □

Lemme 4.2.12 Pour tout $s \in X$ et $u, v \in (A \cup B_m)^*$, si $e(s,u)$ et $e(s,v)$ sont définis alors il existe u' et v' tel que $e(s, uu')$ et $e(s, vv')$ sont définis et égaux.

Preuve du lemme 4.2.12 : La propriété est vrai si l'un des deux mots u ou v est vide. Supposons que la propriété est vrai pour tout $u, v \in (A \cup B_m)^*$ et tel que $|u| + |v| < n$. Soit $s \in X$ et $u, v \in (A \cup B_m)^+$ tel que $e(s,u)$ et $e(s,v)$ sont définis et $|u| + |v| = n$.

Si u et v ont un préfixe commun $w \in (A \cup B_m)^*$, alors, d'après l'hypothèse d'induction, la propriété est vrai.

Soit $x_0 = \pi_A(u)\pi_B(u) = \alpha_0 x_0'$ et $y_0 = \pi_A(v)\pi_B(v) = \beta_0 y_0'$. Soit $x_1 = \pi_B(u)\pi_A(u) = \alpha_1 x_1'$ et $y_1 = \pi_B(v)\pi_A(v) = \beta_1 y_1'$. D'après le lemme 4.2.10, $e(s, x_0)$, $e(s, x_1)$, $e(s, y_0)$ et $e(s, y_1)$ sont définis.

Si $\alpha_0 = \beta_0$, alors d'après l'hypothèse d'induction, il existe u' et v' tel que $e(s, x_0 u')$ et $e(s, y_0 v')$ sont définis et égaux. Or $x_0 u' \equiv uu'$ et $y_0 v' \equiv vv'$, donc, d'après le lemme 4.2.10, $e(s, uu')$ et $e(s, vv')$ sont définis et égaux. Pour $\alpha_1 = \beta_1$ la preuve est similaire.

Supposons maintenant que $\alpha_0 \neq \beta_0$ et $\alpha_1 \neq \beta_1$. D'après la propriété 4.2.3, soit $\alpha_0 \in A$ et $\beta_0 \in B$ ou soit $\alpha_0 \in B$ et $\beta_0 \in A$. Si $\alpha_0 \in A$ et $\beta_0 \in B$, alors $\pi_A(v) = \varepsilon$. Donc $y_1 = \pi_B(v) \neq \varepsilon$, donc $y_1 = \beta_0 y_0'$. Comme $\beta_0 \in B$, d'après la propriété 4.2.3, $\alpha_1 \in A$ et donc $\pi_B(u) = \varepsilon$. Donc $u \in A^+$ et $v \in B^+$. Si $\alpha_0 \in B$ et $\beta_0 \in A$ on démontre de façon similaire que $u \in B^+$ et $v \in A^+$.

Dans les deux cas, d'après le lemme 4.2.11, $e(s, uv)$ et $e(s, vu)$ sont définis et égaux. □

Revenons maintenant à la preuve de la proposition 4.2.9. Soit M l'ensemble des éléments maximaux (pour l'ordre préfixe) de l'ensemble $\mathcal{L}(P) \cap$

$(A \cup B)^*$. Soit $u \in \mathcal{L}(P) \cap (A \cup B)^*$. Si $v \equiv u$ alors, d'après le lemme 4.2.10, $v \in \mathcal{L}(P) \cap (A \cup B)^*$.

Soit $u \in M$ et $v \equiv u$. Si $v \notin M$ alors il existe $w \neq \varepsilon$ tel que $vw \in M$. Or $uw \equiv vw$ donc, d'après le lemme 4.2.10, $uw \in M$. Ce qui est contradictoire avec $u \in M$. Donc $v \in M$.

Soit $u, v \in M$. D'après le lemme 4.2.12, il existe u' et v' tels que $uu', vv' \in \mathcal{L}(P) \cap (A \cup B)^*$ et $uu' \equiv vv'$. Puisque u et v sont maximaux, $u' = v' = \varepsilon$ donc $u \equiv v$. □

Lemme 4.2.13 Les conditions de la proposition 4.2.9 sont définissables par des automates simples excepté la propriété (4.2.5).

Remarque 4.2.14 On peut aussi observer que les conditions de la proposition 4.2.9 à l'exception de la propriété (4.2.5), sont des conditions nécessaires.

Preuve du lemme 4.2.13 : La propriété (4.2.5) n'est pas définissable par un automate simple dont les modèles sont définis à bisimulation près (voir définition 0.4.4 et lemme 1.5.13) car il existe des modèles ne vérifiant pas la propriété (4.2.5) mais bisimilaires à ceux-ci. Nous décrivons maintenant des automates simples correspondant chacun à l'une des conditions de la proposition 4.2.9 excepté la propriété (4.2.5).

Soit $\mathcal{A}(4.2.1) = \langle Z, Q, \emptyset, Q^\forall, q_0, \delta, \emptyset \rangle$ tel que :
- $Q^\forall = \{q_0\} \cup \{q_b^i : i \in]m], b \in B\}$
- $\delta(q_0) = \{(a, q_0) : a \in A\} \cup \{(b, q_b^1) : b \in B\}$
- Pour tout $i \in]n-1]$, $\delta(q_b^i) = \{\rightarrow_b, (b, q_b^{i+1})\} \cup \{\nrightarrow_x : x \in A \cup B - \{a, b\}\}$
- $\delta(q_b^m) = \{\rightarrow_b, (b, q_0)\} \cup \{\nrightarrow_x : x \in A \cup B - \{a, b\}\}$

On a $P \vDash \mathcal{A}(4.2.1)$ si et seulement si P vérifie la propriété (4.2.1).

Soit $\mathcal{A}(4.2.2) = \langle Z, Q, q_0, Q^\forall, q_0, \delta, \emptyset \rangle$ tel que :
- $Q^\forall = \{q_{a,b} : a \in A, b \in B\}$
- $\delta(q_0) = \{(\varepsilon, q_{a,b}) : a \in A, b \in B\}$
- $\delta(q_{a,b}) = \{\rightarrow_a, \rightarrow_b\} \cup \{\nrightarrow_x : x \in A \cup B - \{a, b\}\}$

On a $P \vDash \mathcal{A}(4.2.2)$ si et seulement si P vérifie la propriété (4.2.2).

Soit $\mathcal{A}(4.2.3) = \langle Z, Q, q_0, Q^\forall, q_0, \delta, \emptyset \rangle$ tel que :
- $Q^\forall = \{q_{a,b} : a \in A, b \in B\} \cup \{q_x : x \in A \cup B\} \cup \{q_\emptyset\}$
- $\delta(q_0) = \{(\varepsilon, q_{a,b}), (\varepsilon, q_x) : a \in A, b \in B, x \in A \cup B\}$
- $\delta(q_{a,b}) = \{\rightarrow_a, (a, q_0), \rightarrow_b, (b, q_0)\} \cup \{\nrightarrow_x : x \in A \cup B - \{a, b\}\}$
- $\delta(q_x) = \{\rightarrow_x, (x, q_0)\} \cup \{\nrightarrow_x : x \in A \cup B - \{x\}\}$
- $\delta(q_\emptyset) = \{\nrightarrow_x : x \in A \cup B\}$

On a $P \vDash \mathcal{A}(4.2.3)$ si et seulement si P vérifie la propriété (4.2.3).

Soit $\mathcal{A}(4.2.4) = \langle Z, Q, q_0, Q^\forall, q_0, \delta, \emptyset\rangle$ tel que :
- $Q^\forall = \{q_{a,b} : a \in A, b \in B\} \cup \{q_X : X \subset A \cup B - \{\{a,b\} : a \in A, b \in B\}\} \cup \{q_{b,i}, q^a_{b,i} : a \in A, b \in B, i \in]m]\}$
- $\delta(q_0) = \{(\varepsilon, q_{a,b}) : a \in A, b \in B\} \cup \{(\varepsilon, q_X) : |X \cap A| \neq 1 \text{ et } |X \cap B| \neq 1\}$
- $\delta(q_X) = \{\rightarrow_x, (x, q_0) : x \in X\} \cup \{\nrightarrow_x : x \in A \cup B - X\}$
- $\delta(q_{a,b}) = \{\rightarrow_a, (a, q_0), (a, q_{b,1}), \rightarrow_b, (b, q_0), (b, q^a_{b,1})\} \cup \{\nrightarrow_x : x \in A \cup B - \{a,b\}\}$
- $\delta(q_{b,i}) = \{\rightarrow_b, (b, q_{b,i+1})\}$ pour $i = 1, \ldots, m-1$ et $\delta(q_{b,m}) = \{\rightarrow_b\}$
- $\delta(q^a_{b,i}) = \{\rightarrow_b, (b, q^a_{b,i+1})\}$ pour $i = 1, \ldots, m-1$ et $\delta(q^a_{b,m}) = \{\rightarrow_a\}$

On a $P \vDash \mathcal{A}(4.2.4)$ si et seulement si P vérifie la propriété (4.2.4).

Soit $\mathcal{A}(4.2.6) = \langle Z, Q, q_0, Q^\forall, q_0, \delta, \emptyset\rangle$ tel que :
- $Q^\forall = \{q_a : a \in A\} \cup \{q_X : X \subseteq A \cup B - \{a : a \in A\}\} \cup \{q_A\}$
- $\delta(q_0) = \{(\varepsilon, q_a) : a \in A\} \cup \{(\varepsilon, q_X) : X \subseteq A \cup B, |X \cap A| \neq 1\}$
- $\delta(q_X) = \{\rightarrow_x, (x, q_0) : x \in X\} \cup \{\nrightarrow_x : x \in A \cup B - X\}$
- $\delta(q_a) = \{\rightarrow_a, (a, q_A)\} \cup \{\nrightarrow_x : x \in A \cup B - \{a\}\}$
- $\delta(q_A) = \{(a, q_A) : a \in A\} \cup \{\nrightarrow_x : x \in B\}$

On a $P \vDash \mathcal{A}(4.2.6)$ si et seulement si P vérifie la propriété (4.2.6).

Soit $\mathcal{A}(4.2.7) = \langle Z, Q, \{q_1\}, Q^\forall, q_0, \delta, \emptyset\rangle$ tel que :
- $Q^\forall = \{q^1_{b,i}, q^2_{b,i} : b \in B, i \in]m+1]\} \cup \{q_0\} \cup \{q_X : X \subseteq A \cup B - \{a : a \in A\}\}$
- $\delta(q_0) = \{(b, q^1_{b,1}) : b \in B\} \cup \{(x, q_0) : x \in A \cup B\}$
- $\delta(q^1_{b,i}) = \{(b, q^1_{b,i+1})\}$ pour $i = 1, \ldots, m$
- $\delta(q^1_{b,m}) = \{(b, q_1)\}$
- $\delta(q_1) = \{(\varepsilon, q_X), (\varepsilon, q^2_{b,1})\}$
- $\delta(q_X) = \{\rightarrow_x : x \in X\} \cup \{\nrightarrow_x : x \in A \cup B - X\}$
- $\delta(q^2_{b,1}) = \{\rightarrow_b, (b, q^2_{b,2})\} \cup \{\nrightarrow_x : x \in A \cup B - \{b\}\}$
- $\delta(q^2_{b,i}) = \{(b, q^2_{b,i+1})\}$ pour $i = 2, \ldots, m$
- $\delta(q^2_{b,m+1}) = \{\nrightarrow_x : x \in A\}$

On a $P \vDash \mathcal{A}(4.2.7)$ si et seulement si P vérifie la propriété (4.2.7).

Soit $\mathcal{A}(4.2.8) = \langle Z, Q, \{q_1\}, Q^\forall, q_0, \delta, \emptyset\rangle$ tel que :
- $Q^\forall = \{q_a, q_{b,i} : a \in A, b \in B, i \in]m+1]\} \cup \{q_0\} \cup \{q_X : X \subseteq A \cup B - \{a : a \in A\}\}$
- $\delta(q_0) = \{(a, q_1) : a \in A\} \cup \{(x, q_0) : x \in A \cup B\}$
- $\delta(q_1) = \{(\varepsilon, q_X), (\varepsilon, q_{b,1})\}$
- $\delta(q_X) = \{\rightarrow_x : x \in X\} \cup \{\nrightarrow_x : x \in A \cup B - X\}$
- $\delta(q_{b,1}) = \{\rightarrow_b, (b, q_{b,2})\} \cup \{\nrightarrow_x : x \in A \cup B - \{b\}\}$

- $\delta(q_{b,i}) = \{(b, q_{b,i+1})\}$ pour $i = 2, \ldots, m$
- $\delta(q_{b,m+1}) = \{\nrightarrow_x : x \in A\}$

On a $P \vDash \mathcal{A}(4.2.8)$ si et seulement si P vérifie la propriété (4.2.8).

Définition 4.2.15 Soit $\mathit{PreShuffle}_m$ l'automate simple :

$$\mathcal{A}(4.2.1) \wedge \mathcal{A}(4.2.2) \wedge \mathcal{A}(4.2.3) \wedge \mathcal{A}(4.2.4) \wedge \mathcal{A}(4.2.6) \wedge \big(\mathcal{A}(4.2.7) \vee \mathcal{A}(4.2.8)\big)$$

Proposition 4.2.16 $P \vDash \mathit{PreShuffle}_m$ si et seulement si P vérifie les propriétés (4.2.1), (4.2.2), (4.2.3), (4.2.4), (4.2.6), (4.2.7) et (4.2.8) de la proposition 4.2.9.

Preuve : C'est une conséquence directe du lemme 4.2.13. □

4.3 Indécidabilité de la satisfiabilité

4.3.1 Automate avec comportements (k, p)-indiscernables

Définition 4.3.1 Un *automate avec comportements (k, p)-indiscernables* $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ est un automate simple dont on étend la fonction de transition δ comme suit :

$$\delta : Q \times 2^\Lambda \to \mathcal{P}\Big(\big((A \cup \{\varepsilon\}) \times Q\big) \cup \big(A^k \times A^p \times \{\downdownarrows, \not\downdownarrows\}\big)\Big)$$

Pour simplifier les écritures, on note, pour tout $u \in A^k$ et $v \in A^p$, $\downdownarrows_{u,v}$ et $\not\downdownarrows_{u,v}$ au lieu, respectivement, de $(u, v, \downdownarrows)$ et $(u, v, \not\downdownarrows)$.

La notion de calcul et d'acceptation de ces automates est aussi décrite par un jeu comme suit.

Définition 4.3.2 Soit $\mathcal{A} = \langle A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate avec comportements (k, p)-indiscernables et $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un processus. Le *jeu d'acceptation* $G(\mathcal{A}, P)$ est un graphe $\langle V_0, V_1, v_0, Edge, Acc_G\rangle$ défini comme pour un automate simple, auquel on ajoute de nouvelles arêtes :

- Depuis chaque sommet (q, s) et pour tout $\downdownarrows_{u,v} \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, u)$ et $e(s, v)$ sont définis mais pas égaux, et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\not\downdownarrows_{u,v} \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, u)$ et $e(s, v)$ sont définis et égaux, et vers $\top$ sinon.

Notation 4.3.3 On note $\mathit{AUTO}(A, \Lambda, (k, p), \downdownarrows)$ les automates avec comportements (k, p)-indiscernables.

4.3.2 Indécidabilité de la satisfiabilité

Théoreme 4.3.4 Si $k \geq 2$ ou $p \geq 2$, alors la satisfiabilité des automates avec comportements (k,p)-indiscernables est indécidable. Ce résultat reste vrai dans le cas où les automates sont à condition d'accessibilité (voir définition 1.2.2).

C'est une conséquence directe de la proposition 4.3.8 et du lemme 4.3.9

Notation 4.3.5 On note A, B et Z des alphabets tel que $A \cup B \subset Z$ et $z \in Z - (A \cup B)$.

Définition 4.3.6 Soit $\mathcal{A}_k^p = \langle Z, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate avec comportements (k,p)-indiscernables défini comme suit :

- $Acc = \emptyset$, $Q^\exists = \emptyset$ et $Q^\forall = \{q_0\} \cup \{q_z^i : i \in]k]\}$
- $\delta(q_0) = \{\rightarrow_z, \downdownarrows_{ab^{p-1},z^k}, \downdownarrows_{b^{p-1}a,z^k}\} \cup \{(x,q_0) : x \in A \cup B\} \cup \{(z,q_z^1)\}$
- $\delta(q_z^i) = \{(z,q_z^{i+1}), \rightarrow_z\}$ pour $i = 1,\ldots,k-1$
- $\delta(q_z^k) = \{\rightarrow_z\}$

Lemme 4.3.7 Soit $P = \langle Z, S, s_0, e\rangle$ un processus. Si $P \vDash \mathcal{A}_k^p$ alors pour tout $s \in S$ accessible, et pour tout $a \in A$ et et $b \in B$, si $e(s, ab^{p-1})$ et $e(s, b^{p-1}a)$ sont définis, alors ils sont égaux.

Preuve : Si $e(s_0,u) = s$ avec $u \in (A \cup B \cup \{z\})^*$, alors il existe un chemin depuis la position (q_0,s_0) vers (q_0,s) dans le jeu $G(\mathcal{A}_k^p, P)$. Puisque $\rightarrow_z \in \delta(q_0)$ et pour tout $i \in]k]$,$\rightarrow_z \in \delta(q_z^i)$, $e(s,z^k)$ est défini. Si $e(s,ab^{p-1})$ et $e(s,b^{p-1}a)$ sont définis, puisque $\downdownarrows_{ab^{p-1},z^k}, \downdownarrows_{b^{p-1}a,z^k} \in \delta(q_0)$, on a (voir figure 4.2) :

$$e(s,ab^{p-1}) = e(s,z^k) = e(s,b^{p-1}a)$$

□

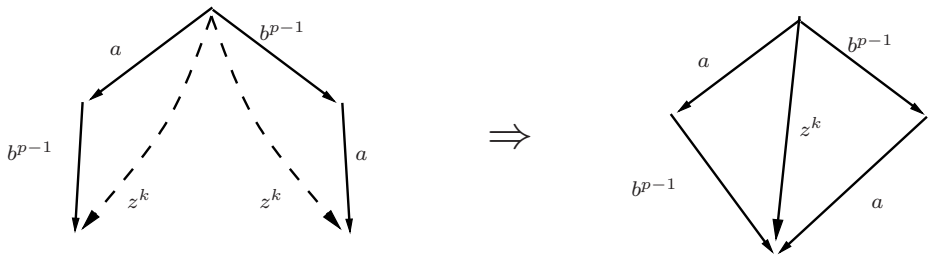


FIG. 4.2 –

Proposition 4.3.8 Le problème de Post a une solution si et seulement si $\mathcal{A}_k^p \wedge PreShuffle_{p-1} \wedge \mathcal{A}^\exists(L_1^{p-1}) \wedge \mathcal{A}^\exists(L_2^{p-1})$ a un modèle.

Preuve : Si $P \vDash \mathcal{A}_k^p$, d'après le lemme 4.3.16, alors P vérifie la condition (4.2.5) de la proposition 4.2.9 avec $m = p-1$. Si $P \vDash PreShuffle_{p-1}$ alors, d'après les propositions 4.2.16, P vérifie les autres conditions de la proposition 4.2.9 pour $m = p-1$. Donc il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = Shuffle_{p-1}(x, y)$. Puisque $P \vDash \mathcal{A}^\exists(L_1^{p-1}) \wedge \mathcal{A}^\exists(L_2^{p-1})$, d'après la proposition 4.2.8, le problème de Post a donc une solution.

Réciproquement, soit $\{i_k\}_{1 \leq k \leq l}$ une solution du problème de Post. Soit $x = u_{i_1} u_{i_2} \cdots u_{i_l} = v_{i_1} v_{i_2} \cdots v_{i_l}$ et $y = \$_{i_1}^{p-1} \$_{i_2}^{p-1} \cdots \$_{i_l}^{p-1}$. Alors on peut construire un processus $P = \langle A \cup B \cup \{z\}, S, s_0, e \rangle$ tel que :

- $\mathcal{L}(P) \cap (A \cup B)^* = Shuffle_{p-1}(x, y)$
- Pour tout $s \in S$, $a \in A$ et $b \in B$, si $e(s, ab^{p-1})$ et $e(s, b^{p-1}a)$ sont définis, alors $e(s, ab^{p-1}) = e(s, z^k) = e(s, b^{p-1}a)$.

On peut vérifier que $P \vDash \mathcal{A}_k^p \wedge PreShuffle_{p-1} \wedge \mathcal{A}^\exists(L_1^{p-1}) \wedge \mathcal{A}^\exists(L_2^{p-1})$. □

Lemme 4.3.9 L'automate $\mathcal{A}_k^p \wedge PreShuffle_{p-1} \wedge \mathcal{A}^\exists(L_1^{p-1}) \wedge \mathcal{A}^\exists(L_2^{p-1})$ est équivalent à un automate avec condition d'accessibilité.

Preuve : On peut vérifier qu'une combinaison booléenne d'automate, sans opération de complémentation, avec *Acc* vide est un automate avec *Acc* vide. □

4.3.3 Automate avec comportements k-inobservables

Définition 4.3.10 Un *automate avec comportements k-inobservables* $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ est un automate simple dont on étend la fonction de transition δ comme suit :

$$\delta : Q \times 2^\Lambda \to \mathcal{P}\Big(\big((A \cup \{\varepsilon\}) \times Q\big) \cup \big(A^k \times \{\circlearrowleft, \not\circlearrowleft\}\big) \Big)$$

Pour simplifier les écritures, on note, pour tout $u \in A^k$, $\circlearrowleft_u$ et $\not\circlearrowleft_u$ au lieu, respectivement, de $(u, \circlearrowleft)$ et $(u, \not\circlearrowleft)$.

La notion de calcul et d'acceptation de ces automates est aussi décrite par un jeu comme suit.

Définition 4.3.11 Soit $\mathcal{A} = \langle A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ un automate avec comportements k-inobservables et $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ un processus. Le *jeu d'acceptation* $G(\mathcal{A}, P)$ est un graphe $\langle V_0, V_1, v_0, Edge, Acc_G \rangle$ défini comme pour un automate simple, auquel on ajoute de nouvelles arêtes :

- Depuis chaque sommet (q, s) et pour tout $\circlearrowleft_u \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, u)$ est défini mais différent de s, et vers $\top$ sinon.
- Depuis chaque sommet (q, s) et pour tout $\not\circlearrowleft_u \in \delta(q, \lambda(s))$ il y a une arête dans $Edge$ vers $\bot$ si $e(s, u)$ est définit et égale à s, et vers $\top$ sinon.

Notation 4.3.12 On note $AUTO(k, \circlearrowleft)$ les automates avec comportements k-inobservables.

4.3.4 Indécidabilité de la satisfiabilité pour $k \geq 3$

Théoreme 4.3.13 Si $k \geq 3$ alors la satisfiabilité des automates avec comportements k-inobservables est indécidable. Ce résultat reste vrai si c'est un automate à condition d'accessibilité.

C'est une conséquence directe de la proposition 4.3.17 et du lemme 4.3.18.

Notation 4.3.14 On note A, B et Z des alphabets tel que $A \cup B \subset Z$ et $z \in Z - (A \cup B)$.

Définition 4.3.15 Soit $\mathcal{A}_k = \langle Z, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ un automate avec comportements k-inobservables défini comme suit :

- $Acc = \emptyset$, $Q^\exists = \emptyset$ et $Q^\forall = \{q_0\} \cup \{q_z^i : i \in]k]\}$
- $\delta(q_0) = \{\rightarrow_z, \circlearrowleft_{z^k}, \circlearrowleft_{abz^{k-2}}, \circlearrowleft_{baz^{k-2}}\} \cup \{(x, q_0) : x \in A \cup B \cup \{z\}\} \cup \{(z, q_z^1)\}$
- $\delta(q_z^i) = \{(z, q_z^{i+1}), \rightarrow_z\}$ pour $i = 1, \ldots, k-1$
- $\delta(q_z^k) = \{\rightarrow_z\}$

Lemme 4.3.16 Soit $P = \langle Z, S, s_0, e \rangle$ un processus. Si $P \vDash \mathcal{A}_k$ alors pour tout $s \in S$ accessible, et pour tout $a \in A$ et et $b \in B$, si $e(s, ab)$ et $e(s, ba)$ sont définis, alors ils sont égaux.

Preuve : Si $e(s_0, u) = s$ avec $u \in (A \cup B \cup \{z\})^*$, alors il existe un chemin depuis la position (q_0, s_0) vers (q_0, s) dans le jeu $G(\mathcal{A}_k, P)$. Puisque $\rightarrow_z \in \delta(q_0)$, et pour tout $i \in]k]$, $\rightarrow_z \in \delta(q_z^i)$, $e(s, z^k)$ est défini. Si $e(s, ab)$ et $e(s, ba)$ sont définis, alors $e(s, abz^k)$ et $e(s, baz^k)$ sont définis. Puisque $\circlearrowleft_{z^k} \in \delta(q_0)$, $e(s, ab) = e(s, abz^k)$ et $e(s, ba) = e(s, baz^k)$. Puisque $\circlearrowleft_{abz^{k-2}}, \circlearrowleft_{baz^{k-2}} \in \delta(q_0)$, on a (voir figure 4.3)

$$e(s, ab) = e(e(s, abz^{k-2}), z^2) = e(e(s, baz^{k-2}), z^2) = e(s, ba)$$

□

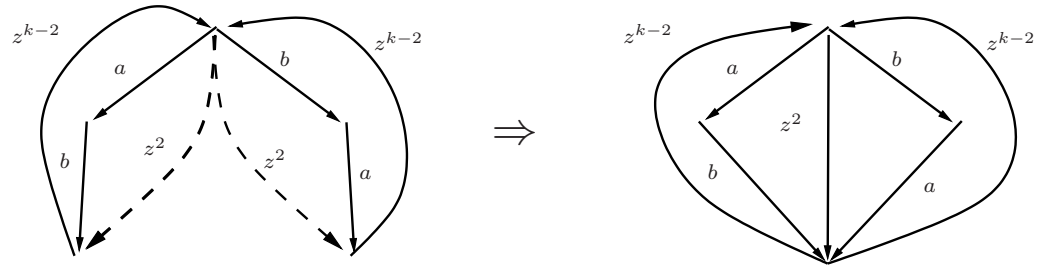


FIG. 4.3 –

Proposition 4.3.17 Le problème de Post a une solution si et seulement si $\mathcal{A}_k \wedge PreShuffle_1 \wedge \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$ a un modèle.

Preuve : Si $P \vDash \mathcal{A}_k$, d'après le lemme 4.3.16, alors P vérifie la condition (4.2.5) de la proposition 4.2.9. Si $P \vDash PreShuffle$ alors, d'après les propositions 4.2.16, P vérifie les autres conditions de la proposition 4.2.9. Donc il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = Shuffle_1(x, y)$. Puisque $P \vDash \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$, d'après la proposition 4.2.8, le problème de Post a donc une solution.

Réciproquement, soit $\{i_k\}_{1 \leq k \leq p}$ une solution du problème de Post. Soit $x = u_{i_1} u_{i_2} \cdots u_{i_p} = v_{i_1} v_{i_2} \cdots v_{i_p}$ et $y = \$_{i_1} \$_{i_2} \cdots \$_{i_p}$. Alors on peut construire un processus $P = \langle A \cup B \cup \{z\}, S, s_0, e \rangle$ tel que :

- $\mathcal{L}(P) \cap (A \cup B)^* = Shuffle_1(x, y)$
- Pour tout $s \in S$, $a \in A$ et $b \in B$, si $e(s_0, a)$ et $e(s_0, b)$ sont définis, alors $e(s_0, az) = s_0$ et $e(s_0, bz) = s_0$
- Pour tout $s \in S$, $a \in A$ et $b \in B$, si $e(s, ab)$ et $e(s, ba)$ sont définis, alors $e(s, ab) = e(s, z^2) = e(s, ba)$ et $e(s, abz^{k-2}) = s = e(s, baz^{k-2})$.

On peut vérifier que $P \vDash \mathcal{A}_k \wedge PreShuffle_1 \wedge \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$. □

Lemme 4.3.18 L'automate $\mathcal{A}_k \wedge PreShuffle_1 \wedge \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$ est équivalent à un automate avec condition d'accessibilité vide.

Preuve : On peut vérifier qu'une combinaison booléenne d'automates (sans complémentation) ayant des conditions d'acceptation *Acc* vides (i.e. des automates n'acceptant que des processus ayant des chemins finis) est équivalent à un automate avec condition d'acceptation vide. □

4.4 Conclusion

Les résultats présentés dans ce chapitre montrent que, de manière générale, il n'est pas possible d'étendre la notion d'automate simple aux cas de

comportements inobservables ou indiscernables sans que la satisfiabilité de ces automates soit indécidable. Grâce à la réduction du problème de Post présentée dans ce chapitre, on peut identifier une propriété qui suffit pour rendre indécidable toute extension du μ-calcul permettant de l'exprimer. Cette propriété est, de façon intuitive, une sorte de "commutativité des événements", du type ab et ba sont indiscernables. En d'autres termes, le contrôleur ne reçoit que le "paquet" $\{a, b\}$. On peut supposer que des cas d'asynchronisme "très simple" suffisent à rendre indécidable la satisfiabilité de spécifications.

Les cas où la satisfiabilité des spécifications est indécidable sont de deux sortes. Soit ce sont des automates qui peuvent spécifier que deux comportements $u, v \in A^+$ sont indiscernables avec $|u| \geq 2$ ou $|v| \geq 2$ (Bien sûr, si u et v sont indiscernables et $|u| = |v| = 1$ c'est le cas décidable des événements indiscernables). Ou soit, ce sont des automates qui peuvent spécifier qu'un comportement $u \in A^*$ est inobservable avec $|u| \geq 3$ (Si u est inobservable et $|u| = 1$ c'est le cas décidable des événements inobservables).

Dans les deux cas, l'indécidabilité de la satisfiabilité a été prouvée avec des automates avec conditions d'accessibilité (Acc est vide).

La satisfiabilité des automates avec comportements 2-inobservables reste un problème ouvert. On peut cependant remarquer que les automates avec événements indiscernables sont un cas particulier des automates avec évènements 2-inobservables. En effet, la propriété $\downdownarrows_{a,b}$ est équivalente à la conjonction des propriétés $\circlearrowleft_{az}, \circlearrowleft_{bz}$ et $\circlearrowleft_{zz}$, z étant un nouveau symbole.

Les automates avec comportements 2-inobservables sont, de plus, une généralisation des "two way automata" décris dans [V98]. En effet, pour tout événement $a \in A$, on peut définir un événement 'dual' $\bar{a}$ tel que on a toujours $\circlearrowleft_{a\bar{a}}$ et $\circlearrowleft_{\bar{a}a}$. Dans ce cas, après avoir effectué une transition a, on peut toujours revenir à l'état précédent en effectuant une transition par l'événement $\bar{a}$; l'événement $\bar{a}$ joue le rôle de "modalité arrière" pour a (backward modality).

Une autre question qui n'est pas étudiée ici est la restriction de l'expressivité des automates pour permettre la satisfiabilité de spécifications avec information partielle sur les comportements. Nous étudierons ce type de restrictions uniquement pour les problèmes de contrôle décentralisé avec événements indiscernables et inobservables.

Quatrième partie

Contrôle décentralisé avec information partielle

Chapitre 5

Indécidabilité du contrôle décentralisé

Le problème de contrôle décentralisé avec évènements indiscernables et inobservables (**DCPIO**) est défini comme suit :

Soit P un processus, $\mathcal{A}$ un automate simple et n automates étendus $\mathcal{B}_1, \ldots, \mathcal{B}_n$. Existe-t-il des contrôleurs $Q_1, Q_2, \ldots, Q_n$ tels que, pour tout i, $Q_i \vDash \mathcal{B}_i$ et $P \times Q_1 \times Q_2 \times \cdots \times Q_n \vDash \mathcal{A}$?

Dans ce chapitre nous démontrons que ce problème est indécidable dans chacun des trois cas : $\mathcal{B}_1, \mathcal{B}_2 \in AUTO(A, \circlearrowleft)$ ou $\mathcal{B}_1, \mathcal{B}_2 \in AUTO(A, \downdownarrows)$ ou $\mathcal{B}_1 \in AUTO(A, \circlearrowleft)$ et $\mathcal{B}_2 \in AUTO(A, \downdownarrows)$.

Ce problème est cependant décidable dans le cas où toutes les spécifications pour les contrôleurs sauf une (i.e. un seul $\mathcal{B}_i$) sont des automates simples (voir partie 6.3).

5.1 Contrôle décentralisé avec $\mathcal{B}_1, \mathcal{B}_2 \in AUTO(A, \circlearrowleft)$

Nous rappelons ici un résultat de [AVW03].

Théoreme 5.1.1 Le problème suivant est indécidable même si $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2$ sont à condition d'accessibilité (voir définition 1.2.2) :

Soit P un processus, $\mathcal{A}$ un automate simple et $\mathcal{B}_1, \mathcal{B}_2$ des automates avec évènements inobservables. Existe-t-il des contrôleurs Q_1, Q_2 tel que, pour

tout i, $Q_i \vDash \mathcal{B}_i$ et $P \times Q_1 \times Q_2 \vDash \mathcal{A}$?

Afin de prouver ce théorème, nous réduisons le problème de POST à ce problème grâce aux outils décrits aux sections 4.2.2 et 4.2.3 de la partie 4.2.

5.1.1 Spécifications $\mathcal{B}_1$ et $\mathcal{B}_2$

Définition 5.1.2 Soit A et B deux alphabets. On note $LOOP(B)$ l'automate à parité avec évènements inobservables et avec l'unique état universel q_0 de parité 1 avec la fonction de transition δ tel que :

$$\delta(q_0) = \{(a, q_0) : a \in A\} \cup \{\circlearrowleft_b : b \in B\}$$

On note $LOOP(A)$ l'automate identique à $LOOP(B)$ où le rôle des alphabets A et B sont échangés.

On peut alors aisément vérifier le lemme suivant :

Lemme 5.1.3 Soit $P = \langle A \cup B, \Lambda, S, s_0, e, \lambda \rangle$ un processus et $X \in \{A, B\}$. Alors $P \vDash LOOP(X)$ si et seulement si, pour tout $s \in S$, si $x \in D_P(s) \cap X$ alors $e(s, x) = s$.

5.1.2 Preuve du théorème 5.1.1

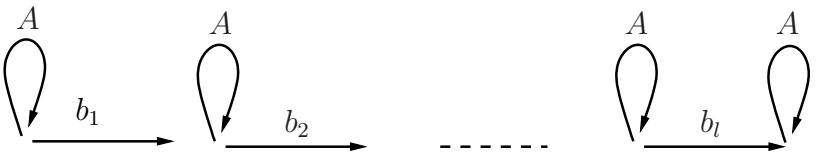
Proposition 5.1.4 Soit $\{(u_i, v_i) : i = 1, \ldots n\}$ un système de Post sur l'alphabet A (voir définition 4.2.1) et B l'alphabet $\{\$_i\}_{1 \leq i \leq n}$. Le problème de Post a une solution si et seulement si il existe $P \vDash LOOP(A)$ et $Q \vDash LOOP(B)$ tel que $P \times Q \vDash \mathit{PreShuffle}_1 \wedge \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$ (voir définitions 4.2.15, 4.2.4 et 4.2.6).

Preuve : Soit $P = \langle A \cup B, \Lambda, S, s_0, e, \lambda \rangle$ et $Q = \langle A \cup B, \Lambda, S', s'_0, e', \lambda' \rangle$ des processus tels que $P \vDash LOOP(A)$ et $Q \vDash LOOP(B)$. Si $e \times e'((s, s'), ab)$ et $e \times e'((s, s'), ba)$ sont définis, alors $e(s, b)$ et $e'(s', a)$ sont définis. Soit $s_1 = e(s, b)$ et $s'_1 = e'(s', a)$. Alors $e \times e'((s, s'), ab) = e \times e'((s, s'_1), b) = (s_1, s'_1)$ et $e \times e'((s, s'), ab) = e \times e'((s_1, s'), b) = (s_1, s'_1)$, donc $e \times e'((s, s'), ab)$ et $e \times e'((s, s'), ba)$ sont égaux.

D'après le lemme 4.3.16, alors P vérifie la condition (4.2.5) de la proposition 4.2.9 avec $m = 1$. Si $P \times Q \vDash \mathit{PreShuffle}_1$ alors, d'après les propositions 4.2.16, $P \times Q$ vérifie les autres conditions de la proposition 4.2.9 pour $m = 1$. Donc il existe $x \in A^*$ et $y \in B^*$ tel que $\mathcal{L}(P) \cap (A \cup B)^* = \mathit{Shuffle}_1(x, y)$. Puisque $P \vDash \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$, d'après la proposition 4.2.8, le problème de Post a donc une solution.

Réciproquement, soit $\{i_k\}_{1 \leq k \leq l}$ une solution du problème de Post. Soit $x = u_{i_1} u_{i_2} \cdots u_{i_l} = v_{i_1} v_{i_2} \cdots v_{i_l}$ et $y = \$_{i_1} \$_{i_2} \cdots \$_{i_l}$. Alors on peut construire un processus $P = \langle A \cup B \cup \{z\}, S, s_0, e \rangle$ (voir figure 5.1) tel que $e(s_0, w)$ est défini si et seulement si

- Pour tout ua préfixe de w tel que $a \in A$, on a $e(s_0, ua) = e(s_0, u)$.
- $\pi_B(w) = y$

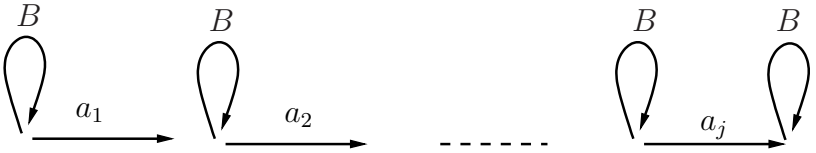


avec $y = b_1 b_2 \cdots b_l$

FIG. 5.1 – Processus P

On construit aussi un processus $Q = \langle A \cup B \cup \{z\}, S', s'_0, e' \rangle$ (voir figure 5.2) tel que $e'(s_0, w)$ est défini si et seulement si

- Pour tout ub préfixe de w tel que $b \in B$, on a $e(s_0, ub) = e(s_0, u)$.
- $\pi_A(w) = x$



avec $x = a_1 a_2 \cdots a_j$

FIG. 5.2 – Processus Q

On peut vérifier que $P \vDash LOOP(A)$, $Q \vDash LOOP(B)$ et :

$$P \times Q \vDash PreShuffle_1 \wedge \mathcal{A}^{\exists}(L_1^1) \wedge \mathcal{A}^{\exists}(L_2^1)$$

□

5.2 Contrôle décentralisé avec $\mathcal{B}_1 \in AUTO(A, \circlearrowleft)$ et $\mathcal{B}_2 \in AUTO(A, \downdownarrows)$

Théoreme 5.2.1 Le problème suivant est indécidable même si $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2$ sont des automates avec condition d'accessibilité (voir définition 1.2.2) :

Soit P un processus, $\mathcal{A}$ un automate simple et $\mathcal{B}_1$ un automate avec évènements inobservables et $\mathcal{B}_2$ un automate avec évènements indiscernables. Existe-t-il des contrôleurs Q_1, Q_2 tels que, pour tout i, $Q_i \vDash \mathcal{B}_i$ et $P \times Q_1 \times Q_2 \vDash \mathcal{A}$?

Nous reprenons ici aussi les travaux sur le problème de Post de la partie 4.2 et les adaptons afin de prouver le théorème 5.2.1.

Soit A et B deux alphabets finis. Soit e et $\#$ deux nouveaux symboles et C l'alphabet $A \cup B \cup \{e\} \cup \{\#\}$.

5.2.1 Spécification $\mathcal{B}_1$

Définition 5.2.2 On note $\mathcal{B}_1$ l'automate avec évènements inobservables $\langle C, Q, q_0, Q^\forall, q_0, \delta, \emptyset\rangle$ défini comme suit :

- $Q^\forall = \{q_a : a \in A\} \cup \{q_\#, q_\emptyset\}$
- $\delta(q_0) = \{(\varepsilon, q_a) : a \in A\} \cup \{(\varepsilon, q_\#)\}$
- Pour tout $a \in A$,
  $\delta(q_a) = \{\rightarrow_a, (a, q_0)\} \cup \{\rightarrow_b, \circlearrowleft_b : b \in B \cup \{e\}\} \cup \{\nrightarrow_x : x \notin B \cup \{e, a\}\}$
- $\delta(q_\#) = \{\rightarrow_\#, (\#, q_\emptyset)\} \cup \{\rightarrow_b, \circlearrowleft_b : b \in B \cup \{e\}\} \cup \{\nrightarrow_x : x \notin B \cup \{\#\}\}$
- $\delta(q_\emptyset) = \{\nrightarrow_x : x \in C\}$

On peut vérifier aisément le lemme suivant illustré par la figure 5.3.

Lemme 5.2.3 Soit $Q_1 = \langle C, S, s_0, e\rangle$ un processus. $Q_1 \vDash \mathcal{B}_1$ si et seulement si il existe $u = a_1 a_2 \dots a_m \in A^*$ tel que, en notant $u_i = a_1 a_2 \cdots a_i$, on a :

- $D_{Q_1}(s_0) = \{a_1\} \cup B \cup \{e\}$ et pour tout $b \in B$, $e(s_0, b) = s_0$
- Pour tout $i = 1, \dots, m-1$, $D_{Q_1}(e(s_0, u_i)) = \{a_{i+1}\} \cup B$
- Pour tout $i = 1, \dots, m$ et pour tout $b \in B$, $e(s_0, u_i b) = e(s_0, u_i)$
- $D_{Q_1}(e(s_0, u)) = \{\#\} \cup B$ et $D_{Q_1}(e(s_0, u\#)) = \emptyset$

FIG. 5.3 – Processus Q_1 avec $u = a_1 a_2 \dots a_m$

5.2.2 Spécification $\mathcal{B}_2$

Définition 5.2.4 On note $\mathcal{B}_2$ l'automate avec évènements indiscernables $\langle C, Q, q_0, Q^\forall, q_0, \delta, \emptyset\rangle$ défini comme suit :
- $Q^\forall = \{q_b : b \in B\} \cup \{q_e, q_\#, q_\emptyset\}$
- $\delta(q_0) = \{(\varepsilon, q_b) : b \in B\} \cup \{(\varepsilon, q_e)\} \cup \{(\varepsilon, q_\#)\}$
- Pour tout $c \in B \cup \{e\}$,
  $\delta(q_c) = \{\rightarrow_c, (c, q_0)\} \cup \{\downdownarrows_{a,c}, \rightarrow_a, (a, q_0) : a \in A\} \cup \{\nrightarrow_x : x \notin A \cup \{c\}\}$
- $\delta(q_\#) = \{\rightarrow_\#, (\#, q_\emptyset)\} \cup \{\nrightarrow_x : x \neq \#\}$
- $\delta(q_\emptyset) = \{\nrightarrow_x : x \in C\}$

On peut vérifier aisément le lemme suivant illustré par la figure 5.4.

Lemme 5.2.5 Soit $Q_2 = \langle C, S', s'_0, e'\rangle$ processus. $Q_2 \vDash \mathcal{B}_2$ si et seulement si il existe $v = y_1 y_2 \dots y_k \in (B \cup \{e\})^*$ tel que, en notant $v_j = y_1 y_2 \cdots y_j$, on a :
- $D_{Q_2}(s'_0) = \{y_1\} \cup A$ et pour tout $a \in A$, $e'(s'_0, a) = e'(s'_0, y_1)$
- Pour tout $j = 1, \dots, k-1$, $D_{Q_2}(e'(s'_0, v_j)) = \{y_{j+1}\} \cup A$
- Pour tout $j = 1, \dots, k-1$ et pour tout $a \in A$, $e'(s'_0, v_j a) = e'(s'_0, v_{j+1})$
- $D_{Q_2}(e'(s'_0, v)) = \{\#\}$ et $D_{Q_2}(e'(s'_0, v\#)) = \emptyset$

FIG. 5.4 – Processus Q_2 avec $v = y_1 y_2 \dots y_k \in (B \cup \{e\})^*$

5.2.3 Bons chemins dans $Q_1 \times Q_2$

Définition 5.2.6 Soit $P = \langle C, S, s_0, e\rangle$ un processus et $u \in C^*$ tel que $e(s_0, u)$ est défini. On dit que u est un *bon chemin* si pour tout préfixe $va \in C^*A$ de u, $D_P(e(s_0, v)) \cap B = \emptyset$.

Lemme 5.2.7 Soit $Q_1 \vDash \mathcal{B}_1$ et $Q_2 \vDash \mathcal{B}_2$. Alors il existe $u \in A^*$ et $v' \in (B \cup \{e\})^*$ tel que, en notant $v = \pi_B(v')$, on a :
- Si $w\# \in \mathcal{L}(Q_1 \times Q_2)$ est un bon chemin alors $\pi_A(w) = u$ et $\pi_B(w) = v$.
- Si $v' \in e^{|u|}(Be^{|u|})^*$ alors pour tout $z \in \mathit{Shuffle}(u, v)$ (voir définition 4.2.3), il existe un bon chemin $w\#$ dans $Q_1 \times Q_2$ tel que $\pi_{A\cup B}(w) = z$.

Preuve : Soit $Q_1 = \langle C, S, s_0, e\rangle$ et $Q_2 = \langle C, S', s'_0, e'\rangle$ des processus tel que $Q_1 \vDash \mathcal{B}_1$ et $Q_2 \vDash \mathcal{B}_2$.

Soit $u \in A^*$ le mot défini au lemme 5.2.3 (voir Figure 5.3). Soit $v' \in A^*$ le mot défini au lemme 5.2.5 (voir Figure 5.4). Soit $v = \pi_B(v')$.

Soit $w\#$ un mot de $Q_1 \times Q_2$. D'après le lemme 5.2.3, pour atteindre $\#$ on doit avoir $\pi_A(w) = u$.

De plus, de tout état $s \in S$ tel que $D_{Q_1}(s) \neq \emptyset$, on a $B \subset D_{Q_1}$. Donc $w\#$ est un bon chemin dans $Q_1 \times Q_2$ si c'est un bon chemin dans Q_2. On peut alors vérifier que nécessairement, $\pi_B(w) = \pi_B(v') = v$.

Supposons que $|u| = m$ et $v' \in e^m(Be^m)^*$. Soit $z \in \mathit{Shuffle}(u, v)$ tel que

$$z = v_0 a_1^1 \cdots a_{p_1}^1 v_{p_1} a_1^2 \cdots a_{p_2}^2 v_{p_2} \cdots a_1^r \cdots a_{p_r}^r v_{p_r}$$

avec

- $v_0 = b_1^0 \cdots b_{k_0}^0 \in B^*$
- Pour tout $i = 1, \ldots r-1$, $v_{p_i} \neq \varepsilon$ et $v_{p_i} = b_1^i \cdots b_{k_i}^i$
- $v_{p_r} = b_1^r \cdots b_{k_r}^r \in B^*$

Soit le mot

$$w = w_0 a_1^1 \cdots a_{p_1}^1 w_{p_1} a_1^2 \cdots a_{p_2}^2 w_{p_2} \cdots a_1^r \cdots a_{p_r}^r w_{p_r}$$

avec

- $w_0 = \varepsilon$ si $v_0 = \varepsilon$ et $w_0 = e^m b_1^0 e^m \cdots e^m b_{k_0}^0$ si $v_0 \neq \varepsilon$
- Pour $i = 1, \ldots, r$, $w_i = e^{m-p_i} b_1^i e^m b_2^i \cdots e^m b_{k_i}^i \in B^{k_i(m+1)+m-p_i}$

On peut vérifier que $w\#$ est un bon chemin dans $Q_1 \times Q_2$. □

Nous illustrons le lemme 5.2.7 par les figures 5.5 et 5.6. Les chemins en trait plein correspondent aux bons chemins. Dans l'exemple de la figure 5.5, il n'y a pas de bons chemins atteignant $\#$. Dans l'exemple de la figure 5.6, pour tout les mots de $\mathit{Shuffle}(a_1a_2, b_1b_2)$, il existe un chemin atteignant $\#$ dont la projection sur $A \cup B$ est ce mot.

5.2.4 Spécification $\mathcal{A}$: bons chemins dans $X_i^e\#$

Définition 5.2.8 Soit A un alphabet et $\{(u_i, v_i) : i = 1, \ldots n\}$ un système de Post sur A (voir définition 4.2.1). Soit B l'alphabet $\{\$_i\}_{1 \leq i \leq n}$. Notons $Y_1 = \{\$_i u_i : i = 1, \ldots n\}^*$ et $Y_2 = \{\$_i v_i : i = 1, \ldots n\}^*$ On note X_1^e l'ensemble des mots w tel qu'il existe $u \in Y_1$ et un entier k tel que $w = Shuffle(u, e^k)$ On définit X_2^e de façon similaire.

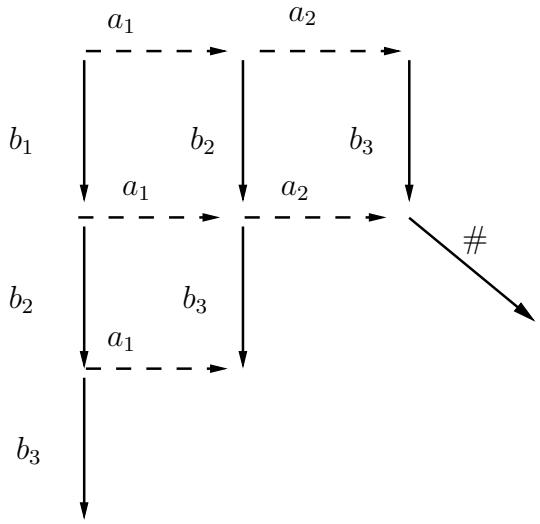


FIG. 5.5 – Chemins $Q_1 \times Q_2$ avec $u = a_1 a_2 \in A^*$ et $v = b_1 b_2 b_3 \in B^*$

Lemme 5.2.9 Soit $P = \langle C, S, s_0, e\rangle$ un processus. Alors, pour $i = 1, 2$, il existe un automate simple, noté $\mathcal{A}^{\exists}(X_i^e)$, tel que $P \vDash \mathcal{A}^{\exists}(X_i^e)$ si et seulement si il existe un bon chemin $u \in \mathcal{L}(P) \cap X_i^e\#$

On peut vérifier que les automates de la définition suivante 5.2.10 permettent la preuve du lemme 5.2.9.

Définition 5.2.10 Notons pour tout $p \in]n]$, $u_p = a_p^1 a_p^2 \cdots a_p^{j_p} \in A^{j_p}$. Soit $\mathcal{A}^{\exists}(X_1^e) = \langle C, Q, Q^{\exists}, Q^{\forall}, q_0, \delta, \emptyset\rangle$ l'automate simple défini comme suit :

$$Q^{\forall} = \{q_{p,\$}, q_p^j, F_p^j : p = 1, \ldots, n \text{ et } j = 1, \ldots, j_p\} \cup \{q_e, q_{\#}, q_{\emptyset}\}$$

$$Q^{\exists} = \{q_0\} \cup \{E_p^j : p = 1, \ldots, n \text{ et } j = 1, \ldots, j_p\}$$

- $\delta(q_0) = \{(\varepsilon, q_{p,\$}) : p = 1, \ldots, n\} \cup \{(\varepsilon, q_e), (\varepsilon, q_{\#})\}$
- $\delta(q_e) = \{\rightarrow_e, (e, q_0)\}$
- $\delta(q_{p,\$}) = \{\rightarrow_{\$_p}, (\$_p, E_p^1)\}$
- $\delta(E_p^j) = \{(\varepsilon, F_p^j), (\varepsilon, q_p^j)\}$
- $\delta(F_p^j) = \{\rightarrow_e, (e, E_p^j)\}$
- $\delta(q_p^j) = \{\rightarrow_{a_p^j}, (a_p^j, E_p^{j+1})\} \cup \{\nrightarrow_b : b \in B\}$ pour $j = 1, \ldots, j_p - 1$
- $\delta(q_p^{j_p}) = \{\rightarrow_{a_p^{j_p}}, (a_p^{j_p}, q_0)\} \cup \{\nrightarrow_b : b \in B\}$
- $\delta(q_{\#}) = \{\rightarrow_{\#}, (\#, q_{\emptyset})\} \cup \{\nrightarrow_x : x \neq \#\}$
- $\delta(q_{\emptyset}) = \{\nrightarrow_x : x \in C\}$

On définit de façon similaire l'automate simple $\mathcal{A}^{\exists}(X_2^e)$.

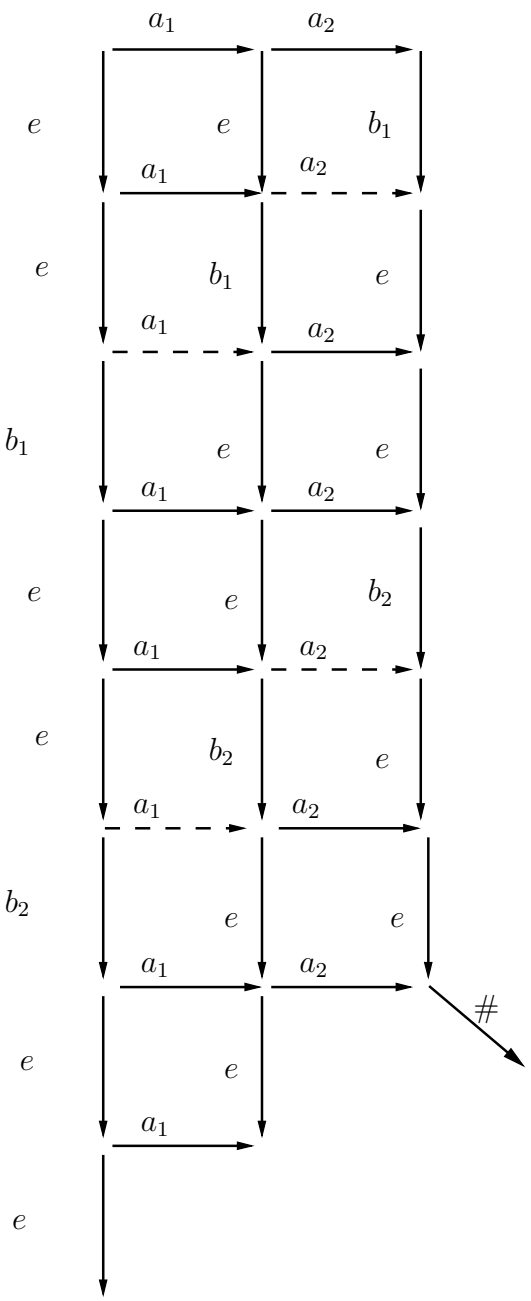


FIG. 5.6 – Chemins $Q_1 \times Q_2$ avec $u = a_1 a_2$ et $v = e^2 b_1 e^2 b_2 e^2$

5.2.5 Preuve du théorème 5.2.1

Théoreme 5.2.11 Le problème de Post a un solution si et seulement si il existe $Q_1 \vDash \mathcal{B}_1$ et $Q_2 \vDash \mathcal{B}_2$ tels que $Q_1 \times Q_2 \vDash \mathcal{A}^{\exists}(X_1^e) \wedge \mathcal{A}^{\exists}(X_1^e)$.

Preuve : Soit $Q_1 \vDash \mathcal{B}_1$ et $Q_2 \vDash \mathcal{B}_2$ tels que $Q_1 \times Q_2 \vDash \mathcal{A}^{\exists}(X_1^e) \wedge \mathcal{A}^{\exists}(X_1^e)$. D'après le lemme 5.2.9, il existe $x\# \in \mathcal{L}(Q_1 \times Q_2) \cap X_1^e\#$ et $y\# \in \mathcal{L}(Q_1 \times Q_2) \cap X_2^e\#$.

Alors il existe $\$_{i_1}u_{i_1}\$_{i_2}u_{i_2}\cdots\$_{i_p}u_{i_p} \in \{\$_iu_i : i = 1, \ldots n\}^*$ tel que :

$$\pi_{A\cup B}(x) = \$_{i_1}u_{i_1}\$_{i_2}u_{i_2}\cdots\$_{i_p}u_{i_p}$$

et $\$_{j_1}v_{j_1}\$_{j_2}v_{j_2}\cdots\$_{j_q}v_{j_q} \in \{\$_iv_i : i = 1, \ldots n\}^*$ tel que :

$$\pi_{A\cup B}(y) = \$_{j_1}v_{j_1}\$_{j_2}v_{j_2}\cdots\$_{j_q}v_{j_q}$$

De plus, d'après le lemme 5.2.7, $\pi_A(x) = \pi_A(y)$ et $\pi_B(x) = \pi_B(y)$. La deuxième égalité implique que $p = q$ et pour tout $l = 1, 2, \ldots, p$, on a $i_l = j_l$. Donc, puisque $\pi_A(x) = \pi_A(y)$, $u_{i_1}u_{i_2}\cdots u_{i_p} = v_{i_1}v_{i_2}\cdots v_{ji_q}$. Donc le problème de Post a une solution.

Soit $\{i_k\}_{1\leq k\leq p}$ une solution du problème de Post. Soit $u = u_{i_1}u_{i_2}\cdots u_{i_p} = v_{i_1}v_{i_2}\cdots v_{i_p} \in A^+$ avec $|u| = n$. On construit Q_1 avec le chemin u (voir Figure 5.3). On a bien $Q_1 \vDash \mathcal{B}_1$.

Soit $v' = e^n\$_{i_1}e^n\$_{i_2}e^n\cdots e^n\$_{i_p}e^n$ et $v = \pi_B(v)$. On construit Q_2 avec le chemin v' (voir Figure 5.4). On a bien $Q_2 \vDash \mathcal{B}_2$.

Soit $z = \$_{i_1}u_{i_1}\$_{i_2}u_{i_2}\cdots\$_{i_p}u_{i_p}$. Comme $v' \in e^{|u|}(Be^{|u|})^*$ et $z \in \mathit{Shuffle}(u, v)$, d'après le lemme 5.2.7, il existe un bon chemin $w\#$ dans $Q_1 \times Q_2$ tel que $\pi_{A\cup B}(w) = z$. De plus $z \in \{\$_iu_i : i = 1, \ldots n\}^*$. Donc il existe un bon chemin $w\#$ dans $X_1^e\#$.

Soit $t = \$_{i_1}v_{i_1}\$_{i_2}v_{i_2}\cdots\$_{i_p}v_{i_p}$. Comme $v' \in e^{|u|}(Be^{|u|})^*$ et $t \in \mathit{Shuffle}(u, v)$, d'après le lemme 5.2.7, il existe un bon chemin $w\#$ dans $Q_1 \times Q_2$ tel que $\pi_{A\cup B}(w) = t$. De plus $t \in \{\$_iv_i : i = 1, \ldots n\}^*$. Donc il existe un bon chemin $w\#$ dans $X_2^e\#$.

Donc $Q_1 \times Q_2 \vDash \mathcal{A}^{\exists}(X_1^e) \wedge \mathcal{A}^{\exists}(X_1^e)$. □

5.3 Contrôle décentralisé avec $\mathcal{B}_1, \mathcal{B}_2 \in AUTO(A, \downdownarrows)$

Théoreme 5.3.1 Le problème, noté **PCDI**, suivant est indécidable :

Soit P un processus, $\mathcal{A}$ un automate simple et $\mathcal{B}_1, \mathcal{B}_2$ des automates avec évènements indiscernables. Existe-t-il des contrôleurs Q_1, Q_2 tels que, pour tout i, $Q_i \vDash \mathcal{B}_i$ et $P \times Q_1 \times Q_2 \vDash \mathcal{A}$?

Nous n'allons pas ici réduire le problème de POST mais le problème de l'arrêt d'une machine de Turing. Commençons d'abord par formaliser la notion de machine de Turing.

5.3.1 Machine de Turing

Définition 5.3.2 Une machine de Turing M est un tuple $(K, k_0, T, \Gamma, \delta)$ tel que :

- K est un ensemble finit d'états.
- $k_0 \in K$ est l'état initial.
- $T \subset K$ est un ensemble d'état terminaux.
- $\Gamma = \Sigma \cup \{\sqcup\}$ où § est un alphabet fini et $\sqcup \notin \Sigma$ est le symbole "blanc".
- $\delta : K \times \Gamma \to K \times \Gamma \times \{\rightarrow, -, \leftarrow\}$ est la fonction de transition.

De plus, nous considérons que la machine de Turing effectue des transitions "bouclées" sur les états terminaux, c'est à dire :

$$\delta(t, \sigma) = (t, \sigma, -) \text{ si } t \in T \text{ et } \sigma \in \Gamma$$

Définition 5.3.3

- Une *configuration* est un mot dans $\Gamma^* K \Gamma^*$.
- Une *configuration initiale* est un mot de la forme $k_0 w$ où $w \in \Sigma^*$.
- Une *configuration terminale* est un mot dans $\Gamma^* T \Gamma^*$.
- On note $\rightarrow_M$ la relation suivante : pour tout $h, k \in K$, $u \in \Gamma^*$, $v \in \Gamma^+$ et $a, b, c \in \Gamma$,
  - $ukav \rightarrow_M ubhv$ si $\delta(k, a) = (h, b, \rightarrow)$
  - $ukav \rightarrow_M uhbv$ si $\delta(k, a) = (h, b, -)$
  - $uckav \rightarrow_M uhcbv$ si $\delta(k, a) = (h, b, \leftarrow)$
  - $uka \rightarrow_M ubh\sqcup$ si $\delta(k, a) = (h, b, \rightarrow)$
  - $kau \rightarrow_M h \sqcup bu$ si $\delta(k, a) = (h, b, \leftarrow)$

Dans la définition de la relation $\rightarrow_M$, seuls les deux derniers cas (i.e. $uka \rightarrow_M ubh\sqcup$ et $kau \rightarrow_M h\sqcup bu$) changent la longueur de la configuration en ajoutant un blanc $\sqcup$.

Définition 5.3.4 Soit C_0 et C_1 deux configurations. On note $C_0 \vdash_M C_1$ quand $C_0 \rightarrow_M C_1$ et que ces deux configurations ont la même longueur. On note $\vdash_M^*$ la clôture transitive de la relation $\vdash_M$.

Finissons cette partie en présentant un résultat classique très utile. En effet, c'est la propriété principale des machines de Turing pour notre preuve d'indécidabilité. Notons d'abord, pour $i = 1, 2$, $\pi_i : (\Gamma \cup K)^2 \to \Gamma \cup K$ la projection de la ième composante étendu à l'ensemble $((\Gamma \cup K)^2)^*$ par morphisme.

Lemme 5.3.5 L'ensemble des mots $u \in ((\Gamma \cup K)^2)^*$ où $\pi_1(u)$ et $\pi_2(u)$ sont des configurations tel que $\pi_1(u) \vdash_M \pi_2(u)$, est un ensemble rationnel.

Preuve : En effet, c'est le sous-ensemble $\Delta^* R \Delta^*$ où $\Delta = \{(x, x) : x \in \Gamma\}$ et $R = R_1 \cup R_2 \cup R_3$ avec :
- $R_1 = \{(k, b)(a, h) : h, k \in K, a, b \in \Gamma \text{ et } \delta(k, a) = (h, b, \to)\}$
- $R_2 = \{(k, h)(a, b) : h, k \in K, a, b \in \Gamma \text{ et } \delta(k, a) = (h, b, -)\}$
- $R_3 = \{(c, h)(k, c)(a, b) : h, k \in K, a, b, c \in \Gamma \text{ et } \delta(k, a) = (h, b, \leftarrow)\}$

□

5.3.2 Le problème $\mathbf{BTM}_\epsilon$

Définition 5.3.6 On note $\mathbf{BTM}_\epsilon$ le problème suivant :
Soit M une machine de Turing, existe-t-il un entier k tel que les configurations atteintes par M sur l'entrée vide sont toutes de longueur bornées par k ?

Proposition 5.3.7 Le problème $\mathbf{BTM}_\epsilon$ est indécidable.

Preuve : Le problème de l'arrêt d'une machine de Turing machine sur l'entrée vide (qui est indécidable) peut être réduit au problème $\mathbf{BTM}_\epsilon$. En effet, si la réponse au problème $\mathbf{BTM}_\epsilon$ est non, alors M n'atteint pas d'état terminaux. Sinon, si la réponse du problème $\mathbf{BTM}_\epsilon$ est oui, alors M atteint deux fois la même configuration. Puisque le nombre de configuration de longueur k est fini, on peut construire une machine de Turing M' qui accepte M si M atteint deux fois une même configuration terminale et rejette M sinon. □

Proposition 5.3.8 La réponse au problème $\mathbf{BTM}_\epsilon$ est oui si et seulement si il existe une configuration C et deux entiers m et p tel que $C_0 \vdash^+ C \vdash^* C$ et $C_0 = \sqcup^m k_0 \sqcup^p$.

Nous allons maintenant réduire le problème $\mathbf{BTM}_\epsilon$ au problème **PCDI**. Plus précisément, nous allons définir un plant $P(M)$, une spécification pour le système supervisé $\mathcal{A}(M)$ et deux spécifications pour les contrôleurs que nous notons $PATH_i(M) \wedge \Pi_i$ $(i = 1, 2)$, dépendant d'une machine de Turing M, tel que le théorème suivant soit vrai.

Théoreme 5.3.9 Le problème **PCDI** avec $P(M)$, $\mathcal{A}(M)$, $PATH_1(M) \wedge \Pi_1$ et $PATH_2(M) \wedge \Pi_2$ a une solution si et seulement si il existe une configuration C et deux entiers m et p tel que $C_0 \vdash^+ C \vdash^* C$ et $C_0 = \sqcup^m k_0 \sqcup^p$.

5.3.3 Spécifications pour les contrôleurs

Définition 5.3.10 On note B l'alphabet contenant $(\Gamma \cup K \cup \{\$, \#\})^2$ et les symboles spéciaux $\{\alpha, \beta\}$. On étend, pour $i = 1, 2$, les fonctions π_i à l'alphabet B de la façon suivante. π_i est la projection sur la $i^{\text{ème}}$ composante pour tout $b \in (\Gamma \cup K \cup \{\$, \#\})^2$ et, par convention, $\pi_i(\alpha) = \pi_i(\beta) = \gamma$ avec γ un nouveau symbole n'appartenant pas à $\Gamma \cup K \cup \{\$, \#\}$.

Nous définissons maintenant, pour $i = 1, 2$, l'automate simple $PATH_i(M)$ et les automates avec évènements indiscernables Π_i sur l'alphabet B. Intuitivement, notre but est que tous les chemins d'un modèle Q_i de l'automate $PATH_i(M) \wedge \Pi_i$ sont tel que leurs projections sont de la forme :

$$\gamma \$^n \# C_0^n \# C_1^n \cdots \# C_j^n \cdots$$

où C_j^n sont des configurations et $C_0^n = \sqcup^m k_0 \sqcup^p$ pour des entiers m et p. Remarquons que les configurations C_j^n n'ont pas nécessairement la même longueur. De plus, nous allons spécifié que ces chemins existent pour tout entier n et sont infinis. La Figure 5.7 permet de visualiser la forme des chemins d'un processus Q_i.

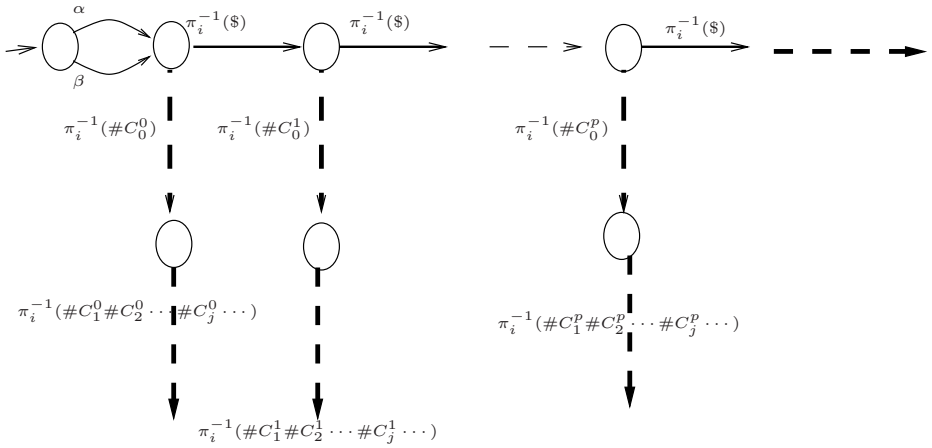


FIG. 5.7 – Contrôleur Q_i

Définition 5.3.11 Soit Π_i un automate avec un seul état universel q_0 et la fonction de transition :

$$\delta(q_0) = \{\downdownarrows_{b,b'} : \pi_i(b) = \pi_i(b')\} \cup \{(b, q_0) : b \in B\}$$

Lemme 5.3.12 $Q_i \vDash \Pi_i$ si et seulement si pour tout $u, v \in B^*$ tel que $\pi_i(u) = \pi_i(v)$, si $e^i(s_0^i, u)$ et $e^i(s_0^i, v)$ sont définis alors ils sont égaux.

Définition 5.3.13 Soit $PATH_i(M) = \langle B, Q, Q^\exists, Q^\forall q_0, \delta, r\rangle$ l'automate simple à parité défini comme suit (voir Figures 5.8 et 5.9) :

- $Q^\exists = \{q_1, q_2, q_3, q_4\}$
- $Q^\forall = \{q_0, q_\$, 1_\sqcup, q_{k_0}, 2_\sqcup, q_\#\} \cup \{p_\sigma, q_\sigma : \sigma \in \Gamma\} \cup \{q_k : k \in K\}$
- $r(q_0) = r(q_\$) = r(q_1) = r(q_{k_0}) = r(q_2) = r(q_3) = r(q_k) = r(q_4) = 0$
- $r(1_\sqcup) = r(2_\sqcup) = r(p_\sigma) = r(q_\sigma) = 1$ pour tout $\sigma \in \Gamma$
- $r(q_\#) = 2$
- $\delta(q_0) = \{(\alpha, q_\$), (\beta, q_\$), \rightarrow_\alpha, \rightarrow_\beta\} \cup \{\not\rightarrow_b : b \neq \alpha, \beta\}$
- $\delta(q_\$) = \{(x, q_\$), \rightarrow_x : \pi_i(x) = \$\} \cup \{(x, q_1), \rightarrow_x : \pi_i(x) = \#\} \cup \{\not\rightarrow_b : \pi_i(b) \notin \{\$, \#\}\}$
- $\delta(q_1) = \{(\epsilon, q_{k_0}), (\epsilon, 1_\sqcup)\}$
- $\delta(1_\sqcup) = \{(x, q_1) : \pi_i(x) = \sqcup\} \cup \{\not\rightarrow_b : \pi_i(b) \neq \sqcup\}$
- $\delta(q_{k_0}) = \{(x, q_2), \rightarrow_x : \pi_i(x) = k_0\} \cup \{\not\rightarrow_b : \pi_i(b) \neq k_0\}$
- $\delta(q_2) = \{(\epsilon, q_\#), (\epsilon, 2_\sqcup)\}$
- $\delta(2_\sqcup) = \{(x, q_2) : \pi_i(x) = \sqcup\} \cup \{\not\rightarrow_b : \pi_i(b) \neq \sqcup\}$
- $\delta(q_\#) = \{(x, q_3), \rightarrow_x : \pi_i(x) = \#\} \cup \{\not\rightarrow_b : \pi_i(b) \neq \#\}$
- $\delta(q_3) = \{(\epsilon, p_\sigma) : \sigma \in \Gamma\} \cup \{(\epsilon, q_k) : k \in K\}$
- $\delta(p_\sigma) = \{(x, q_3), \rightarrow_x : \pi_i(x) = \sigma\} \cup \{\not\rightarrow_b : \pi_i(b) \neq \sigma\}$
- $\delta(q_k) = \{(x, q_4), \rightarrow_x : \pi_i(x) = k\} \cup \{\not\rightarrow_b : \pi_i(b) \neq k\}$
- $\delta(q_4) = \{(\epsilon, q_\sigma) : \sigma \in \Gamma\} \cup \{(\epsilon, q_\#)\}$
- $\delta(q_\sigma) = \{(x, q_4), \rightarrow_x : \pi_i(x) = \sigma\} \cup \{\not\rightarrow_b : \pi_i(b) \neq \sigma\}$

Le lemme suivant caractérise les chemins d'un processus Q_i (voir Figure 5.7).

Lemme 5.3.14 Pour $i = 1, 2$, $Q_i \vDash PATH_i(M) \wedge \Pi_i$ si et seulement si

- Depuis tout état de Q_i il existe une transition.
- Il existe des fonctions $f_i : \mathbb{N} \rightarrow (\Sigma \cup K \cup \{\$, \#\})^\omega$, $r_i : \mathbb{N} \rightarrow \mathbb{N}$ et $g_i : \mathbb{N} \rightarrow \mathbb{N}$ tel que :
  $w \in B^\omega$ est un chemin infini dans Q_i si et seulement si, soit $\pi_i(w) = \gamma\$^\omega$ ou soit il existe $n \geq 0$ tel que $\pi_i(w) = \gamma\$^n\# \sqcup^{r_i(n)} k_0 \sqcup^{g_i(n)} f_i(n)$ avec $f_i(n) = \#C_1^n\# \cdots C_j^n \cdots$ où les C_j^n sont des configurations.
- Si $u, v \in B^\omega$ sont des chemins dans Q_i avec $\pi_i(u) = \pi_i(v)$ alors $e^i(s_0^i, u) = e^i(s_0^i, v)$.

Remarquons que Q_i est complètement défini par les fonctions f_i, k_i et g_i.

Preuve : Nous allons prouver que $Q_i \vDash PATH_i(M) \wedge \Pi_i$ implique les trois conditions du lemme 5.3.14. Observons que la dernière condition est une conséquence directe du lemme 5.3.12.

On peut vérifier que les transitions de l'automate $PATH_i(M)$ sont identiques pour tout $b, b' \in B$ tel que $\pi_i(b) = \pi_i(b')$. C'est pourquoi, par la suite, nous identifierons les chemins dans Q_i avec leurs projections par π_i.

Observons d'abord que $\gamma\$^n\#$ est un chemin dans Q_i. De plus, si u est un chemin dans Q_i alors soit $u = \gamma\$^\omega$ ou soit il existe un entier n tel que $\gamma\$^n\#$ est un préfixe de u. On peut le prouver avec la figure suivante 5.8 qui représente les premières transitions de l'automate $PATH_i(M)$.

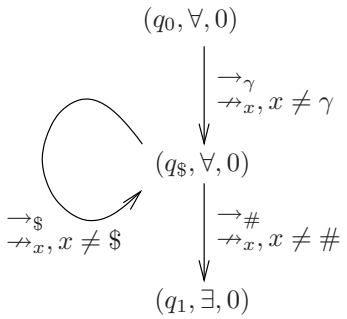


FIG. 5.8 – Chemins $\gamma\$^n\#$ et $\gamma\$^\omega$

La position $(q_0, \forall, 0)$ représente l'état de l'automate $PATH_i(M)$ (ici q_0), la nature de l'état (ici q_0 est un état universel : $q_0 \in Q^\forall$) et la parité de l'état (ici $r(q_0) = 0$). Les autres positions sont définis de façon similaires. Depuis un état universel, les symboles de la forme $\rightarrow_x$ exprime le fait que le processus doit avoir une x-transition et $\not\rightarrow_x$ exprime le fait que l'automate interdit les x-transition.

L'état q_0 est universel. Ainsi, le processus Q_i doit avoir une unique transition par γ depuis son état initial. Après la γ-transition, Q_i doit satisfaire les contraintes de l'état universel $q_\$$. Donc, Q_i doit avoir exactement une $-transition et une #-transition. De plus, après une $-transition, Q_i doit satisfaire les contraintes de l'état universel $q_\$$. Donc Q_i doit avoir exactement tous les chemins de la forme $\gamma\$^n\#$ avec $n \geq 0$ et le chemin $\gamma\$^\omega$.

Lemme 5.3.15 Pour tout entier $n \geq 0$, il existe un chemin infini unique dans Q_i de la forme $\gamma\$^n\# \sqcup^m k_0 \sqcup^p \#C_1^n\#C_2^n\#\cdots$ où les C_j^n sont des configurations.

On peut prouver ce lemme à l'aide de le figure 5.9 qui représente les transitions de l'automate $PATH_i(M)$ après le chemin $\gamma\$^n\#$. Comme nous pouvons le voir sur la figure 5.8, après le chemin $\gamma\$^n\#$, l'automate $PATH_i(M)$ est

dans l'état existentiel q_1. Sur la figure 5.9, les transitions de l'automate, depuis un état existentiel , étiqueté par ϵ, expriment les choix nondéterministe de l'automate.

On peut observer sur la figure 5.9 que, dans chaque état universel (i.e. $1_\sqcup$, q_{k_0}, $2_\sqcup$, $q_\#$, p_σ, q_k et q_σ), l'automate force le processus à avoir *exactement une transition*. Ainsi, pour tout n, il y a un unique chemin avec le préfixe $\gamma\$^n\#$ et, de plus, il doit être infini.

A cause des parité, l'automate ne peut rester dans l'une des quatre boucles, appelées *Loop*. Ainsi il visite l'état $q_\#$ infiniment souvent. De plus, avant d'atteindre l'état $q_\#$ pour la première fois, il lit un unique mot fini dans $\sqcup^* k_0 \sqcup^*$. Finalement, entre deux états $q_\#$ l'automate doit lire la lettre $\#$ suivit par un mot dans $\Gamma^* K \Gamma^*$, c'est à dire une configuration. □

5.3.4 Plant et spécification du système supervisé

Nous définissons maintenant le plant $P(M)$ avec la figure 5.10. Le lemme 5.3.5 nous assure que le plant $P(M)$ est bien défini, particulièrement le chemin depuis l'état 5 vers l'état 6. Définissons la spécification $\mathcal{A}$ pour le système supervisé $P \times Q_1 \times Q_2$.

Définition 5.3.16 Soit $\mathcal{A} = \langle B, Q, Q^\exists, Q^\forall, q_0, \delta, r\rangle$ un automate à parité avec $Q^\exists = \{q_0\}$, $Q^\forall = \{q_\$\} \cup \{q_\#\} \cup \{q_X : X \subseteq B, X \neq \emptyset\}$ et

- $r(q_0) = 0$, $r(q_X) = 1$ pour tout $X \subseteq B$ et $r(q_\$) = r(q_\#) = 2$
- $\delta(q_0) = \{(\epsilon, q_\#)\} \cup \{(\epsilon, q_\$)\} \cup \{(\epsilon, q_X) : X \subseteq B, X \neq \emptyset\}$
- $\delta(q_\$) = \{((\$, \$), q_0), \rightarrow_{(\$,\$)}\} \cup \{\nrightarrow_x : x \neq (\$, \$)\}$
- $\delta(q_\#) = \{((\#, \#), q_0), \rightarrow_{(\#,\#)}\} \cup \{\nrightarrow_x : x \neq (\#, \#)\}$
- $\delta(q_X) = \{(x, q_0), \rightarrow_x : x \in X\} \cup \{\nrightarrow_x : x \notin X\}$

Proposition 5.3.17 $P \vDash \mathcal{A}$ si et seulement si les chemins dans P sont préfixes de chemins infinis contenant une infinité de lettres $(\#, \#)$ ou une infinité de lettres $(\$, \$)$.

Preuve : Dans le jeu $G(\mathcal{A}, P)$, depuis toute position universelle, il doit y avoir une transition donc tous les chemins de P doivent être préfixes de chemins infinis. De plus, si il existe dans P un chemin avec une infinité de transitions par a, alors la partie associé dans $G(\mathcal{A}, P)$ doit passer infiniment par les positions $q_\$$ ou $q_\#$ pour être de parité paire. Donc ce chemin doit contenir une infinité de $(\#, \#)$ ou de $(\$, \$)$. □

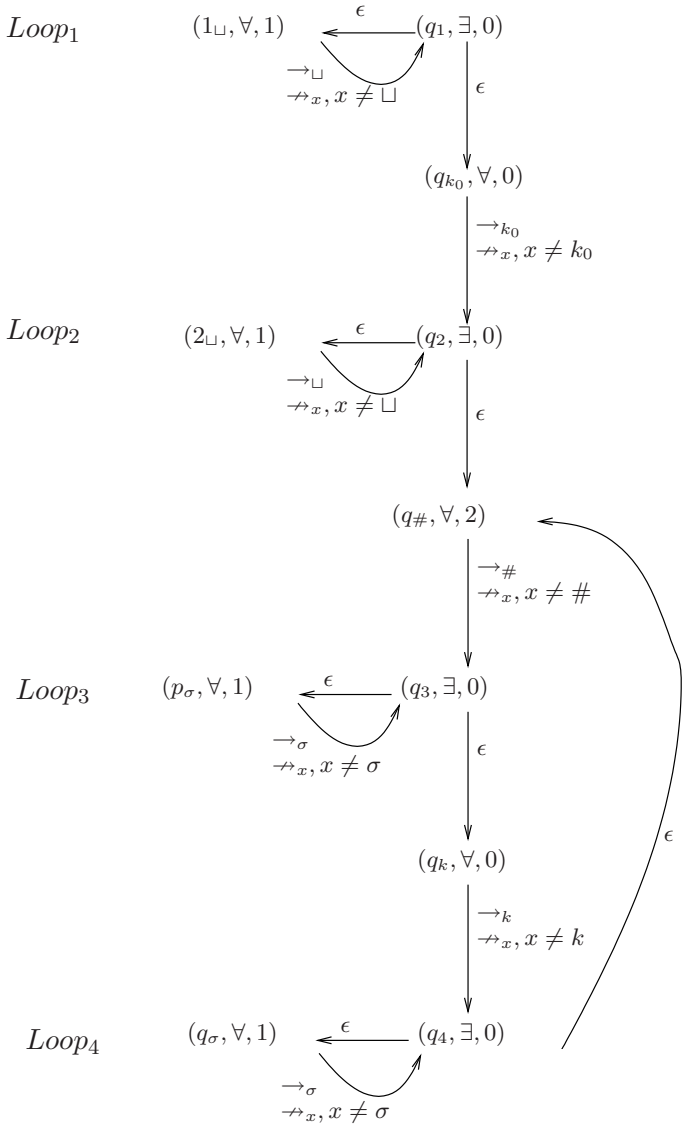


FIG. 5.9 – Chemins après $\gamma\$^n\#$

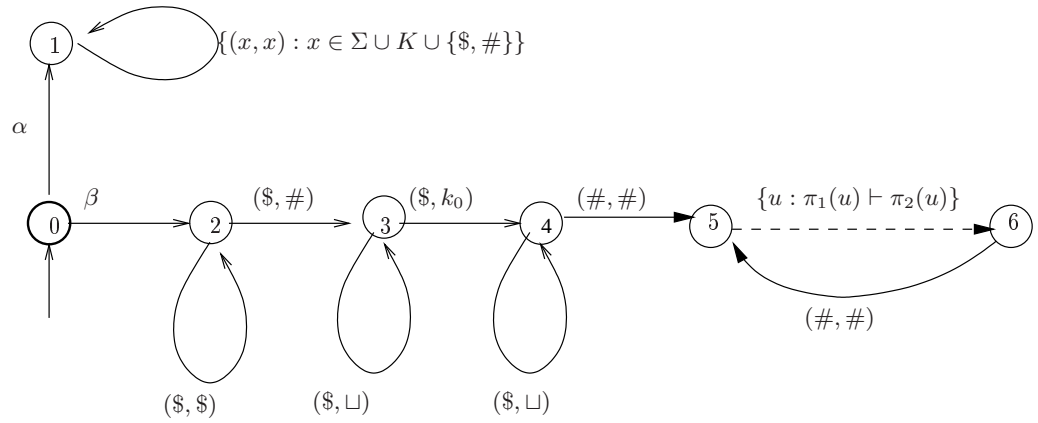


FIG. 5.10 – Processus $P(M)$

5.3.5 Indécidabilité : preuve du théorème 5.3.1

Théoreme 5.3.18 Soit M une machine de Turing. Supposons qu'il existe deux processus Q_1 et Q_2 tel que, pour $i = 1, 2$, $Q_i \vDash PATH_i(M) \wedge \Pi_i$ et $P(M) \times Q_1 \times Q_2 \vDash \mathcal{A}$. Alors il existe $C_0 = \sqcup^m k_0 \sqcup^n$ pour des entiers $m, n \geq 0$ et une configuration C tel que $C_0 \vdash^* C \vdash^+ C$.

Preuve :. Soit, pour $i = 1, 2$, les fonctions f_i, r_i et g_i, qui caractérisent complètement les chemins dans le processus Q_i (voir Lemme 5.3.14). Intuitivement, dans un premier temps, nous prouvons que les chemins avec le préfixe α dans $P(M)$, qui sont composées de lettres avec des composantes identiques, forcent les contrôleurs Q_i à être caractérisés par les même fonctions (i.e. $f_1 = f_2$, $g_1 = g_2$ et $r_1 = r_2$).

Soit $n \geq 0$ un entier. Notons que $u = \alpha(\$, \$)^n(\#, \#)$ est un chemin dans $P(M) \times Q_1 \times Q_2$. D'après le lemme 5.3.14, w est un chemin infini dans Q_i avec le préfixe u si et seulement si $\pi_i(w) = \gamma \$^n \# \sqcup^{r_i(n)} k_0 \sqcup^{g_i(n)} f_i(n)$ avec $f_i(n) = \#C_1^n \# \cdots C_j^n \cdots$ où les C_j^n sont des configurations. De plus, w est un chemin infini dans $P(M)$ avec le préfixe u si et seulement si $\pi_1(w) = \pi_2(w)$ (à cause du préfixe α). Mais, la spécification $\mathcal{A}$ implique que tous les chemins finis dans $P(M) \times Q_1 \times Q_2$ sont préfixes de chemins infinis. Ainsi $f_1(n) = f_2(n)$, $g_1(n) = g_2(n)$ et $r_1(n) = r_2(n)$. Puisque n est arbitraire, ces propriétés sont vrais pour tout $n \geq 0$. Donc $f_1 = f_2$, $g_1 = g_2$ et $r_1 = r_2$.

Soit $n \geq 0$ un entier. Observons d'abord que $v = \beta(\$, \$)^n(\$, \#)$ est un chemin dans $P(M) \times Q_1 \times Q_2$. La spécification $\mathcal{A}$ implique qu'il existe un chemin infini w dans $P(M) \times Q_1 \times Q_2$ avec le préfixe v. De plus, w doit contenir une infinité de $(\#, \#)$ (En effet, le chemin dans $P(M)$ avec le préfixe v ne contient

pas d'autres $(\$, \$)$). D'après la definition de $P(M)$, il existe deux entiers $p, q \geq 0$ tel que $w = v \cdot (\$, \sqcup)^p(\$, k_0)(\$, \sqcup)^q(\#, \#)(C_1, D_1)(\#, \#)(C_2, D_2)\cdots$ où C_j et D_j sont des configurations et $C_j \vdash D_j$ pour tout $j \geq 1$. En d'autres termes :

$$\pi_1(w) = \gamma\$^{n+1+p+q}\#C_1\#C_2\cdots$$
$$\pi_2(w) = \gamma\$^n\#D_0\#D_1\#D_2\cdots$$

tel que $C_i \vdash D_i$ pour tout $i \geq 1$ et $D_0 = \sqcup^p k_0 \sqcup^q$. Le chemin infini w correspond à un chemin dans Q_1 avec la projection sur la première composante défini par $f_1(n+p+1+q)$, $r_1(n+p+1+q)$ et $g_1(n+p+1+q)$, c'est à dire :

$$\pi_1(w) = \gamma\$^{n+p+1+q}\# \sqcup^{r_1(n+1+p+q)} k_0 \sqcup^{g_1(n+1+p+q)} \#f_1(n+1+p+q)$$

De plus, le chemin infini w correspond à un chemin dans Q_2 avec la projection sur la deuxième composante défini par $f_2(n)$, $r_2(n)$ et $g_2(n)$, c'est à dire :

$$\pi_2(w) = \gamma\$^n\# \sqcup^{r_2(n)} k_0 \sqcup^{g_2(n)} \#f_2(n)$$

Si l'on compare le w dans $P(M)$ et dans Q_2, on a $p = r_2(n)$ et $q = g_2(n)$. Soit h la fonction définie par $h(n) = n + r_2(n) + 1 + g_2(n)$. Alors, on peut écrire le chemin w dans Q_1 comme suit :

$$\pi_1(w) = \gamma\$^{h(n)}\# \sqcup^{r_1(h(n))} k_0 \sqcup^{g_1(h(n))} \#f_1(h(n))$$

Comme nous l'avons dit précédemment, $f_1 = f_2$, $g_1 = g_2$ et $r_1 = r_2$. Donc il existe un chemin infini w' dans Q_2 tel que $\pi_2(w') = \pi_1(w)$. En d'autres termes, il existe un chemin infini w' dans Q_2 avec :

$$\pi_2(w') = \gamma\$^{h(n)}\# \sqcup^{r_2(h(n))} k_0 \sqcup^{g_2(h(n))} \#f_2(h(n))$$

Résumons certaines propriétés des chemins du processus Q_2 que nous avons montré. Il existe deux chemins infinis w' et w dans le processus Q_2 tel que :

- $\pi_2(w') = \gamma\$^{h(n)}\#C_1\#C_2\cdots$
- $\pi_2(w) = \gamma\$^n\#D_0\#D_1\#D_2\cdots$
- C_j et D_j sont configurations, $C_j \vdash D_j$ pour tout $j \geq 1$.
- $C_1 = \sqcup^{r_2(h(n))}k_0\sqcup^{g_2(h(n))}$

Écrivons $C_j^{h(n)}$ la configuration C_j et C_j^n la configuration D_j. Plus généralement, on écrit $C_0^kC_1^k\cdots$ la succession des configurations dans la seconde projection du chemin infini dans Q_2 qui a le préfixe u avec $\pi_2(u) = \gamma\$^k\#$. Puisque n est arbitraire et avec nos nouvelles notations, on a montré que pour tout entier n, il existe $m = h(n) > n$ tel que pour tout $j \geq 0$, $C_j^m \vdash C_{j+1}^n$. Donc pour tout entier l, on peut construire the suite :

$$C_0^{h^l(0)} \vdash C_1^{h^{l-1}(0)} \vdash C_2^{h^{l-2}(0)} \vdash \cdots \vdash C_{l-1}^{h(0)} \vdash C_l^0$$

Mais l'ensemble des états de Q_2 est fini et donc le nombre de configurations dans Q_2 est aussi fini. Donc, il existe des entiers l, i, j, avec $i \neq j$, tel que $C_i^{h^{l-i}(0)} = C_j^{h^{l-j}(0)}$. Finalement, il existe $C_0 = C_0^{h^l(0)} = \sqcup^{r_2(h^l(0))} k_0 \sqcup^{g_2(h^l(0))}$ et $C = C_j^{h^{l-j}(0)}$ tel que $C_0 \vdash^* C \vdash^+ C$. □

Afin de prouver le théorème 5.3.9 (i.e. le problème **PCDI** est indécidable), il reste à prouver la réciproque du théorème 5.3.18. Supposons qu'il existe $C_0 = \sqcup^m k_0 \sqcup^n$ pour des entiers $m, n \geq 0$ et une configuration C tel que $C_0 \vdash^* C \vdash^+ C$. Alors il existe des configurations tel que :

$$C_0 \vdash C_1 \vdash \cdots C_{j-1} \vdash C \vdash C_{j+1} \cdots \vdash C_p \vdash C$$

Soit Q_i le processus défini par la figure 5.11, pour $i = 1, 2$.

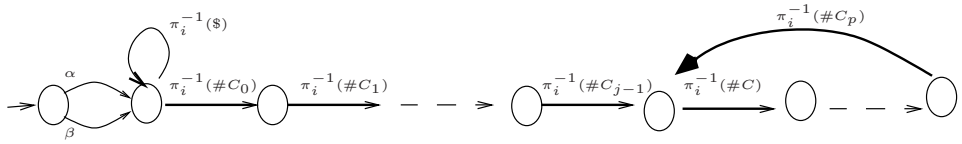


FIG. 5.11 – Processus Q_i

On peut aisément vérifier que $Q_i \vDash PATH_i(M) \wedge \Pi_i$. De plus, observons que les chemins infinis dans $P(M) \times Q_1 \times Q_2$ sont d'une des trois formes suivantes :

- $\alpha(\$, \$)^k(\#C_0, \#C_0)(\#C_1, \#C_1) \cdots$
- $\beta(\$, \$)^k(\$^{|\#C_0|}, \#C_0)(\#C_0, \#C_1)(\#C_1, \#C_2) \cdots$
- $\alpha(\$, \$)^\omega$

Ceci nous permet de conclure que $P(M) \times Q_1 \times Q_2 \vDash \mathcal{A}$. □

5.4 Conclusion

Le problème du contrôle décentralisé avec événements indiscernables ou inobservables est donc indécidable si les spécifications, pour au moins deux contrôleurs, sont des automates étendus.

Le cas où un seul des contrôleurs a pour spécification un automate étendu est décidable puisqu'il s'agit de la satisfiabilité de l'automate

$$\left(\mathcal{A}/\mathcal{B}_n/\mathcal{B}_{n-1}/\cdots\mathcal{B}_2/P\right) \wedge \mathcal{B}_1$$

où seul l'automate $\mathcal{B}_1$ est un automate étendu, les autres automates étant des automates simples (voir la partie 2.2).

On peut observer aussi que les spécifications pour les contrôleurs et le système supervisé, présentées dans ce chapitre afin de prouver l'indécidabilité du problème **DCPIO**, sont des cas très particuliers d'automates étendus.

En effet, dans le cas où un des contrôleurs a des contraintes d'observabilité, toutes les spécifications sont des automates étendus avec condition d'accessibilité (*Acc* est vide). Dans le cas du contrôle avec événements indiscernables, ce sont des automates étendus à parité dans $\{0, 1, 2\}$. Dans les trois cas, les événements indiscernables ou inobservables sont fixés, il ne varie pas en fonction du comportement du SED.

Chapitre 6

Cas décidables

Nous présentons dans ce chapitre deux cas où le contrôle décentralisé est décidable. Dans la partie 6.1, nous prouvons qu'en restreignant fortement les spécifications des contrôleurs et du système supervisé, le problème **DCPIO** devient décidable. La partie 6.2 est dédiée au cas où l'on restreint le type de transitions dans le système supervisé.

Nous présentons enfin, à la partie 6.3, l'opération de quotient d'un automate étendu par un automate simple.

6.1 Restrictions des spécifications

Bien que le contrôle décentralisé avec évènements inobservables ou indiscernables est indécidable, il existe des spécifications pour lesquelles la décidabilité a été prouvée (voir [CDFV88] et [RuWo92]). Dans [AVW03], les auteurs reprennent les travaux de [RuWo92] avec le formalisme des automates comme suit.

Soit P un DES. On considère, pour $i = 1, 2$, des sous-ensembles d'évènements A_c^i et A_o^i représentant, respectivement, les évènements contrôlables et observables pour un contrôleur Q_i. Le langage du système supervisé $\mathcal{L}(P \times Q_1 \times Q_2)$ doit être un sous-langage $K \subset \mathcal{L}(P)$.

Avec, les exemples 2.3.1 et 3.3.1, cela revient à dire que $Q_i \vDash \mathcal{A}(A_c^i) \wedge \mathcal{A}(A_o^i)$. De plus, si K est le langage marqué (par $E \subset \Lambda$) $\mathcal{L}_E(R)$ d'un processus R, avec l'exemple 2.5.1, on doit avoir $P \times Q_1 \times Q_2 \vDash \mathcal{A}_R^E$.

On peut remarquer que les automates $\mathcal{A}(A_c^i)$, $\mathcal{A}(A_o^i)$ et $\mathcal{A}_R^E$ n'ont pas d'états existentiels (voir les exemples déjà cités). De plus $\mathcal{A}_R^E$ n'a qu'un seul modèle à bisimulation près. Des restrictions sur les spécifications peuvent donc permettre la décidabilité du contrôle décentralisé.

Dans cette partie, nous montrons qu'en restreignant les spécifications pour le DES et les contrôleurs, on peut obtenir un cas décidable de contrôle décentralisé (qui prend en compte le cas ci-dessus). Plus précisément, nous allons décrire une sous-classe de spécifications et des modèles "particuliers" de ces spécifications tel que le problème **DCPIO** a une solution si et seulement si les modèles particuliers sont une solution.

6.1.1 Préordre sur les processus

Nous rappelons d'abord la définition 3.7.1.

Définition 6.1.1 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ et $P' = \langle A, \Lambda, S', s'_0, e', \lambda'\rangle$ deux processus. On dit que P est *moins défini* que P' et l'on note $P \leq_d P'$, si $P \leq P'$ (voir Définition 0.4.2) et, pour tout état (s, s') du produit $P \times P'$, les propriétés suivantes sont vérifiées :

- Si $e(s, a)$ est défini et égale à s, alors $e'(s', a) = s'$.
- Si $e(s, a)$ et $e(s, b)$ sont définis et égaux, alors $e'(s', a) = e'(s', b)$.
- $\lambda_P(s) \supseteq \lambda_{P'}(s')$

On note $P =_d P'$ quand $P \leq_d P'$ et $P' \leq_d P$.

Proposition 6.1.2 $\leq_d$ est un préordre.

Preuve : Supposons que $P \leq_d P'$ et $P' \leq_d P''$. Soit (s, s'') un état de $P \times P''$ accessible depuis (s_0, s''_0) avec le chemin $u \in A^*$ tel que $e(s, a)$ et $e(s, a)$ sont définis. Comme $P \leq P'$, il existe un état s' tel que $e'(s'_0, u)$ est défini et égale à s' et, de plus, $e'(s', a)$ et $e'(s', b)$ sont définis. Puisque $P' \leq P''$, $e''(s'', a)$ et $e''(s'', a)$ sont aussi définis. Supposons que $e(s, a) = s$. alors $e'(s', a) = s'$ et $P' \leq_d P''$ implique que $e''(s'', a) = s''$. Supposons que $e(s, a) = e(s, a)$. Puisque $P \leq_d P'$, $e'(s', a) = e'(s', b)$. Puisque $P' \leq_d P''$, $e''(s'', a) = e''(s'', a)$. De plus, $P \leq_d P'$ implique que $\lambda_P(s) \supseteq \lambda_{P'}(s')$ et $P' \leq_d P''$ implique que $\lambda_P(s') \supseteq \lambda_{P''}(s'')$. Donc $\lambda_P(s) \supseteq \lambda_{P''}(s'')$. Ainsi $P \leq_d P''$. □

Nous présentons maintenant des propriétés très utiles.

Lemme 6.1.3 Soit P_1, P_2, Q_1, Q_2 des processus.

- Si $P_1 \leq_d Q_1$ et $P_2 \leq_d Q_2$, alors $P_1 \times P_2 \leq_d Q_1 \times Q_2$.
- Si $P_1 \leq_d Q_1 \times Q_2$, alors $P_1 \leq_d Q_1$ et $P_2 \leq_d Q_2$.
- Si $P_1 \leq_d P_2$, alors $P_1 \times P_2 =_d P_1$.

Preuve : Soit des processus, pour $i = 1, 2$, $P_i = \langle A, \Lambda, S_i, s_0^i, e_i, \lambda_i \rangle$ et $Q_i = \langle A, \Lambda, S_i', s_0'^i, e_i', \lambda_i' \rangle$.

Montrons la première implication. Supposons que $P_i \leq_d Q_i$. Soit $u \in A^*$ tel que, $e_i(s_0^i, u)$ et $e_i'(s_0'^i, u)$ sont définis et égaux respectivement à s_i et s_i'. Soit $a, b \in A$ tel que, $e_1 \times e_2((s_1, s_2), a)$ et $e_1 \times e_2((s_1, s_2), b)$ sont définis. Puisque $P_i \leq Q_i$, $e_i'(s_i', a)$ et $e_i'(s_i', b)$ sont définis. Alors $P_1 \times P_2 \leq Q_1 \times Q_2$. Supposons que $e_1 \times e_2((s_1, s_2), a) = (s_1, s_2)$. Alors $P_i \leq_d Q_i$ implique que $e_i'(s_i', a) = s_i'$. Supposons que $e_1 \times e_2((s_1, s_2), a) = e_1 \times e_2((s_1, s_2), b)$. Alors $P_i \leq_d Q_i$ implique que $e_i'(s_i', a)$ et $e_i'(s_i', b)$ sont égaux. De plus, $P_i \leq_d Q_i$ implique que $\lambda_{P_i}(s_i) \supseteq \lambda_{Q_i}(s_i')$. Donc $\lambda_{P_1}(s_1) \cup \lambda_{P_2}(s_2) \supseteq \lambda_{Q_1}(s_1') \cup \lambda_{Q_2}(s_2')$. Donc $P_1 \times P_2 \leq_d Q_1 \times Q_2$.

La preuve de la deuxième propriété (i.e. $P_1 \leq_d Q_1 \times Q_2$ alors $P_1 \leq_d Q_1$ et $P_2 \leq_d Q_2$) est similaire.

Supposons maintenant que $P_1 \leq_d P_2$. $P_1 \leq_d P_1 \times P_2$ est une conséquence de la première propriété. Il reste à prouver que $P_1 \times P_2 \leq_d P_1$. Soit $u \in A^*$ tel que $e_1 \times e_2((s_0^1, s_0^2), u)$ est défini et égale à (s_1, s_2). Soit $a, b \in A$ tel que $e_1 \times e_2((s_1, s_2), a)$ et $e_1 \times e_2((s_1, s_2), b)$ sont définis. Alors $e_1(s_1, a)$ et $e_1(s_1, b)$ sont définis. Si $e_1 \times e_2((s_1, s_2), a) = e_1 \times e_2((s_1, s_2), b)$, alors $e_1(s_1, a) = e_1(s_1, b)$. De plus, $\lambda_{P_1 \times P_2}(s_1, s_2) = \lambda_{P_1}(s_1) \cup \lambda_{P_2}(s_2) \supseteq \lambda_{P_1}(s_1)$. □

6.1.2 Modèle minimal d'un automate étendu

Définition 6.1.4 Soit $\mathcal{A}$ automate étendu. $\mathcal{A}$ a un *modèle minimal* si il existe un modèle $M_\mathcal{A}$ de $\mathcal{A}$ tel que pour tout modèle P de $\mathcal{A}$, on a $M_\mathcal{A} \leq_d P$.

Lemme 6.1.5 Soit $\mathcal{A}$ un automate étendu avec un modèle minimal $M_\mathcal{A}$. Soit P un processus. $P \vDash \mathcal{A}/M_\mathcal{A}$ si et seulement si $M_\mathcal{A} \leq_d P$

Preuve : Supposons que $P \vDash \mathcal{A}/M_\mathcal{A}$. Alors $M_\mathcal{A} \times P \vDash \mathcal{A}$. Alors $M_\mathcal{A} \leq_d M_\mathcal{A} \times P$. D'après le lemme 6.1.3, $M_\mathcal{A} \leq_d P$. Réciproquement, si $M_\mathcal{A} \leq_d P$ alors, d'après le lemme 6.1.3, $M_\mathcal{A} \times P =_d M_\mathcal{A}$. Donc $M_\mathcal{A} \times P \vDash \mathcal{A}$ car $M_\mathcal{A}$ est un modèle minimale de $\mathcal{A}$. □

6.1.3 Un cas décidable

Théoreme 6.1.6 Supposons que $\mathcal{A}$ a un modèle minimal $M_\mathcal{A}$ et, pour tout processus P et tout modèle P' de $\mathcal{A}$, si $M_\mathcal{A} \leq_d P \leq_d P'$ alors $P \vDash \mathcal{A}$. De plus, supposons que pour tout $i = 1, \ldots, n$, $\mathcal{B}_i \wedge \mathcal{A}/M_\mathcal{A}$ a un modèle

minimal M_i. Alors le problème **DCPIO** a une solution si et seulement si $P \times M_1 \times \cdots \times M_n \vDash \mathcal{A}$.

Preuve : Puisque $M_i \vDash \mathcal{B}_i$, si $P \times M_1 \times \cdots \times M_n \vDash \mathcal{A}$ alors M_i $(i = 1, \ldots, n)$ est une solution.

Supposons que le problème a une solution Q_i $(i = 1, \ldots, n)$. Nous allons maintenant prouver que $P \times M_1 \times \cdots \times M_n \vDash \mathcal{A}$. D'après les propriétés de $\mathcal{A}$, il est suffisant de prouver que $M_\mathcal{A} \leq_d P \times M_1 \times \cdots \times M_n \leq_d P \times Q_1 \times \cdots \times Q_n$.

Puisque $P \times Q_1 \times \cdots \times Q_n \vDash \mathcal{A}$, on a $M_\mathcal{A} \leq_d P \times Q_1 \times \cdots \times Q_n$. D'après Le Lemme 6.1.3, pour tout $i = 1, \ldots, n$, $M_\mathcal{A} \leq_d Q_i$. D'après Le Lemme 6.1.5, pour tout $i = 1, \ldots, n$, $Q_i \vDash \mathcal{A}/M_\mathcal{A}$. Mais $Q_i \vDash \mathcal{B}_i$, donc $M_i \leq_d Q_i$. D'après Le Lemme 6.1.3, on a $P \times M_1 \times \cdots \times M_n \leq_d P \times Q_1 \times \cdots \times Q_n$.

On a $M_\mathcal{A} \leq_d P \times Q_1 \times \cdots \times Q_n$. D'après Le Lemme 6.1.3, $M_\mathcal{A} \leq_d P$. De Plus, d'après le lemme 6.1.5, pour tout $i = 1, \ldots, n$, $M_\mathcal{A} \leq_d M_i$ car $M_i \vDash \mathcal{A}/M_\mathcal{A}$. D'après Le Lemme 6.1.3, on a $M_\mathcal{A} \leq_d P \times M_1 \times \cdots \times M_n$. □

6.1.4 Sous-classes d'automates

Dans ce paragraphe, nous allons décrire des sous-classes d'automates étendus qui permettent de satisfaire les hypothèses du théorème 6.1.6.

Définition 6.1.7 Soit $\mathcal{A} = \langle A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc \rangle$ un automate étendu biparti. On dit que $\mathcal{A}$ est *déterministe* si, pour tout $q \in Q^\exists$, on a $|\delta(q)| \leq 1$.

Remarque 6.1.8 Soit P un processus et $\mathcal{A}$ un automate étendu déterministe. Dans le jeu d'acceptation $G(\mathcal{A}, P)$, il existe seulement une stratégie qui peut être gagnante. Néanmoins, $\mathcal{A}$ peut avoir plusieurs modèles. En effet, soit $\mathcal{A}^a$ un automate simple $\langle A, Q, \{q_0\}, \{q_1\}, q_0, \delta, Q^\omega \rangle$ avec $\delta(q_0) = \{\epsilon, q_1\}$ et $\delta(q_1) = \{(a, q_0)\} \cup \{\nrightarrow_b : b \neq a\}$. Les modèles de $\mathcal{A}^a$ sont les processus ayant uniquement des a-transitions.

Proposition 6.1.9 Si un automate étendu déterministe $\mathcal{A}$ a un modèle, alors il a un modèle minimal $M_\mathcal{A}$.

Preuve : Ce modèle minimal peut être effectivement calculé : c'est l'unique modèle (à la relation $=_d$ près) de l'automate déterministe $\mathcal{A}^-$ obtenu en ajoutant dans l'automate $\mathcal{A}$, pour tout $q \in Q^\forall$, $\nrightarrow_a \in \delta(q)$ si $\rightarrow_a \notin \delta(q)$. □

Remarquons que la réciproque n'est pas vrai. En effet, l'automate $\mathcal{A}^a \vee \mathcal{A}^b$ (voir Remarque 6.1.8) a un modèle minimal (le processus sans transitions), mais n'est pas équivalent à un automate déterministe.

Proposition 6.1.10 Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates étendus déterministes et P un processus. Les automates $\mathcal{A} \wedge \mathcal{B}$ et $\mathcal{A}/P$ sont eux aussi déterministes.

Preuve : On peut vérifier que nous avons défini dans la section 3.6.1 de la partie 3.6 un quotient $\mathcal{A}/P$ qui reste déterministe si $\mathcal{A}$ est déterministe. □

Définition 6.1.11 Soit $\mathcal{A} = \langle A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate étendu. On dit que $\mathcal{A}$ est *conique* si $\mathcal{A}$ est *déterministe* et pour tout $q, q', q'' \in Q$ et $a \in A$ tel que $(a, q') \in \delta(q)$ et $(\epsilon, q'') \in \delta(q')$, si $\rightarrow_a \in \delta(q'')$ alors $\rightarrow_a \in \delta(q)$.

Proposition 6.1.12 Soit $\mathcal{A}$ un automate étendu conique avec un modèle minimal $M_\mathcal{A}$. Pour tout modèle P de $\mathcal{A}$ et tout processus Q, si $M_\mathcal{A} \leq_d Q \leq_d P$ alors Q est un modèle de $\mathcal{A}$

Preuve : Soit $\mathcal{A} = \langle A, Q, Q^\exists, Q^\forall, q_0, \delta, Acc\rangle$ un automate étendu conique avec un modèle minimal $M_\mathcal{A} = \langle A, \Lambda, S^M, s_0^M, e^M, \lambda^M\rangle$. Soit $P = \langle A, \Lambda, S, s_0, e, \lambda\rangle$ un modèle de $\mathcal{A}$ et $Q = \langle A, \Lambda, S', s_0', e', \lambda'\rangle$ un processus tel que $M_\mathcal{A} \leq_d Q \leq_d P$. Soit u un mot dans A^* tel que $e'(s_0', u)$ est défini et égale à s'. Alors $e(s_0, u)$ est défini car $Q \leq P$, disons $e(s_0, u) = s$. Puisque $\mathcal{A}$ est déterministe, u correspond à une unique suite de transitions de $\mathcal{A}$, disons $q_0 q_1 q_1' q_2 q_2' \cdots q_n q_n' q_{n+1}$ où $q_i' \in Q^\exists$ et $q_i \in Q^\forall$. La propriété $Q \leq_d P$ et le fait que (q_{n+1}, s) est une position gagnante dans le jeu $G(\mathcal{A}, P)$ est suffisant pour prouver qu'il n'y a pas d'arête depuis (q_{n+1}, s') vers $\bot$ dans le jeu $G(\mathcal{A}, Q)$ si $\nrightarrow_a \in \delta(q_{n+1})$ ou si $\downdownarrows_{a,b} \in \delta(q_{n+1})$ ou si $\not\rightrightarrows_{a,b} \in \delta(q_{n+1})$. Supposons maintenant que $\rightarrow_a \in \delta(q_{n+1})$. Puisque $\mathcal{A}$ est conique, si $\rightarrow_a \in \delta(q_{k+1})$ alors $\rightarrow_a \in \delta(q_k)$. Donc, pour tout $k = 1, \ldots, n$, $\rightarrow_a \in \delta(q_k)$. Mais $M_\mathcal{A} \vDash \mathcal{A}$, donc $e^M(s_0^M, ua)$ doit être défini. Comme $M_\mathcal{A} \leq Q$, $e'(s_0', ua)$ doit être défini. Finalement, observons que les chemins infinis depuis la position initiale dans le jeu $G(\mathcal{A}, Q)$ correspondent à des suites d'états de l'automate qui sont parmi celles du jeu $G(\mathcal{A}, P)$. Ainsi, ces suites satisfont la condition Acc. □

Théoreme 6.1.13 Si $\mathcal{A}$ est conique et, si, pour tout i, $\mathcal{B}_i$ est déterministe, alors le problème **DCPIO** est décidable.

Preuve : Avec les propositions 6.1.9, 6.1.10 et 6.1.12, on peut facilement vérifier que les hypothèses du théorème 6.1.6 sont satisfaites. □

Les automates déterministes permettent d'obtenir des spécifications pour les contrôleurs intéressantes. Ils n'empêchent pas de spécifier un ensemble d'événements incontrôlables et un ensemble d'événements inobservables ou une partition d'événements indiscernables (voir les exemples 2.3.1, 3.3.1 et 3.3.2). En effet, les automates sans état existentiel (i.e. $Q^\exists = \emptyset$) sont équivalents à des automates déterministes. On peut aussi définir des contraintes dynamiques comme aux exemples 2.3.3 et 3.3.4. Cependant le cas où deux

événements ne peuvent être contrôlé en même temps ne peut être spécifié par un automate déterministe (voir exemple 2.3.2).

Les automates coniques peuvent permettre de spécifier que les comportements finis d'un système supervisé $P \times Q$ réalise un language requis $\mathcal{L}_R$ mais reste dans un language admissible $\mathcal{L}_A$:

$$\mathcal{L}_R \leq \mathcal{L}(P \times Q) \leq \mathcal{L}_A$$

Il faut toutefois que les languages $\mathcal{L}_R$ et $\mathcal{L}_A$ soit des languages réguliers clôs par préfixes. Pour cela, supposons que les comportements admissibles soient décris par un processus $Ad = \langle A, S_a, s_0^a, e_a \rangle$ et que les comportement requis soit définis par le processus $Re = \langle A, S_r, s_0^r, e_r \rangle$. On définit alors $\mathcal{A}(Re, Ad) = \langle A, Q, \emptyset, (S_r \times S_a) \cup S_a, (s_0^r, s_0^a), \delta_{r,a}, Q^\omega \rangle$ un automate simple tel que, pour tout $s \in S_r$ et $s' \in S_a$, $\delta_{r,a}(s, s')$ contient :

- $\{\rightarrow_b, (b, (e_r(s, b), e_a(s', b))\}$ si $e_r(s, b)$ et $e_a(s', b)$ sont définis.
- $(b, e_a(s', b))$ si $e_r(s, b)$ n'est pas défini et $e_a(s', b)$ est défini.
- $\nrightarrow_b$ si $e_a(s', b)$ n'est pas défini.

Et, pour tout $s' \in S_a$, $\delta_{r,a}(s')$ contient $(b, e_a(s', b))$ si $e_a(s', b)$ est défini et $\nrightarrow_b$ sinon. On peut vérifier que $\mathcal{A}(Re, Ad)$ est équivalent à un automate conique (les états existentiels ont juste été supprimés), que les transitions "exigées" sont celles du processus Re et que les transitions interdites sont celles pour lesquelles Ad n'est pas défini.

6.2 Synchronisation des processus

Nous avons vu dans le chapitre 5 que le problème de contrôle décentralisé avec évènements inobservables ou indiscernables est indécidable. Il n'est donc pas possible d'effectuer l'opération de quotient d'automates pour les automates étendus comme cela a été présenté pour les automates simples au chapitre 2.

Dans cette partie, nous présentons un cas où il est possible d'effectuer cette opération de quotient d'automates étendus. Il s'agit de considérer que les transitions d'un produit $P \times Q$ de deux processus ne peuvent s'effectuer que dans certains cas décrits au paragraphe suivant.

6.2.1 Résultat principal

Définition 6.2.1 Soit $\pi = \{N, L, A_1, \ldots, A_p\} \in \Pi_A$ et $\pi' = \{N', L', A'_1, \ldots, A'_k\} \in \Pi_A$ (voir Notation 3.4.2).

On note $C_1(\pi, \pi')$ la condition :
Il existe $k_0 \in]p]$ tel que $A_{k_0} \cap L' \neq \emptyset$ et

$$\bigcup_{i=1}^{k} A'_i \subseteq N \text{ et } \bigcup_{\substack{j=1 \\ j \neq k_0}}^{p} A_j \subseteq N'$$

On note $C_2(\pi, \pi')$ la condition :
Pour tout $j \in]p]$, $A_j \cap L' = \emptyset$ et pour tout $i \in]k]$, il existe $j \in]p]$ tel que :

$$A'_i \subseteq A_j \cup N \text{ ou } A'_i \subseteq L \cup N$$

Définition 6.2.2 Soit $P = \langle A, \Lambda, S, s_0, e, \lambda \rangle$ et $R = \langle A, \Lambda, S', s'_0, e', \lambda' \rangle$ deux processus. On dit que le processus $P \times R$ est *bien synchronisé* si pour tout état accessible (s, s'), la condition $C_1(\Pi_P(s), \Pi_R(s'))$ ou la condition $C_2(\Pi_P(s), \Pi_R(s'))$ (voir Définition 3.4.3) est vrai.

Remarque 6.2.3 En d'autres termes, $P \times R$ est bien synchronisé (voir aussi la figure 6.1) si :

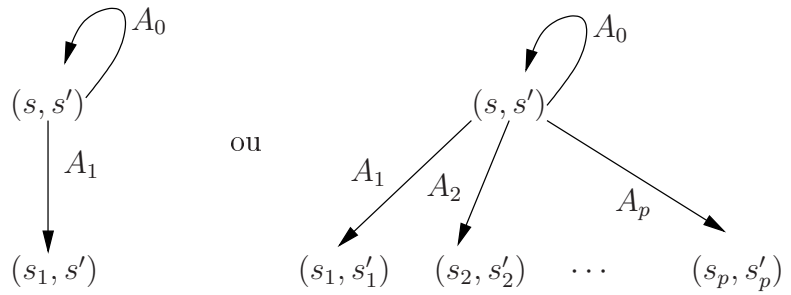
- Si il existe $a \in A$ tel que $a \in D_P(s) \cap D_R(s')$, $e'(s', a) = s'$ et $e(s, a) \neq s$ alors, pour tout $b \in D_P(s) \cap D_R(s')$, $e'(s', b) = s'$ et soit $e \times e'((s, s'), b) = e \times e'((s, s'), a)$ ou soit $e \times e'((s, s'), b) = (s, s')$.
- Si pour tout $a \in \ \in D_P(s) \cap D_R(s')$, $(e'(s', a) \neq s'$ ou $e(s, a) = s)$, alors pour tout $b, c \in D_P(s) \cap D_R(s')$ tel que $e'(s', b) = e'(s', c)$, on a $e(s, b) = e(s, c)$.

Théoreme 6.2.4 Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates étendus. Il existe un automate étendu, noté $\mathcal{A}/_{syn}/\mathcal{B}$, tel que $P \vDash \mathcal{A}/_{syn}/\mathcal{B}$ si et seulement si il existe $R \vDash \mathcal{B}$ tel que $P \times R \vDash \mathcal{A}$ et $P \times R$ est bien synchronisé.

Corollaire 6.2.5 L'automate $((((\mathcal{A}/_{syn}/\mathcal{B}_n)/_{syn}/\mathcal{B}_{n-1}) \cdots /_{syn}/\mathcal{B}_2)/P) \wedge \mathcal{B}_1$ a un modèle si et seulement si le problème de contrôle décentralisé **DCPOI** a une solution $Q_1, Q_2, \ldots, Q_n$ tel que, pour tout $i = 2, 3, \ldots n$, $Q_i \times R_{i-1}$ est bien synchronisé avec $R_{i-1} = Q_1 \times \cdots \times Q_{i-1}$.

Preuve : Soit $i < n$. D'après le théorème 6.2.4, on a pour tout $k < i$, $Q_k \vDash \mathcal{B}_k$ et

$$P \times Q_1 \times \cdots \times Q_{i-1} \times Q_i \vDash (((\mathcal{A}/_{syn}/\mathcal{B}_n)/_{syn}/\mathcal{B}_{n-1}) \cdots /_{syn}/\mathcal{B}_{i+1})$$



Avec $s_1 \neq s$

Avec $s'_i \neq s'_j$ et $s'_i \neq s'$

Et $A_0, A_1, \ldots, A_p \subseteq A$
Et $A_i \cap A_j = \emptyset$

FIG. 6.1 – $P \times R$ bien synchronisé

si et seulement si il existe $Q_{i+1} \vDash \mathcal{B}_{i+1}$ et

$$P \times Q_1 \times \cdots \times Q_i \times Q_{i+1} \vDash (((\mathcal{A}/_{syn}\mathcal{B}_n)/_{syn}\mathcal{B}_{n-1}) \cdots /_{syn}\mathcal{B}_{i+2})$$

et $R_i \times Q_{i+1}$ est bien synchronisé. □

Nous verrons dans les sections 6.2.5 et 6.2.6 deux applications du théorème 6.2.4.

6.2.2 Définition de l'automate $\mathcal{A}/_{syn}\mathcal{B}$

Définition 6.2.6 Soit $\mathcal{A} = \langle A, Q_\mathcal{A}, {Q_\mathcal{A}}^\exists, {Q_\mathcal{A}}^\forall, {q_\mathcal{A}}^0, \delta_\mathcal{A}, Acc_\mathcal{A}\rangle$ et $\mathcal{B} = \langle A, Q_\mathcal{B}, {Q_\mathcal{B}}^\exists, {Q_\mathcal{B}}^\forall, {q_\mathcal{B}}^0, \delta_\mathcal{B}, Acc_\mathcal{B}\rangle$ deux automates étendus nondéterministes à parité. Nous supposons que $\mathcal{A}$ et $\mathcal{B}$ sont complets et complètement spécifiés (voir Définition 3.8.9). On suppose de plus que $\mathcal{B}$ est élagué (voir Définition 3.8.10) et vérifie les condition suivantes : pour tout $a \in A$ et $q, q_1, q_2 \in Q_\mathcal{B}$ on a :

$$\text{si } (\varepsilon, q_1) \in \delta_\mathcal{B}(q) \text{ et } (a, q_2) \in \delta_\mathcal{B}(q_1) \text{ alors } q_2 \neq q \tag{6.2.1}$$

et

$$\text{si } (a, q'_2), (b, q'_3), \not\asymp_{a,b} \in \delta_\mathcal{B}(q'_1) \text{ alors } q'_2 \neq q'_3 \tag{6.2.2}$$

(Il suffit pour cela de dupliquer certains états de $\mathcal{B}$)

Soit $\mathcal{A}/_{syn}\mathcal{B} = \langle A, Q_/, Q^\exists_/, Q^\forall_/, q^0_/, \delta_/, Acc_/\rangle$ un automate défini comme suit :

$$Q^\exists_/ = {Q_\mathcal{A}}^\exists \times {Q_\mathcal{B}}^\exists \cup {Q_\mathcal{A}}^\exists \times {Q_\mathcal{B}}^\forall \cup \{q_\perp\}$$

$$Q_/^\forall = {Q_\mathcal{A}}^\forall \times {Q_\mathcal{B}}^\forall \quad \text{et} \quad q_/^0 = ({q_\mathcal{A}}^0, {q_\mathcal{B}}^0)$$

$$Acc_/ = \left\{ (q_0, q_0')(q_1, q_1') \cdots \Big| \quad q_0 q_1 \cdots \in Acc_\mathcal{A} \text{ et } q_0' q_1' \cdots \in Acc_\mathcal{B} \right\}$$

- Soit $(q_1, q_1') \in Q_/^\forall$. On note $D(q_1, q_1')$ la conjonction des conditions suivantes :
  - Si $\rightarrow_a \in \delta_\mathcal{A}(q_1)$ alors $\rightarrow_a \in \delta_\mathcal{B}(q_1')$
  - Si $\circlearrowleft_a \in \delta_\mathcal{A}(q_1)$ alors $\circlearrowleft_a \in \delta_\mathcal{B}(q_1')$
  - Si $\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$ alors ($\downdownarrows_{a,b} \in \delta_\mathcal{B}(q_1')$ ou $\circlearrowleft_a, \circlearrowleft_b \in \delta_\mathcal{B}(q_1')$).
- Pour tout état existentiel $(q, q') \in Q_\mathcal{A}^\exists \times Q_\mathcal{B}^\exists$, $\delta_/(q, q')$ contient $(\varepsilon, (q_1, q_1'))$ si $(\varepsilon, q_1) \in \delta_\mathcal{A}(q)$ et $(\varepsilon, q_1') \in \delta_\mathcal{B}(q')$ et $D(q_1, q_1')$ est vrai.
- Pour tout état existentiel $(q, q_1') \in Q_\mathcal{A}^\exists \times Q_\mathcal{B}^\forall$, $\delta_/(q, q_1')$ contient $(\varepsilon, (q_1, q_1'))$ si $(\varepsilon, q_1) \in \delta_\mathcal{A}(q)$ et $D(q_1, q_1')$ est vrai.
- $\delta_/(q_\perp) = \emptyset$

Soit $(q_1, q_1') \in Q_/^\forall$. On note $C(q_1, q_1')$, $C_1(q_1, q_1')$ et $C_2(q_1, q_1')$ les conditions suivantes :

$C(q_1, q_1') \equiv \{a \in A : \rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{A}(q_1) \text{ et } \rightarrow_a, \circlearrowleft_a \in \delta_\mathcal{B}(q_1')\} \neq \emptyset$

$C_1(q_1, q_1') \equiv$ Pour tout $a, b \in A$, si $\rightarrow_a, \rightarrow_b \in \delta_\mathcal{A}(q_1)$, alors $\circlearrowleft_a, \circlearrowleft_b \in \delta_\mathcal{B}(q_1')$ et si $\not\circlearrowleft_a, \not\circlearrowleft_b \in \delta_\mathcal{A}(q_1)$, alors $\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$.

$C_2(q_1, q_1') \equiv$ Pour tout $a, b \in A$, si $\downdownarrows_{a,b} \in \delta_\mathcal{B}(q_1')$ et $\rightarrow_a, \rightarrow_b \in \delta_\mathcal{A}(q_1)$, alors $\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$ ou $\circlearrowleft_a, \circlearrowleft_b \in \delta_\mathcal{A}(q_1)$.

Définissons maintenant les transitions depuis un état universel $(q_1, q_1') \in Q_/^\forall$. $\delta_/(q_1, q_1')$ contient :

- $\rightarrow_a$, si $\rightarrow_a \in \delta_\mathcal{A}(q_1)$
- $\circlearrowleft_a$, si $\circlearrowleft_a \in \delta_\mathcal{A}(q_1)$
- $\downdownarrows_{a,b}$, si $\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$
- $\not\rightarrow_a$, si $\not\rightarrow_a \in \delta_\mathcal{A}(q_1)$ et $\rightarrow_a \in \delta_\mathcal{B}(q_1')$
- $\not\downdownarrows_{a,b}$, si $\not\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$ et ($\downdownarrows_{a,b} \in \delta_\mathcal{B}(q_1')$ ou $\circlearrowleft_a, \circlearrowleft_b \in \delta_\mathcal{B}(q_1')$)

De plus,

- Si $C(q_1, q_1')$ est vrai et $C_1(q_1, q_1')$ est vrai, alors $\delta_/(q_1, q_1')$ contient : $\not\circlearrowleft_a, (a, (q_2, q_1'))$ si $(a, q_2), \rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{A}(q_1)$ et $\circlearrowleft_a \in \delta_\mathcal{B}(q_1')$.
- Si $C(q_1, q_1')$ est faux et $C_2(q_1, q_1')$ est vrai, alors $\delta_/(q_1, q_1')$ contient : $(a, (q_2, q_2'))$ si $(a, q_2), \rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{A}(q_1)$ et $(a, q_2'), \rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{B}(q_1')$.
- Sinon, $\delta_/(q_1, q_1')$ contient $\{(\varepsilon, q_\perp)\}$.

6.2.3 Preuve : Si $P \vDash \mathcal{A}/_{syn}/\mathcal{B}$ alors il existe $R \vDash \mathcal{B}$ tel que $P \times R \vDash \mathcal{A}$ et $R \times P$ est bien synchronisé

Supposons qu'il existe une stratégie gagnante $\sigma_/ : (Q_/^\exists \times S_P) \times H \to Q_/^\forall \times S_P$ avec l'histoire H dans $G(\mathcal{A}/_{syn}/\mathcal{B}, P)$ pour le processus $P = \langle A, S_P, s_P^0, e_P \rangle$ avec la fonction de mise à jour $hist : H \times (Q_/ \times S_P) \to H$.

Lemme 6.2.7 Soit $(q_1, q_1', s) \in Q_/^\forall \times S_P$ une position du jeu $G(\mathcal{A}/_{syn}/\mathcal{B}, P)$ accessible par la stratégie $\sigma_/$. Si $C(q_1, q_1')$ est vrai et $C_1(q_1, q_1')$ est vrai, alors la position (q_1, q_1', s) a exactement un unique successeur dans $Q_/^\exists \times S_P$ de la forme (q_2, q_1', s).

Preuve : D'après la condition $C(q_1, q_1')$, il existe $a \in A$ tel que $\to_a, \not\oslash_a \in \delta_\mathcal{A}(q_1)$ et $\to_a, \circlearrowleft_a \in \delta_\mathcal{B}(q_1')$. Puisque $\mathcal{A}$ est complet, il existe q_2 tel que $(a, q_2) \in \delta_\mathcal{A}(q_1)$.

Si il existe $b \in A$ tel que $\to_b, \not\oslash_b \in \delta_\mathcal{A}(q_1)$, d'après la condition $C_1(q_1, q_1')$, alors $\downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$. Puisque $\mathcal{A}$ est nondéterministe, si $(b, q_3) \in \delta_\mathcal{A}(q_1)$ alors $q_3 = q_2$. De plus, $\to_a, \to_b, \downdownarrows_{a,b} \in \delta_\mathcal{A}(q_1)$ implique que $\to_a, \to_b, \downdownarrows_{a,b} \in \delta_/(q_1, q_1')$. Comme (q_1, q_1', s) est une position gagnante du jeu $G(\mathcal{A}/_{syn}/\mathcal{B}, P)$, $e_P(s, a)$ et $e_P(s, b)$ sont définis et égaux. □

Si $\sigma_/((q, q'), s, h)$ est défini, alors on peut ainsi définir une unique suite que l'on note $Seq(q, q', s, h)$:

$$\begin{array}{rll} (q, q', s, h) & \to_{\sigma_/} & (q_1, q_1', s, h_1) \to_{A_2} (q_2, q_1', s_2, h_2) \\ & \to_{\sigma_/} & (q_3, q_1', s_2, h_3) \to_{A_4} (q_4, q_1', s_4, h_4) \\ & \cdots & \\ & \to_{\sigma_/} & (q_{2k+1}, q_1', s_{2k}, h_{2k+1}) \\ & \cdots & \end{array} \tag{6.2.3}$$

tel que :

- $\sigma_/((q, q'), s, h) = ((q_1, q_1'), s)$
- $\sigma_/((q_{2i}, q_1'), s_{2i}, h_{2i}) = ((q_{2i+1}, q_1'), s_{2i})$
- Pour tout $a, b \in A_{2i}$, on a $\to_a, \to_b, \downdownarrows_{a,b}, (a, q_{2i}), (b, q_{2i}) \in \delta_\mathcal{A}(q_{2i-1})$.
- Pour tout $a \notin A_{2i}$, on a $\not\to_a \in \delta_\mathcal{A}(q_{2i-1})$.
- $h_{2i+1} = hist(h_{2i}, (q_{2i}, q_1'), s_{2i})$ et $h_{2i} = hist(h_{2i-1}, (q_{2i-1}, q_1'), s_{2i})$

Lemme 6.2.8

- La suite $Seq(q, q', s, h)$ est infinie si pour tout $k > 0$, $C(q_{2k+1}, q_1')$ est vrai et $C_1(q_{2k+1}, q_1')$ est vrai.

– La suite $Seq(q, q', s, h)$ fini en $(q_{2k+1}, q'_1, s_{2k}, h_{2k+1})$ si $C(q_{2k+1}, q'_1)$ est faux et $C_2(q_{2k+1}, q'_1)$ est vrai.

Preuve : Si, pour tout k, $C(q_{2k+1}, q'_1)$ est vrai et $C_1(q_{2k+1}, q'_1)$ est vrai, alors, d'après le lemme, la suite $Seq(q, q', s, h)$ est infinie. Sinon, il existe $k > 0$ tel que $C(q_{2k+1}, q'_1)$ est faux et $C_2(q_{2k+1}, q'_1)$ est vrai, car, autrement, $\sigma_/$ atteindrait la position perdante $(q_\perp, s_{2k})$ du jeu $G(\mathcal{A}/_{syn}/\mathcal{B}, P)$. □

Puisque $\mathcal{B}$ est élagué, pour tout état $q' \in {Q_\mathcal{B}}^\forall$, il existe un processus $R_{q'} = \langle A, S_{q'}, s^0_{q'}, e_{q'}\rangle$ tel que $R_{q'} \vDash \mathcal{B}_{q'}$ où $\mathcal{B}_{q'}$ est l'automate $\mathcal{B}$ avec l'état initial changé en q'. Définissons maintenant le processus R.

Définition 6.2.9 Soit R le processus $\langle A, S_R, s^0_R, e_R\rangle$ tel que

$$S_R = \bigcup_{q' \in {Q_\mathcal{B}}^\forall} S_{q'} \cup \left\{(q, q', s, h) \in {Q_/}^\exists \times S_P \times H \,\middle|\, \sigma_/((q, q'), s, h) \text{ est définit}\right\}$$

avec $s^0_R = ({q_\mathcal{A}}^0, {q_\mathcal{B}}^0, s^0_P, h^0)$ et $e_R : S_R \times A \to S_R$ est définit comme suit :

1. Pour tout $a \in A$ et $(q, q', s, h) \in S_R$ où $\sigma_/((q, q'), s, h) = ((q_1, q'_1), s)$, on a :
   – Si $\nrightarrow_a \in \delta_\mathcal{B}(q'_1)$ alors $e_R((q, q', s, h), a)$ n'est pas définit.
   – Si $\rightarrow_a, \circlearrowleft_a \in \delta_\mathcal{B}(q'_1)$ alors $e_R((q, q', s, h), a) = (q, q', s, h)$
   – Si $\rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{B}(q'_1)$ et la suite $Seq(q, q', s, h)$ est infinie, alors $e_R((q, q', s, h), a) = s^0_{q'_2}$ avec $(a, q'_2) \in \delta_\mathcal{B}(q'_1)$.
   – Si $\rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{B}(q'_1)$ et la suite $Seq(q, q', s, h)$ fini en $(q_{2k+1}, q'_1, s_{2k}, h_{2k+1})$ avec $\nrightarrow_a \in \delta_\mathcal{A}(q_{2k+1})$ et (pour tout $b \in B$, si $\rightarrow_b \in \delta_\mathcal{A}(q_{2k+1})$ alors $\not\downarrow_{a,b} \in \delta_\mathcal{A}(q_{2k+1})$), alors $e_R((q, q', s, h), a) = s^0_{q'_2}$ avec $(a, q'_2) \in \delta_\mathcal{B}(q'_1)$.
   – Si $\rightarrow_a, \not\circlearrowleft_a \in \delta_\mathcal{B}(q'_1)$ et la suite $Seq(q, q', s, h)$ fini en $(q_{2k+1}, q'_1, s_{2k}, h_{2k+1})$ et ($\rightarrow_a \in \delta_\mathcal{A}(q_{2k+1})$ ou il existe $b \in A$ tel que $\rightarrow_b \in \delta_\mathcal{A}(q_{2k+1})$ et $\downarrow\downarrow_{a,b} \in \delta_\mathcal{B}(q'_1)$), alors $e_R((q, q', s, h), a) = (q_{2k+2}, q'_2, s_{2k+2}, h_{2k+2})$ avec
     – $(b, q_{2k+2}) \in \delta_\mathcal{A}(q_{2k+1})$
     – $(b, q'_2) \in \delta_\mathcal{B}(q'_1)$
     – $e_P(s_{2k}, b) = s_{2k+2}$
     – $h_{2k+2} = hist(h_{2k+1}, ((q_{2k+1}, q'_1), s_{2k}))$
2. Pour tout $a \in A$, $q' \in {Q_\mathcal{B}}^\exists$ et $r \in S_{q'}$ on a $e_R(r, a) = e_{q'}(r, a)$.

Lemme 6.2.10 Si $P \vDash \mathcal{A}/\mathcal{B}$ alors il existe une stratégie gagnante dans $G(\mathcal{B}, R)$ depuis $({q_\mathcal{B}}^0, ({q_\mathcal{A}}^0, {q_\mathcal{B}}^0, s^0_P, h^0))$.

Preuve. Nous allons maintenant décrire une stratégie gagnante $\sigma_\mathcal{B}$ dans $G(\mathcal{B}, R)$. Supposons que $(q', (q, q', s, h))$ est une position du jeu $G(\mathcal{B}, R)$ tel

que $\sigma_/(((q,q'),s),h)$ est définit et égale à $((q_1,q'_1),s)$. Alors $\sigma_{\mathcal{B}}(q',(q,q',s,h)) = (q'_1,(q,q',s,h))$.

Supposons que la suite $Seq(q,q',s,h)$ (voir 6.2.3) est infini. Si il existe $a \in A$ tel que $\rightarrow_a, \not\circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$, comme $\mathcal{B}$ est complet et, d'après la définition de e_R, alors on a une arête de $(q'_1,(q,q',s,h))$ vers la position gagnante $(q'_2, s^0_{q'_2})$ où $(a,q'_2) \in \delta_{\mathcal{B}}(q'_1)$. De plus, d'après la définition de e_R, il n'y a pas d'arêtes depuis $(q'_1,(q,q',s,h))$ vers $\perp$ lorsque $\not\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$. Donc $(q'_1,(q,q',s,h))$ est une position gagnante.

Supposons que la suite $Seq(q,q',s,h)$ fini en $(q_{2k+1},q'_1,s_{2k},h_{2k+1})$. Supposons qu'il existe une arête de $(q'_1,(q,q',s,h))$ vers $(q'_3,(q_{2k+2},q'_2,s_{2k+2},h_{2k+2}))$. Alors il existe $a \in A$ tel que $(a,q'_3) \in \delta_{\mathcal{B}}(q'_1)$ et il existe $b \in A$ tel que $e_R((q,q's,h),b) = (q_{2k+2},q'_2,s_{2k+2},h_{2k+2})$ avec $\rightarrow_b \downdownarrows_{a,b} \delta_{\mathcal{B}}(q'_1)$. Donc $(b,q'_2)\not\circlearrowleft_b \in \delta_{\mathcal{B}}(q'_1)$. Donc $(a,q'_2), \rightarrow_a, \not\circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$. Donc $q'_2 = q'_3$. Donc l'arête depuis $(q'_1,(q,q',s,h))$ vers $(q'_2,(q_{2k+2},q'_2,s_{2k+2},h_{2k+2}))$ dans le jeu $G(\mathcal{B},R)$ correspond à l'unique chemin depuis la position $((q_1,q'_1),s)$ avec l'histoire h vers la position $((q_{2k+2},q'_2),s_{2k+2})$ avec l'histoire h_{2k+2} compatible avec $\sigma_/$ et $hist$ dans le jeu $G(\mathcal{A}/\mathcal{B},P)$. Les chemins infinis dans le jeu $G(\mathcal{B},R)$ sont donc gagnant puisque les parités qui apparaissent sont les mêmes que celles du jeu $G(\mathcal{A}/\mathcal{B},P)$.

Il reste à prouver qu'il n'y a pas d'arête vers $\perp$ depuis $(q'_1,(q,q',s,h))$. Les cas où $\rightarrow_a, \circlearrowleft_a, \not\rightarrow_a \in \delta_{\mathcal{B}}(q'_1)$ sont des conséquences directe de la définition de e_R. Les cas où $\not\circlearrowleft_a \not\downdownarrows_{a,b} \in \delta_{\mathcal{B}}(q'_1)$ sont des conséquences directe des propriétés 6.2.1 et 6.2.2. Supposons que $\rightarrow_a, \rightarrow_b, \downdownarrows_{a,b} \in \delta_{\mathcal{B}}(q'_1)$. Si $e_R((q,q's,h),a) = s^0_{q'_2}$ et $e_R((q,q's,h),b) = s^0_{q'_3}$ alors $q'_2 = q'_3$ car $\mathcal{B}$ est nondéterministe.

Supposons que $e_R((q,q's,h),a) = (q_{2k+2},q'_2,s_{2k+2},h_{2k+2})$. Alors, d'après la définition de e_R, soit $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ ou soit il existe $c \in A$ tel que $\rightarrow_c \in \delta_{\mathcal{A}}(q_1)$ et $\downdownarrows_{a,c} \in \delta_{\mathcal{B}}(q'_1)$. Dans le premier cas $e_R((q,q's,h),a) = e_R((q,q's,h),b)$ par définition de e_R et dans le deuxième cas $e_R((q,q's,h),a) = e_R((q,q's,h),c) = e_R((q,q's,h),b)$. □

Lemme 6.2.11 Si $P \vDash \mathcal{A}/\mathcal{B}$ alors il existe une stratégie gagnante dans $G(\mathcal{A}, P \times R)$ depuis la position $({q_{\mathcal{A}}}^0,(s^0_P,({q_{\mathcal{A}}}^0,{q_{\mathcal{B}}}^0,s^0_P,h^0)))$ et $R \times P$ est bien synchronisé.

Preuve. Nous allons maintenant décrire une stratégie $\sigma_{\mathcal{A}}$ dans $G(\mathcal{A}, P \times R)$.

Supposons que $(q,(s,(q,q',s,h)))$ est une position du jeu $G(\mathcal{A}, P \times R)$ tel que la stratégie dans la position $((q,q'),s)$ du jeu $G(\mathcal{A}/\mathcal{B},P)$ avec l'histoire

h est définit et $\sigma_/((q,q'),s,h) = ((q_1,q'_1,s)$. Alors la stratégie $\sigma_\mathcal{A}$ depuis $(q,(s,(q,q',s,h)))$ est d'aller vers $(q_1,(s,(q,q',s,h)))$.

Supposons que $(q_{2k},(s_{2k},(q,q',s,h)))$ est une position du jeu $G(\mathcal{A},P\times R)$ tel que la stratégie dans la position $((q,q'),s)$ du jeu $G(\mathcal{A}/\mathcal{B},P)$ avec l'histoire h est définit et $\sigma_/((q,q'),s,h) = (q_1,q'_1,s)$. De plus, on considère que la suite $Seq(q,q',s,h)$ (voir 6.2.3) atteint $(q_{2k},q'_1,s_{2k},h_{2k})$ avec $\sigma_/((q_{2k},q'_1),s_{2k},h_{2k}) = (q_{2k+1},q'_1,s_{2k})$. Alors la stratégie $\sigma_\mathcal{A}$ depuis $(q_{2k},(s_{2k},(q,q',s,h)))$ est d'aller vers $(q_{2k+1},(s_{2k},(q,q',s,h)))$.

Soit $(q,(s,(q,q',s,h)))$ une position du jeu $G(\mathcal{A},P\times R)$ accessible par $\sigma_\mathcal{A}$. Supposons que la suite $Seq(q,q',s,h)$ ne fini pas en $(q_{2k+1},q'_1,s_{2k},h_{2k+1})$ et qu'il existe une arête depuis $(q_{2k+1},(s_{2k},(q,q',s,h)))$ vers $(q'',(s'',r))$. Alors il existe $a\in A$ tel que $(a,q'')\in\delta_\mathcal{A}(q_{2k})$, $e_P(s_{2k},a)=s''$ et $e_R((q,q's,h),a)=r''$. Comme $\mathcal{A}$ est nondéterministe $\circlearrowleft_a\notin\delta_\mathcal{A}(q_{2k+1})$. Comme $\mathcal{A}$ est complètement spécifié, $\rightarrow_a \not\circlearrowleft_a\in\delta_\mathcal{A}(q_{2k+1})$. Or la condition $C_1(q_{2k+1},q'_1)$ est vrai donc $\circlearrowleft_a\in\delta_\mathcal{B}(q'_1)$ et donc, d'après la définition de e_R, on a $r''=(q,q',s,h)$. Donc, il y a une arête $(q_{2k+1},(s_{2k},(q,q',s,h)))$ vers $(q'',(s'',(q',q',s,h)))$ dans $G(\mathcal{A},P\times R)$ et une arête de $((q_{2k+1},q'_1),s_{2k})$ vers $((q'',q'_1),s'')$ dans le jeu $G(\mathcal{A}/\mathcal{B},P)$. Montrons qu'il n'y a pas d'arêtes vers $\perp$ depuis $(q_{2k+1},(s_{2k},(q,q',s,h)))$.

- Supposons que $\not\rightarrow_a\in\delta_\mathcal{A}(q_{2k+1})$. Si $\rightarrow_a\in\delta_\mathcal{B}(q'_1)$ alors $\not\rightarrow_a\in\delta_/(q_{2k+1},q'_1)$ Donc $e_P(s_{2k},a)$ n'est pas définit.
- Si $\rightarrow_a,\not\circlearrowleft_a\in\delta_\mathcal{A}(q_{2k+1})$ alors $\rightarrow_a\in\delta_/(q_{2k+1},q'_1)$ et, d'après la condition $C_1(q_{2k+1},q'_1)$, $\circlearrowleft_a\in\delta_\mathcal{B}(q'_1)$. Donc $e_R((q,q',s,h),a)=(q,q',s,h)$. De plus $\not\circlearrowleft_a\in\delta_/(q_{2k+1},q'_1)$ car $C(q_{2k+1},q'_1)$ et $C_1(q_{2k+1},q'_1)$ sont vrais. Donc, puisque $((q_{2k+1},q'_1),s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B},P)$, $e_P(s_{2k},a)$ est définit mais pas égale à s_{2k}.
- Si $\rightarrow_a,\circlearrowleft_a\in\delta_\mathcal{A}(q_{2k+1})$ alors $\circlearrowleft_a\in\delta_/(q_{2k+1},q'_1)\cap\delta_\mathcal{B}(q'_1)$. Donc $e_R((q,q',s,h),a)=(q,q',s,h)$ et puisque $((q_{2k+1},q'_1),s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B},P)$, $e_P(s_{2k},a)=s_{2k}$
- Si $\rightarrow_a,\rightarrow_b,\downdownarrows_{a,b}\in\delta_\mathcal{A}(q_{2k+1})$. Donc $\downdownarrows_{a,b}\in\delta_\mathcal{A}(q_{2k+1},q'_1)$ et, d'après la condition $C_1(q_{2k+1},q'_1)$, $\circlearrowleft_a,\circlearrowleft_b\in\delta_\mathcal{B}(q'_1)$ et, puisque $((q_{2k+1},q'_1),s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B},P)$, $e_P(s_{2k},a)=e_P(s_{2k},b)$.
- Si $\rightarrow_a,\rightarrow_b,\not\downdownarrows_{a,b}\in\delta_\mathcal{A}(q_{2k+1})$. D'après la condition $C_1(q_{2k+1},q'_1)$, $\circlearrowleft_a,\circlearrowleft_b\in\delta_\mathcal{B}(q'_1)$. Donc $\not\downdownarrows_{a,b}\in\delta_\mathcal{A}(q_{2k+1},q'_1)$, donc $e_P(s_{2k},a)\neq s_{2k}$

Soit $(q,(s,(q,q',s,h)))$ une position du jeu $G(\mathcal{A},P\times R)$ accessible par $\sigma_\mathcal{A}$. Supposons que la suite $Seq(q,q',s,h)$ (voir 6.2.3) fini en $(q_{2k+1},q'_1,s_{2k},h_{2k+1})$ et qu'il existe une arête depuis $(q_{2k+1},(s_{2k},(q,q',s,h)))$ vers $(q'',(s'',r))$. Alors il existe $a\in A$ tel que $(a,q'')\in\delta_\mathcal{A}(q_{2k})$, $e_P(s_{2k},a)=s''$ et $e_R((q,q's,h),a)=r''$. Comme $\mathcal{A}$ est nondéterministe $\circlearrowleft_a\notin\delta_\mathcal{A}(q_{2k})$. Comme $\mathcal{A}$ est complètement

spécifié, $\rightarrow_a \in \delta_{\mathcal{A}}(q_{2k})$. De plus la condition $C(q_{2k}, q'_1)$ est fausse donc $\circlearrowleft_a \notin \delta_{\mathcal{B}}(q'_1)$. Donc il existe q'_2 tel que $\rightarrow_a \oslash_a, (a, q'_2) \in \delta_{\mathcal{B}}(q'_1)$ et, d'après la définition de e_R, $r'' = (q'', q'_2, s'', h'')$ avec $h'' = hist(h_{2k+1}, ((q_{2k+1}, q'_1), s_{2k}))$. Donc, il y a une arête $(q_{2k+1}, (s_{2k}, (q, q', s, h)))$ vers $(q'', (s'', (q'', q'_2, s'', h'')))$ dans $G(\mathcal{A}, P \times R)$ et une arête de $((q_{2k+1}, q'_1), s_{2k})$ vers $((q'', q'_2), s'')$ dans le jeu $G(\mathcal{A}/\mathcal{B}, P)$. Montrons qu'il n'y a pas d'arêtes vers $\perp$ depuis $(q_{2k+1}, (s_{2k}, (q, q', s, h)))$.

- Supposons que $\nrightarrow_a \in \delta_{\mathcal{A}}(q_{2k+1})$. Si $\rightarrow_a \in \delta_{\mathcal{B}}(q'_1)$ alors $\nrightarrow_a \in \delta_{/}(q_{2k+1}, q'_1)$ Donc $e_P(s_{2k}, a)$ n'est pas définit.
- Si $\rightarrow_a, \oslash_a \in \delta_{\mathcal{A}}(q_{2k+1})$ alors $\rightarrow_a \in \delta_{/}(q_{2k+1}, q'_1) \cap \delta_{\mathcal{B}}(q'_1)$. Puisque la condition $C(q_{2k+1}, q'_1)$ est fausse, $\oslash_a \in \delta_{\mathcal{B}}(q'_1)$. La condition 6.2.1 permet de s'assurer que $e_R((q, q', s, h), a) \neq (q, q', s, h)$ et, puisque $((q_{2k+1}, q'_1), s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B}, P)$, $e_P(s_{2k}, a)$ est définit.
- Si $\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{A}}(q_{2k+1})$ alors $\circlearrowleft_a \in \delta_{/}(q_{2k+1}, q'_1) \cap \delta_{\mathcal{B}}(q'_1)$. Donc $e_R((q, q', s, h), a) = (q, q', s, h)$ et puisque $((q_{2k+1}, q'_1), s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B}, P)$, $e_P(s_{2k}, a) = s_{2k}$
- Si $\rightarrow_a, \rightarrow_b, \downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_{2k+1})$. Donc $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_{2k+1}, q'_1)$ et ($\downdownarrows_{a,b} \in \delta_{\mathcal{B}}(q'_1)$ ou $\circlearrowleft_a, \circlearrowleft_b \in \delta_{\mathcal{B}}(q'_1)$). Comme $((q_{2k+1}, q'_1), s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B}, P)$, $e_P(s_{2k}, a) = e_P(s_{2k}, b)$. De plus, comme $\mathcal{A}$ et $\mathcal{B}$ sont complet et nondéterministe, il existe q_2, q'_2 tel que $(a, q_2)(b, q_2) \in \delta_{\mathcal{A}}(q_{2k+1})$ et $(a, q'_2)(b, q'_2) \in \delta_{\mathcal{B}}(q'_1)$. Donc $e_R((q, q', s, h), a) = (q_2, q'_2, s_{2k}, h_2) = e_R((q, q', s, h), b)$.
- Si $\rightarrow_a, \rightarrow_b, \nLeftrightarrow_{a,b} \in \delta_{\mathcal{A}}(q_{2k+1})$ alors $\rightarrow_a, \rightarrow_b \in \delta_{\mathcal{B}}(q'_1)$.
  Si $\downdownarrows_{a,b} \in \delta_{\mathcal{B}}(q'_1)$ ou $\circlearrowleft_a, \circlearrowleft_b \in \delta_{\mathcal{B}}(q'_1)$ alors $\nLeftrightarrow_{a,b} \in \delta_{\mathcal{A}}(q_{2k+1}, q'_1)$. Comme $((q_{2k+1}, q'_1), s_{2k})$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{B}, P)$, $e_P(s_{2k}, a) \neq e_P(s_{2k}, b)$.
  Si $\nLeftrightarrow_{a,b}, \oslash_a, \oslash_b \in \delta_{\mathcal{B}}(q'_1)$ Comme $\mathcal{B}$ est complet et complètement spécifié, la propriété 6.2.2 nous assure $e_R((q, q', s, h), a) \neq e_R((q, q', s, h), b)$.

□

6.2.4 Preuve : Si il existe $R \vDash \mathcal{B}$ tel que $P \times R \vDash \mathcal{A}$ et $R \times P$ est bien synchronisé alors $P \vDash \mathcal{A}/_{syn}/\mathcal{B}$

Lemme 6.2.12 Si $R \vDash \mathcal{B}$ et $P \times R \vDash \mathcal{A}$ alors il existe une stratégie gagnante dans $G(\mathcal{A}/\mathcal{B}, P)$ depuis $((q_{\mathcal{A}}{}^0, q_{\mathcal{B}}{}^0), s^0)$.

Preuve. Soit $P = \langle A, S_P, s_P^0, e_P \rangle$ et $\sigma_{\mathcal{A}}$ une stratégie positionnelle gagnante dans $G(\mathcal{A}, P \times R)$. Soit $R = \langle A, S_R, s_R^0, e_R \rangle$ et $\sigma_{\mathcal{B}}$ une stratégie positionnelle gagnante dans $G(\mathcal{B}, R)$.

Nous allons maintenant décrire une stratégie $\sigma : (Q_{/}^{\exists} \times S_P) \times S_R \rightarrow Q_{/}^{\forall} \times S_P$ avec la fonction d'histoire $hist : S_R \times (Q_{/} \times S_P) \rightarrow S_R$ dans $G(\mathcal{A}/\mathcal{B}, P)$.

Supposons que $((q,q'),s) \in Q_{\mathcal{A}}^{\exists} \times Q_{\mathcal{B}}^{\exists} \times S_P$ soit une position du jeu $G(\mathcal{A}/\mathcal{B},P)$ avec l'histoire s' tel que les stratégies dans les positions $(q,(s,s'))$ de $G(\mathcal{A},P\times R)$ et (q',s') de $G(\mathcal{B},R)$ soient définit. Soit $(q_1,(s,s')) = \sigma_{\mathcal{A}}(q,(s,s'))$ et $(q'_1,s') = \sigma_{\mathcal{B}}(q,s')$. Alors la stratégie σ depuis $((q,q'),s)$ est d'aller en $((q_1,q'_1),s)$ avec $hist(s',(q,s)) = s'$.

Supposons que $((q,q'_1),s) \in Q_{\mathcal{A}}^{\exists} \times Q_{\mathcal{B}}^{\forall} \times S_P$ soit une position du jeu $G(\mathcal{A}/\mathcal{B},P)$ avec l'histoire s' tel que la stratégie dans la position $(q,(s,s'))$ de $G(\mathcal{A},P\times R)$ est définit avec $(q_1,(s,s')) = \sigma_{\mathcal{A}}(q,(s,s'))$ et (q'_1,s') est une position accessible par la stratégie $\sigma_{\mathcal{B}}$. Alors la stratégie σ depuis $((q,q'_1),s)$ est d'aller en $((q_1,q'_1),s)$ avec $hist(s',(q,s)) = s'$.

Remarquons d'abord que $D(q_1,q'_1)$ est vrai car les positions $(q_1,(s,s'))$ et (q'_1,s') sont gagnantes et donc σ est bien définit (si $hist$ est bien définit).

Montrons maintenant qu'il n'y a pas d'arête depuis $((q,q'_1),s)$ vers $(q_\perp,s)$ dans $G(\mathcal{A}/\mathcal{B},P)$.

Supposons que $C(q_1,q'_1)$ est vrai. Donc il existe $a \in A$ tel que $\rightarrow_a, \not\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ et $\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$. Or les positions $(q_1,(s,s'))$ et (q'_1,s') sont gagnantes $e(s,a) \neq s$ et $e'(s',a) = s'$. Comme $R\times P$ est bien synchronisé, pour tout $b \in A$, $(e'(s',b) = s'$ ou $e'(s',b)$ n'est pas définit) et soit $e\times e'((s,s'),b)$ n'est pas définit, soit $e\times e'((s,s'),b) = (s,s')$ ou soit $e\times e'((s,s'),b) = e\times e'((s,s'),a)$. Donc, puisque les positions $(q_1,(s,s'))$ et (q'_1,s') sont gagnantes, si $\rightarrow_a, \rightarrow_b \in \delta_{\mathcal{A}}(q_1)$ alors $\circlearrowleft_b \circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$ et si $\not\circlearrowleft_a, \not\circlearrowleft_b \in \delta_{\mathcal{A}}(q_1)$ alors $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$. Donc $C_1(q_1,q'_1)$ est vrai.

Supposons que $C(q_1,q'_1)$ est faux. Donc pour tout $a \in A$ si $\rightarrow_a, \not\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ alors $\circlearrowleft a \notin \delta_{\mathcal{B}}(q'_1)$. Or les positions $(q_1,(s,s'))$ et (q'_1,s') sont gagnantes, donc $e(s,a) \neq s$ et $e'(s',a) \neq s'$. Comme $R\times P$ est bien synchronisé, pour tout $b \in A$, soit $e\times e'((s,s'),b)$ n'est pas définit, soit si $e'(s',a) = e'(s',b)$ alors $e(s,a) = e(s,b)$. Donc, puisque les positions $(q_1,(s,s'))$ et (q'_1,s') sont gagnantes, on peut vérifier que $C_2(q_1,q'_1)$ est vrai.

Soit $((q_1,q'_1),s) \in Q_{/} \times S_P$ une position accessible par σ avec l'histoire s'.

Si il y a une arête depuis $((q_1,q'_1),s)$ vers $((q_2,q'_2),s_2)$ avec $q'_2 \neq q'_1$. Alors il existe $a \in A$ tel que $(a,q_2), \rightarrow_a \ \in \delta_{\mathcal{A}}(q_1)$ et $(a,q'_2) \rightarrow_a \ \in \delta_{\mathcal{B}}(q'_1)$ et $e(s,a) = s_2$. Puisque (q'_1,s') est gagnante, $e'(s',a)$ est définit, disons égale à s'_2. Donc il y a une arête depuis $(q_1,(s,s'))$ vers $(q_2,(s_2,s'_2))$ dans $G(\mathcal{A},P\times R)$ et une arête depuis (q'_1,s') vers (q'_2,s'_2) dans $G(\mathcal{B},R)$. On définit alors $hist(s',((q_1,q'_1),s)) = s'_2$.

Si il y a une arête depuis $((q_1,q'_1),s)$ vers $((q_2,q'_1),s_2)$. De plus il existe $a \in A$ tel que $(a,q_2), \rightarrow_a \ \in \delta_{\mathcal{A}}(q_1)$ et $\rightarrow_a \ \circlearrowleft_a \in \delta_{\mathcal{B}}(q'_1)$ et $e(s,a) = s_2$. Puisque (q'_1,s') est gagnante, $e'(s',a)$ est définit et égale à s'. Donc il y a une arête depuis $(q_1,(s,s'))$ vers $(q_2,(s_2,s'))$ dans $G(\mathcal{A},P\times R)$. On définit alors

$hist(s', ((q_1, q_1'), s)) = s'$.

Il reste à prouver qu'il n'y a pas d'arête vers $\bot$ depuis $((q_1, q_1'), s)$ dans $G(\mathcal{A}/_{syn}/\mathcal{B}, P)$.

- Si $\nrightarrow_a \in \delta_/(q_1, q_1')$ alors $\nrightarrow_a \in \delta_{\mathcal{A}}(q_1)$ et $\rightarrow_a \in \delta_{\mathcal{B}}(q_1')$. Puisque (q_1', s') est gagnante, $e'(s', a)$ est définit et, comme $(q_1, (s, s'))$ est une position gagnante, $e(s, a)$ n'est pas définit.
- Si $\rightarrow_a, \circlearrowleft_a \in \delta_/(q_1, q_1')$ alors $\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$. Puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times R)$, $e_P(s, a) = s$.
- Si $\rightarrow_a, \not\circlearrowleft_a \in \delta_/(q_1, q_1')$ alors $\rightarrow_a \not\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ et $\circlearrowleft_a \in \delta_{\mathcal{B}}(q_1')$. Puisque (q_1', s') est gagnante, $e_R(s', a) = s'$. Alors, puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times R)$, $e_P(s, a) \neq s$
- Si $\rightarrow_a, \rightarrow_b, \downdownarrows_{a,b} \in \delta_/(q_1, q_1')$ alors $\rightarrow_a, \rightarrow_b, \downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$. Alors, puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times R)$, $e_P(s, a) = e_P(s, b)$.
- Si $\rightarrow_a, \rightarrow_b, \not\downdownarrows_{a,b} \in \delta_/(q_1, q_1')$ alors $\rightarrow_a, \rightarrow_b, \not\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$ et $(\circlearrowleft_a, \circlearrowleft_b \in \delta_{\mathcal{B}}(q_1')$ ou $\downdownarrows_{a,b} \in \delta_{\mathcal{B}}(q_1'))$ Puisque (q_1', s') est gagnante, $e_R(s', a) = e_R(s', b)$. Alors, puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times R)$, $e_P(s, a) \neq e_P(s, b)$.

□

6.2.5 Exemple : observation chaînée

Soit $A_O^1, A_O^2, \ldots, A_O^n \subseteq A$ des sous ensembles d'évènements tels que :

$$A_O^1 \subseteq A_O^2 \subseteq \ldots \subseteq A_O^n \subseteq A$$

Dans [Rie03], Stéphane Riedweg montre que le contrôle décentralisé avec évènements inobservables est décidable si, pour tout i, le contrôleur Q_i ne peut observer que les évènements A_O^i, c'est à dire $Q_i \vDash \mathcal{A}(A_O^i)$ d'après l'exemple 3.3.1.

On peut identifier ce cas à un cas particulier de contrôleurs bien synchronisés.

Soit, pour tout $i \in]n]$, $\mathcal{B}(A_O^i)$ un automate avec évènements indiscernables ayant un seul état universel q de parité nulle et la fonction de transition :

$$\delta(q) = \{(a, q) : a \in A\} \cup \{\not\downdownarrows_{x,y}, \not\circlearrowleft_x : x \in A, y \notin A_O^i\}$$

Remarquons que pour tout processus P, il existe un processus Q bisimilaire tel que $Q \vDash \mathcal{B}(A_O^i)$. Notons, pour tout $i \in]n]$, $\mathcal{C}_i$ l'automate $\mathcal{B}(A_O^i) \wedge \mathcal{A}(A_O^i)$.

Lemme 6.2.13 Soit, pour tout $i \in]n]$, des processus $Q_i = \langle A, \Lambda, S_i, s'_{0,i}, e'_i, \lambda'_i \rangle$ tels que $Q_i \vDash \mathcal{C}_i$. Alors, pour tout $i = 2, 3, \ldots n$, $R_{i-1} \times Q_i$ est bien synchronisé avec $R_{i-1} = Q_1 \times \cdots \times Q_{i-1}$.

Preuve : Soit $R_{i-1} = \langle A, \Lambda, S, s_0, e_R, \lambda \rangle$. Soit $(s_i, s) = (s_i, s_1, s_2, \ldots, s_{i-1})$ un état de $Q_i \times R_{i-1}$. Si $e_{R_{i-1}}(s, a) = s$ alors, puisque $Q_k \vDash \mathcal{A}(A^k_O)$ et $A^k_O \subseteq A^i_O$ pour tout $k < i$, on a $e_i(s_i, a) = s_i$. Donc, si $e_i \times e_{R_{i-1}}((s, s_i), b)$ est défini mais n'est pas égale à (s, s_i), alors $e_{R_{i-1}}(s, b) \neq s$. De plus, puisque $Q_i \vDash \mathcal{B}(A^i_O)$, si $e_i(s_i, b)$ et $e_i(s_i, a)$ (avec $a \neq b$) sont définis, alors $e_i(s_i, b) \neq e_i(s_i, a)$. Donc la condition $C_2(\Pi_{R_{i-1}}(s), \Pi_{Q_i}(s_i))$ est vrai. Donc $Q_i \times R_{i-1}$ est bien synchronisé (voir Définition 6.2.2). □

Le corollaire 6.2.5 permet de conclure que le contrôle décentralisé est décidable avec la contrainte proposée dans [Rie03].

6.2.6 Exemple : le pipeline

Considérons le cas du pipeline (voir figure 6.2). Soit $A^1, A^2, \ldots, A^n \subseteq A$ des sous ensembles d'évènements disjoints.

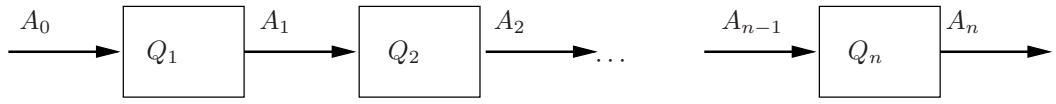


FIG. 6.2 – Pipeline de n contrôleurs

Le contôleur Q_i, soit lit un évènement dans A_i envoyé par le contrôleur Q_{i-1} (ou par l'environement pour Q_1), soit envoie un évènement dans A_{i+1} au contrôleur Q_{i+1}. De plus, il ne voit pas les évènements A_k pour $k < i$ et $k > i + 1$.

Soit, pour tout $i \in]n]$, $\mathcal{D}(A^{i-1}, A^i)$ un automate avec évènements indiscernables ayant un état existentiel q_0 et des états universels $\{q^a_{i-1}, q^a_i : a \in A\}$, tous de parité nulles, et la fonction de transition :

- $\delta(q_0) = \{(\varepsilon, q^a_{i-1}) : a \in A^{i-1}\} \cup \{(\varepsilon, q^a_i) : a \in A^i\}$
- $\delta(q^a_{i-1}) = \{(a, q_0)\} \cup \{\twoheadrightarrow_x : x \in A_i\}$
- $\delta(q^a_i) = \{(a, q_0)\} \cup \{\twoheadrightarrow_x : x \neq a\}$

Notons, pour tout $i \in]n]$, $\mathcal{C}'_i$ l'automate :

$$\mathcal{B}(A^{i-1} \cup A^i) \wedge \mathcal{D}(A^{i-1}, A^i) \wedge \bigwedge_{k<i} \mathcal{A}(A^k) \wedge \bigwedge_{k>i+1} \mathcal{A}(A^k)$$

Voir l'exemple de la partie 6.2.5 pour la défintion de $\mathcal{B}(A^{i-1} \cup A^i)$ et l'exemple 3.3.1 pour la définition de $\mathcal{A}(A^k)$.

Lemme 6.2.14 Soit, pour tout $i \in]n]$, des processus $Q_i = \langle A, \Lambda, S_i, s'_{0,i}, e'_i, \lambda'_i \rangle$ tel que $Q_i \vDash \mathcal{C}'_i$. Alors, pour tout $i = 2, 3, \ldots n$, $Q_i \times R_{i-1}$ est bien synchronisé avec $R_{i-1} = Q_1 \times \cdots \times Q_{i-1}$.

Preuve : Soit $R_{i-1} = \langle A, \Lambda, S, s_0, e_R, \lambda \rangle$. Soit $(s_i, s) = (s_i, s_1, s_2, \ldots, s_{i-1})$ un état de $R_{i-1} \times Q_i$.

Si $e_{R_{i-1}}(s, a) = s$ alors $a \in A^i \cup A^{i+1} \cup \cdots \cup A^n$ car $Q_k \vDash \mathcal{B}(A^{k-1} \cup A^k)$ pour tout $k < i$. Si $a \in A^{i+1} \cup \cdots \cup A^n$ alors soit $e_i(s_i, a) = s_i$ ou $e_i(s_i, a)$ n'est pas défini. Si $a \in A^i$ et $e_i(s_i, a)$ est défini alors, comme $Q_i \vDash \mathcal{D}(A^{i-1}, A^i)$, $D_{Q_i}(s_i) = \{a\}$. Donc la condition $C_1(\Pi_{Q_i}(s_i), \Pi_{R_{i-1}}(s))$ est vrai.

Si $a \in A$ tel que $e_{R_{i-1}}(s, a)$ est défini, on a $e_{R_{i-1}}(s, a) \neq s$ alors $a \in A^0 \cup A^1 \cup \cdots \cup A^{i-1}$. Si $a \in A^0 \cup A^1 \cup \cdots \cup A^{i-2}$ et $e_i(s_i, a)$ est défini alors $e_i(s_i, a) = s_i$. Si $a, b \in A^{i-1}$ et $e_i(s_i, a)$ et $e_i(s_i, b)$ sont définis, alors, puisque $Q_i \vDash \mathcal{B}(A^{i-1} \cup A^i)$, $e_i(s_i, b) \neq e_i(s_i, a)$. Donc la condition $C_2(\Pi_{Q_i}(s_i), \Pi_{R_{i-1}}(s))$ est vrai. Donc $Q_i \times R_{i-1}$ est bien synchronisé. □

Le corollaire 6.2.5 permet de conclure que le contrôle décentralisé est décidable pour le pipeline.

6.3 Quotient d'un automate étendu par un automate simple

Dans cette partie, nous montrons comment définir l'opération de quotient d'un automate étendu par un automate simple (un cas particulier est donc le quotient de deux automates simples).

6.3.1 Résultat principal

Théoreme 6.3.1 Soit $\leq_A$ un ordre totale sur A. Soit $\mathcal{A}$ un automate étendu et $\mathcal{C}$ un automate simple *réduit* par rapport à $\leq_A$ (voir définition 3.4.8). Il existe un automate étendu $\mathcal{A}/\mathcal{C}$ tel que :

$$P \vDash \mathcal{A}/\mathcal{C} \text{ si et seulement si il existe } \widehat{Q} \vDash \mathcal{C} \text{ tel que } P \times Q \vDash \mathcal{A}.$$

(voir Définition 3.4.3 pour la notation $\widehat{Q}$).

On peut en déduire aisément qu'il est possible de faire le quotient d'automate étendu par un automate simple comme l'affirme le corollaire suivant.

Corollaire 6.3.2 Soit $\mathcal{A}$ un automate étendu nondéterministe et $\mathcal{B}$ un automate simple. Il existe un automate étendu $\mathcal{A}/\mathcal{B}$ tel que :

$$P \vDash \mathcal{A}/\mathcal{B} \text{ si et seulement si il existe } Q \vDash \mathcal{B} \text{ tel que } P \times Q \vDash \mathcal{A}.$$

Preuve du corollaire 6.3.2 Soit $\mathcal{C}$ l'automate simple réduit $R(\widehat{\mathcal{B}})$ (voir définitions 3.4.5 et 3.4.12). D'après le théorème 3.4.13, $\widehat{P} \vDash R(\widehat{\mathcal{B}})$ si et seulement si $\widehat{P} \vDash \widehat{\mathcal{B}}$. D'après la proposition 3.4.6, $\widehat{P} \vDash \widehat{\mathcal{B}}$ si et seulement si $P \vDash \mathcal{B}$. D'après le théorème 6.3.1, il suffit donc que l'automate étendu $\mathcal{A}/\mathcal{B}$ soit l'automate $\mathcal{A}/\mathcal{C} = \mathcal{A}/(R(\widehat{\mathcal{B}}))$. □

Nous allons maintenant démontrer le théorème 6.3.1.

6.3.2 Définition de l'automate $\mathcal{A}/\mathcal{C}$

Définition 6.3.3 Soit $\mathcal{A} = \langle A, Q_{\mathcal{A}}, {Q_{\mathcal{A}}}^{\exists}, {Q_{\mathcal{A}}}^{\forall}, {q_{\mathcal{A}}}^{0}, \delta_{\mathcal{A}}, Acc_{\mathcal{A}} \rangle$ un automate étendu nondéterministe à parité. Nous supposons que $\mathcal{A}$ est complet et complètement spécifié (voir définition 3.8.9) et que pour tout $a \in A$ et $q, q_1, q_2 \in Q_{\mathcal{A}}$ on a :

$$\text{si } (\varepsilon, q_1) \in \delta_{\mathcal{A}}(q) \text{ et } (a, q_2) \in \delta_{\mathcal{A}}(q_1) \text{ alors } q_2 \neq q \tag{6.3.1}$$

(Il suffit pour cela de dupliquer certains états de $\mathcal{A}$)

Soit $\mathcal{C} = \langle A, \Pi_A, Q_{\mathcal{C}}, {Q_{\mathcal{C}}}^{\exists}, {Q_{\mathcal{C}}}^{\forall}, {q_{\mathcal{C}}}^{0}, \delta_{\mathcal{C}}, Acc_{\mathcal{C}} \rangle$ un automate simple nondéterministe réduit à parité. On suppose que $\mathcal{C}$ est élagué (voir définition 3.8.10) et que pour tout $q \in Q_{\mathcal{C}}^{\forall}$, $\pi = \{N, L, A_1, \ldots, A_p\} \in \Pi_A$ et $i \in]p]$ (voir notation 3.4.2),

$$\delta_{\mathcal{C}}(q, \pi) \cap (A_i \times Q_{\mathcal{C}}^{\exists}) \neq \emptyset$$

(Il suffit d'ajouter des transitions dans $A_i \times Q_{\mathcal{C}}$ vers un nouvel état gagnant). Il existe donc $q' \in Q^{\forall}$ tel que (voir notation 3.4.7) :

$$\delta_{\mathcal{C}}(q, \pi) \cap A_i \times Q_{\mathcal{C}} = (min_{\leq_A}\{x \in A_i\}, q') \tag{6.3.2}$$

Enfin, on suppose que pour tout $a, b \in A$ et $q, q_1, q_2 \in Q_{\mathcal{C}}$, on a :

$$\text{si } (a, q_1), (b, q_2) \in \delta_{\mathcal{C}}(q, \pi) \text{ alors } q_1 \neq q_2 \tag{6.3.3}$$

(Il suffit pour cela de dupliquer certains états de $\mathcal{C}$)

Soit $\mathcal{A}/\mathcal{C} = \langle A, Q_/, Q_/^{\exists}, Q_/^{\forall}, q_/^0, \delta_/, Acc_/ \rangle$ un automate défini comme suit :

$$Q_/^{\exists} = {Q_{\mathcal{A}}}^{\exists} \times {Q_{\mathcal{C}}}^{\exists}, \quad Q_/^{\forall} = {Q_{\mathcal{A}}}^{\forall} \times {Q_{\mathcal{C}}}^{\forall} \times \Pi_A \quad \text{et} \quad q_/^0 = ({q_{\mathcal{A}}}^0, {q_{\mathcal{C}}}^0)$$

$$Acc_/ = \left\{ (q_0, q_0')(q_1, q_1', \pi_1) \cdots \middle| \quad q_0 q_1 \cdots \in Acc_{\mathcal{A}} \text{ et } q_0' q_1' \cdots \in Acc_{\mathcal{C}} \right\}$$

- Pour tout état existentiel $(q, q') \in Q_/^{\exists}$, $\delta_/(q, q')$ contient $(\epsilon, (q_1, q'_1, \pi))$ avec $\pi = \{N, L, A_1, \ldots, A_p\}$ si :
  - $(\epsilon, q_1) \in \delta_{\mathcal{A}}(q)$
  - $(\epsilon, q'_1) \in \delta_{\mathcal{C}}(q', \pi)$
  - $L = \{a :\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)\}$
  - Si $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ alors $a \notin N$
  - Si $\nrightarrow_a \in \delta_{\mathcal{C}}(q_1, \pi)$ alors $a \in N$
  - Pour tout $i \in]p]$ et $a, b \in A$, on a $a, b \in A_i$ si et seulement si $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$.
- Pour tout état universel $(q_1, q'_1, \pi) \in Q_/^{\forall}$ avec $\pi = \{N, L, A_1, \ldots, A_p\}$, $\delta_/(q_1, q'_1, \pi)$ contient :
  - $(a, (q_2, q'_2))$, si $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$, $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$ et $(min_{\leq_A}\{x \in A_i\}, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$
  - $\rightarrow_a$, si $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$
  - $\nrightarrow_a$, si $\nrightarrow_a \in \delta_{\mathcal{A}}(q_1)$ et $a \notin N$
  - $\circlearrowleft_a$, si $\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$
  - $\downdownarrows_{a,b}$, si $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$

6.3.3 Preuve : Si $P \vDash \mathcal{A}/\mathcal{C}$ alors il existe $\widehat{R} \vDash \mathcal{C}$ tel que $P \times R \vDash \mathcal{A}$.

Supposons qu'il existe une stratégie gagnante $\sigma_/ : (Q_/^{\exists} \times S_P) \times H \rightarrow Q_/^{\forall} \times S_P$ avec l'histoire H dans $G(\mathcal{A}/\mathcal{C}, P)$ pour le processus $P = \langle A, S_P, s_P^0, e_P \rangle$ avec la fonction de mise à jour $hist : H \times (Q_/ \times S_P) \rightarrow H$.

Puisque $\mathcal{C}$ est élagué, pour tout état $q' \in {Q_{\mathcal{C}}}^{\forall}$, il existe un processus $R_{q'} = \langle A, S_{q'}, s_{q'}^0, e_{q'} \rangle$ tel que $R_{q'} \vDash \mathcal{C}_{q'}$ où $\mathcal{C}_{q'}$ est l'automate $\mathcal{C}$ avec l'état initial changé en q'.

Définition 6.3.4 Soit R le processus $\langle A, S_R, s_R^0, e_R \rangle$ tel que

$$S_R = \bigcup_{q' \in {Q_{\mathcal{C}}}^{\forall}} S_{q'} \cup \left\{ (q, q', s, h) \in Q_/^{\exists} \times S_P \times H \middle| \sigma_/((q, q'), s, h) \text{ est défini} \right\}$$

avec $s_R^0 = ({q_{\mathcal{A}}}^0, {q_{\mathcal{C}}}^0, s_P^0, h^0)$ et $e_R : S_R \times A \rightarrow S_R$ est défini comme suit :

1. Pour tout $a \in A$ et $(q, q', s, h) \in S_R$ où $\sigma_/((q, q'), s, h) = ((q_1, q'_1, \pi), s)$ avec $\pi = \{N, L, A_1, \ldots, A_p\}$, on a :
   - Si $a \in N$ alors $e_R((q, q', s, h), a)$ n'est pas défini.
   - Si $a \in L$ alors $e_R((q, q', s, h), a) = (q, q', s, h)$
   - Pour tout i, si $a \in A_i$ et $\rightarrow_a \notin \delta_{\mathcal{A}}(q_1)$ alors $e_R((q, q', s, h), a) = s_{q'_2}^0$ avec $(min_{\leq_A}\{x \in A_i\}, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$ ($s_{q'_2}^0$ est l'état initial de $R_{q'_2}$).

- Pour tout i, si $a \in A_i$ et $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ alors $e_R((q, q', s, h), a) = (q_2, q'_2, s_2, h_2)$ avec :
  - $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$
  - $(min_{\leq_A}\{x \in A_i\}, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$
  - $e_P(s, a) = s_2$
  - $h_2 = hist(hist(h, ((q, q'), s)), ((q_1, q'_1), s))$

2. Pour tout $a \in A$, $q' \in Q_{\mathcal{C}}{}^{\exists}$ et $r \in S_{q'}$ on a $e_R(r, a) = e_{q'}(r, a)$.

Lemme 6.3.5 R est bien défini.

Preuve : Soit $\pi = \{N, L, A_1, \ldots, A_p\}$.

D'après la propriété 6.3.2, pour tout i, il existe q'_2 tel que $(min_{\leq_A}\{x \in A_i\}, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$.

Puisque $\mathcal{A}$ est complet, si $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$, il existe q_2 tel que $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$.

On sait, de plus que la position $((q_1, q'_1, \pi), s)$ est gagnante dans le jeu $G(\mathcal{A}/\mathcal{C}, P)$ car elle est accessible par la stratégie $\sigma_/$. Or $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ implique $\rightarrow_a \in \delta_/(q_1, q'_1, \pi)$, donc il existe s_2 tel que $e_P(s, a) = s_2$. □

Lemme 6.3.6 Pour tout $(q, q', s, h) \in S_R$ où $\sigma_/((q, q'), s, h) = ((q_1, q'_1, \pi), s)$, on a $\Pi_R(q, q', s, h) = \pi$.

Preuve : On a $(\varepsilon, (q_1, q'_1, \pi)) \in \delta_/(q, q')$. Soit $\pi = \{N, L, A_1, \ldots, A_p\}$.

Soit $a, b \in A_i$ et $\rightarrow_a \ \in \delta_{\mathcal{A}}(q_1)$. D'après les conditions permettant les transitions depuis $\delta_/(q, q')$, on a $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$. Or $\mathcal{A}$ est complètement spécifié, donc $\rightarrow_b \ \in \delta_{\mathcal{A}}(q_1)$. Comme $\mathcal{A}$ est nondéterministe, si $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$ et $(b, q_3) \in \delta_{\mathcal{A}}(q_1)$ alors $q_2 = q_3$. De plus, $\downdownarrows_{a,b} \in \delta_/(q_1, q'_1, \pi)$. Puisque que $((q_1, q'_1, \pi), s)$ est une position gagnante, on a $e_P(s, a) = e_P(s, b)$. Donc $e_R((q, q', s, h), a) = e_R((q, q', s, h), b)$

Soit $a \in A_i$ et $b \in A_j$ avec $i \neq j$. La condition 6.3.3 nous assure que $e_R((q, q', s, h), a) \neq e_R((q, q', s, h), b)$.

Soit $a \in A_i$ et $b \in L$. La condition 6.3.1 nous assure que $e_R((q, q', s, h), a) \neq e_R((q, q', s, h), b)$. □

Lemme 6.3.7 Si $P \vDash \mathcal{A}/\mathcal{C}$ alors il existe une stratégie gagnante dans $G(\mathcal{C}, \widehat{R})$ depuis $({q_{\mathcal{C}}}^0, ({q_{\mathcal{A}}}^0, {q_{\mathcal{C}}}^0, s_P^0, h^0))$.

Preuve. Nous allons maintenant décrire une stratégie gagnante $\sigma_{\mathcal{C}}$ dans $G(\mathcal{C}, \widehat{R})$. Supposons que $(q', (q, q', s, h))$ est une position du jeu $G(\mathcal{C}, \widehat{R})$ tel que $\sigma_/(((q, q'), s), h)$ est défini et égale à $((q_1, q'_1, \pi), s)$. Alors $\sigma_{\mathcal{C}}(q', (q, q', s, h)) = (q'_1, (q, q', s, h))$.

Afin de prouver que $\sigma_{\mathcal{C}}$ est gagnante, supposons d'abord qu'il y a une arête depuis $(q'_1, (q, q', s, h))$ vers $(q'_3, (s^0_{q'_2}))$ dans $G(\mathcal{C}, \widehat{R})$. Par définition de e_R, $q'_3 = q'_2$

et $(q'_2, (s^0_{q'_2}))$ est une position gagnante puisque $R_{q'_2} \vDash \mathcal{C}_{q'_2}$.

Supposons que l'on a une arête depuis $(q'_1, (q, q', s, h))$ vers $(q'_2, (q_2, q'_3, s_2, h_2)))$ dans $G(\mathcal{C}, \widehat{R})$. Donc il existe $a \in A$ tel que $(a, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$, et $e_R((q, q', s, h), a) = (q_2, q'_3, s_2, h_2)$. Puisque $\mathcal{C}$ est réduit, d'après la propriété 6.3.2, il existe i tel que $a = min_{\leq_A}\{x \in A_i\}$ et, d'après la définition de e_R, on a $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$, $(min_{\leq_A}\{x \in A_i\}, q'_3) \in \delta_{\mathcal{C}}(q'_1, \pi)$ et $e_P(s, a) = s_2$. Donc, puisque $\mathcal{C}$ est nondéterministe $q'_2 = q'_3$ et, de plus, il y a une arête $((q_1, q'_1, \pi), s)$ vers $((q_2, q'_2), s_2)$ dans $G(\mathcal{A}/\mathcal{C}, P)$. Ainsi, les chemins d'une partie compatible avec $\sigma_{\mathcal{C}}$ dans $G(\mathcal{C}, \widehat{R})$ correspondent aux chemins d'une partie compatible avec $\sigma_/$ dans $G(\mathcal{A}/\mathcal{C}, P)$.

Il reste à prouver qu'il n'y a pas d'arête vers $\perp$ depuis $(q'_1, (q, q', s, h))$.

- Si $\rightarrow_a \in \delta_{\mathcal{C}}(q'_1, \pi)$ alors, puisque $\mathcal{C}$ est réduit, $a \notin N$. Donc $e_R((q, q', s, h), a)$ est défini.
- Si $\nrightarrow_a \in \delta_{\mathcal{C}}(q'_1, \pi)$ alors, d'après les conditions permettant les transitions depuis $\delta_/(q, q')$, on a $a \in N$. Donc $e_R((q, q', s, h), a)$ n'est pas défini.

□

Lemme 6.3.8 Si $P \vDash \mathcal{A}/\mathcal{C}$ alors il existe une stratégie gagnante dans $G(\mathcal{A}, P \times R)$ depuis la position $({q_{\mathcal{A}}}^0, (s^0_P, ({q_{\mathcal{A}}}^0, {q_{\mathcal{C}}}^0, s^0_P, h^0)))$.

Preuve. Nous allons maintenant décrire une stratégie $\sigma_{\mathcal{A}}$ dans $G(\mathcal{A}, P \times R)$. Supposons que $(q, (s, (q, q', s, h)))$ est une position du jeu $G(\mathcal{A}, P \times R)$ tel que la stratégie dans la position $((q, q'), s)$ du jeu $G(\mathcal{A}/\mathcal{C}, P)$ avec l'histoire h est défini et $\sigma_/((q, q'), s, h) = ((q_1, q'_1, \pi), s)$ avec $\pi = \{N, L, A_1, \ldots, A_p\}$. Alors la stratégie $\sigma_{\mathcal{A}}$ depuis $(q, (s, (q, q', s, h)))$ est d'aller vers $(q_1, (s, (q, q', s, h)))$.

Afin de prouver que $\sigma_{\mathcal{A}}$ est gagnante, remarquons d'abord qu'il ne peut avoir d'arête depuis $(q_1, (s, (q, q', s, h)))$ vers une arête de la forme $(q_2, (s_2, s^0_{q'_2}))$. En effet, si c'était le cas, il existerait $a \in A$ tel que $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$ et, puisque $e_R((q, q', s, h), a) = s^0_{q'_2}$, on aurait $\rightarrow_a \notin \delta_{\mathcal{A}}(q_1)$. D'où $\nrightarrow_a \in \delta_{\mathcal{A}}(q_1)$ et alors $\delta_{\mathcal{A}}(q_1) \cap (\{a\} \times Q_{\mathcal{A}}) = \emptyset$ ($\mathcal{A}$ est complètement spécifié), ce qui serait contradictoire avec $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$.

Supposons que l'on une arête depuis $(q_1, (s, (q, q', s, h)))$ vers $(q_2, (s_2, (q_3, q'_2, s_3, h_2)))$. Alors il existe $a \in A$ tel que $(a, q_2) \in \delta_{\mathcal{A}}(q_1)$, $e_P(s, a) = s_2$ et $e_R((q, q', s, h), a) = (q_3, q'_2, s_3, h_2)$. Par définition de e_R, il existe $i \in]p]$ tel que $a \in A_i$, $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$, $(a, q_3) \in \delta_{\mathcal{A}}(q_1)$, $(min_{\leq_A}\{x \in A_i\}, q'_2) \in \delta_{\mathcal{C}}(q'_1, \pi)$ et $e_P(s, a) = s_3$. Donc $s_2 = s_3$ et, puisque $\mathcal{A}$ est nondéterministe, $q_2 = q_3$. De plus ceci permet de conclure qu'il y a une arête depuis $((q_1, q'_1), s)$ vers $((q_2, q'_2), s_2)$ dans

$G(\mathcal{A}/\mathcal{C}, P)$. Ainsi une partie compatible avec $\sigma_{\mathcal{A}}$ dans $G(\mathcal{A}, P \times R)$ correspond à une partie compatible avec $\sigma_/$ dans $G(\mathcal{A}/\mathcal{C}, P)$.

Il reste à prouver qu'il n'y a pas d'arête vers $\perp$ depuis $((q_1, (s, (q, q', s, h))$.

- Supposons que $\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$. D'après les conditions permettant les transitions depuis $\delta_/(q, q')$, $a \in L$. Donc $e_R((q, q', s, h), a) = (q, q', s, h)$. De plus $\circlearrowleft_a, \rightarrow_a \in \delta_/(q_1, q_1', \pi)$. Or $((q_1, q_1', \pi), s)$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{C}, P)$, donc $e_P(s, a) = s$.
- Supposons que $\rightarrow_a, \not\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$. Alors, d'après les conditions permettant les transitions depuis $\delta_/(q, q')$, $a \notin N \cup L$. Donc il existe $i \in]p]$ tel que $a \in A_i$ et $e_R((q, q', s, h), a)$ est défini. De plus $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ implique que $\rightarrow_a \in \delta_/(q_1, q_1', \pi)$ et qu'il existe q_2, q_2', s_2, h_2 tel que $e_P(s, a) = s_2$ et $e_R((q, q', s, h), a) = (q_2, q_2', s_2, h_2)$. La propriété 6.3.1 nous assure que $(q, q', s, h) \neq (q_2, q_2', s_2, h_2)$.
- Supposons que $\downdownarrows_{a,b}, \rightarrow_a, \rightarrow_b \in \delta_{\mathcal{A}}(q_1)$. D'après les conditions permettant les transitions depuis $\delta_/(q, q')$, il existe i tel que $a, b \in A_i$. Donc $e_R((q, q', s, h), a) = e_R((q, q', s, h), b)$. De plus $\downdownarrows_{a,b} \in \delta_/(q_1, q_1', \pi)$. Or $((q_1, q_1', \pi), s)$ est une position gagnante du jeu $G(\mathcal{A}/\mathcal{C}, P)$, donc $e_P(s, a) = e_P(s, b)$.
- Supposons que $\not\downdownarrows_{a,b}, \rightarrow_a, \rightarrow_b \in \delta_{\mathcal{A}}(q_1)$. Puisque $\mathcal{A}$ est complètement spécifié, $\downdownarrows_{a,b} \notin \delta_{\mathcal{A}}(q_1)$. D'après les conditions š permettant les transitions depuis $\delta_/(q, q')$, il existe i, j tel que $i \neq j$ et soit ($a \in A_i$ et $b \in A_j$), soit ($a \in A_i$ et $b \in L$), soit ($a \in L$ et $b \in A_j$). Dans tous les cas, $e_R((q, q', s, h), a) \neq e_R((q, q', s, h), b)$.
- Supposons que $\nrightarrow_a \in \delta_{\mathcal{A}}(q_1)$. Si $a \in N$ alors $e_R((q, q', s, h), a)$ n'est pas défini. Si $a \notin N$ alors, d'après la fonction de transition $\delta_/(q_1, q_1', \pi)$, $\nrightarrow_a \in \delta_/(q_1, q_1', \pi)$ donc $e_P(s, a)$ n'est pas défini.

□

6.3.4 Preuve : Si il existe $\widehat{Q} \vDash \mathcal{C}$ tel que $P \times Q \vDash \mathcal{A}$ alors $P \vDash \mathcal{A}/\mathcal{C}$.

Lemme 6.3.9 Si $\widehat{Q} \vDash \mathcal{C}$ et $P \times Q \vDash \mathcal{A}$ alors il existe une stratégie gagnante dans $G(\mathcal{A}/\mathcal{C}, P)$ depuis $((q_{\mathcal{A}}{}^0, q_{\mathcal{C}}{}^0), s^0)$.

Preuve. Soit $P = \langle A, S_P, s_P^0, e_P \rangle$ et $Q = \langle A, S_Q, s_Q^0, e_Q \rangle$. Soit $\sigma_{\mathcal{A}}$ une stratégie positionnelle gagnante dans $G(\mathcal{A}, P \times Q)$.

Définition 6.3.10 Soit R le processus $\langle A, S_R, s_R^0, e_R \rangle$ tel que :

- $S_R = (S_Q \times Q_{\mathcal{A}} \times S_P) \cup S_Q$ et $s_R^0 = (s_Q^0, q_{\mathcal{A}}^0, s_P^0)$

- Pour tout $s' \in S_Q$, $s \in S_P$ et $q \in Q_{\mathcal{A}}$, $D_R(s', q, s) = D_Q(s')$ et $D_R(s') = D_Q(s')$.
- Pour tout $s' \in S_Q$, $s \in S_P$, $q \in Q_{\mathcal{A}}$ tel que $\sigma_{\mathcal{A}}(q, (s, s')) = (q_1, (s, s'))$ et $a \in D_R(s', q, s)$, on a :
  - $e_R((s', q, s), a) = (s'_2, q_2, s_2)$ si $\rightarrow_a, \not\circlearrowleft_a, (a, q_2) \in \delta_{\mathcal{A}}(q_1)$, $e_Q(s', a) = s'_2$ et $e_P(s, a) = s_2$.
  - $e_R((s', q, s), a) = (s'_2, q, s)$ si $\rightarrow_a, \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ et $e_Q(s', a) = s'_2$.
  - $e_R((s', q, s), a) = s'_2$ si $\not\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ et $e_Q(s', a) = s'_2$
  - $e_R(s', a) = e_Q(s', a)$

Lemme 6.3.11 $\widehat{R} \vDash \mathcal{C}$

Preuve : En effet, puisque $D_R(s', q, s) = D_Q(s')$ et $D_R(s') = D_Q(s')$, R et Q ont le même dépliage et $\mathcal{C}$ est un automate simple donc $\mathcal{C}$ accepte tous les processus en bisimulation avec l'un de ses modèles. □

Soit $\sigma_{\mathcal{C}}$ une stratégie positionnelle gagnante dans $G(\mathcal{C}, \widehat{R})$.

Nous allons maintenant décrire une stratégie $\sigma_/$ dans $G(\mathcal{A}/\mathcal{C}, P)$. Supposons que $((q, q'), s)$ soit une position du jeu $G(\mathcal{A}/\mathcal{C}, P)$ tel que les stratégies dans les positions $(q, (s, s'))$ de $G(\mathcal{A}, P \times Q)$ et $(q', (s', q, s))$ de $G(\mathcal{C}, \widehat{R})$ soient défini. Soit $(q_1, (s, s')) = \sigma_{\mathcal{A}}(q, (s, s'))$ et $(q'_1, (s', q, s)) = \sigma_{\mathcal{C}}(q, (s', q, s))$. Alors la stratégie $\sigma_/$ depuis $((q, q'), s)$ est d'aller en $((q_1, q'_1, \pi), s)$ où $\pi = \Pi_R(s', q, s) = \{N_R(s', q, s), L_R(s', q, s), A_1, \ldots, A_p\}$.

Montrons tout d'abord que $\sigma_/$ est bien défini ; c'est à dire que $(\varepsilon, (q_1, q'_1, \pi)) \in \delta_/(q, q')$.

- On a bien $(\varepsilon, q_1) \in \delta_{\mathcal{A}}(q)$ et $(\varepsilon, q'_1) \in \delta_{\mathcal{C}}(q', \Pi_R(s', q, s))$.
- Si $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$ alors, puisque la position $(q_1, (s, s'))$ du jeu $G(\mathcal{A}, P \times Q)$ est gagnante, $e_Q(s', a)$ est défini donc $e_R(s', a)$, donc $a \notin N_R(s', q, s)$.
- Si $\not\rightarrow_a \in \delta_{\mathcal{C}}(q'_1, \pi)$ alors, puisque la position (q'_1, s') du jeu $G(\mathcal{C}, \widehat{R})$ est gagnante, $e_R(s', a)$ n'est pas défini, donc $a \in N_R(s', q, s)$.
- Si $\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ alors, puisque $(q_1, (s, s'))$ est une position gagnante du jeu $G(\mathcal{A}, P \times Q)$, $e_Q(s', a) = s'$ donc $a \in L_R(s', q, s)$. Si $\circlearrowleft_a \notin \delta_{\mathcal{A}}(q_1)$ alors, $\not\circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)$ et, puisque $(q_1, (s, s'))$ est une position gagnante du jeu $G(\mathcal{A}, P \times Q)$, soit $e_Q(s', a) \neq s'$ ou soit $e_P(s, a) \neq s$. Donc $L_R(s', q, s) = \{a : \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1)\}$.
- Si $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$ alors, puisque $(q_1, (s, s'))$ est une position gagnante du jeu $G(\mathcal{A}, P \times Q)$, $e_P(s', a) = e_P(s', b)$ et $e_Q(s', a) = e_Q(s', b)$. De plus, comme $\mathcal{A}$ est nondéterministe et complet, si $(a, q_2)(b, q_3) \in \delta_{\mathcal{A}}(q_1)$ alors $q_2 = q_3$. Donc il existe i tel que $a, b \in A_i$. Si $\downdownarrows_{a,b} \notin \delta_{\mathcal{A}}(q_1)$ alors $\not\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$ et, puisque $(q_1, (s, s'))$ est une position gagnante du jeu

$G(\mathcal{A}, P \times Q)$, $e_P(s', a) \neq e_P(s', b)$ ou $e_Q(s', a) \neq e_Q(s', b)$. Donc pour tout i, $\{a, b\} \not\subset A_i$.

Afin de prouver que $\sigma_/$ est gagnante, supposons que l'on a une arête depuis $((q_1, q_1', \pi), s)$ vers $((q_2, q_2'), s_2)$. Donc il existe $i \in]p]$ tel que $a \in A_i$, $e_P(s, a) = s_2$, $\rightarrow_a, (a, q_2) \in \delta_{\mathcal{A}}(q_1)$ et $(min_{\leq_A}\{x \in A_i\}, q_2') \in \delta_{\mathcal{C}}(q_1', \pi)$.

Puisque la position $(q_1, (s, s'))$ de $G(\mathcal{A}, P \times Q)$ est gagnante, $e_Q(s', a)$ est défini, disons s_2' et donc il y a une arête depuis $(q_1, (s, s'))$ vers $(q_2, (s_2, s_2'))$ dans $G(\mathcal{A}, P \times Q)$.

De plus, puisque $a \in A_i$, pour tout $b \in A_i$, on a $\downdownarrows_{a,b} \in \delta_{\mathcal{A}}(q_1)$. Comme $\mathcal{A}$ est nondéterministe et complet, pour tout $b \in A_i$, $(b, q_2) \in \delta_{\mathcal{A}}(q_1)$ et donc $(min_{\leq_A}\{x \in A_i\}, q_2) \in \delta_{\mathcal{A}}(q_1)$.

Puisque la position $(q_1, (s, s'))$ de $G(\mathcal{A}, P \times Q)$ est gagnante, on a pour tout $b \in A_i$, $e_P(s, b) = e_P(s, a) = s_2$ et donc $e_P(s, min_{\leq_A}\{x \in A_i\}) = s_2'$.

Donc il y a une arête depuis $(q_1', (s', q, s))$ vers $(q_2', (s_2', q_2, s_2))$ dans $G(\mathcal{C}, \widehat{R})$.

Il reste à prouver qu'il n'y a pas d'arête vers $\perp$ depuis $((q_1, q_1', \pi), s)$ dans $G(\mathcal{A}, P \times R)$.

Si $\rightarrow_a \in \delta_/(q_1, q_1, \pi)$ alors $\rightarrow_a \in \delta_{\mathcal{A}}(q_1)$. Puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times Q)$, $e_P(s, a)$ est défini.

Si $\twoheadrightarrow_a \in \delta_/(q_1, q_1, \pi)$ alors $\twoheadrightarrow_a \in \delta_{\mathcal{A}}(q_1)$ et $a \notin N_R(s', q, s)$ donc $e_R(s', q, s)$ est défini. Donc, puisque $(q_1, (s, s'))$ est une position gagnante dans $G(\mathcal{A}, P \times Q)$, $e_P(s, a)$ n'est pas défini.

Pour les propriétés $\downdownarrows_{a,b} \in \delta_/(q_1, q_1, \pi)$ et $\circlearrowleft_a \in \delta_/(q_1, q_1, \pi)$ les preuves sont similaires. □

6.4 Conclusion

Nous avons présenté dans ce chapitre deux types de restrictions (parties 6.1 et 6.2) permettant au problème de contrôle décentralisé avec événements indiscernables et inobservables, d'être décidable. (La partie 6.3 permet seulement de conclure à la décidabilité si un seul des contrôleurs a un automate étendu pour spécification.)

Le premier type de restrictions concerne les spécifications des contrôleurs et du système supervisé.

Les contrôleurs doivent satisfaire des spécifications définies par des automates étendus déterministes. Ces spécifications permettent de définir les contraintes d'observabilité classique (voir exemple 3.3.1) mais aussi des contraintes

dynamique comme l'exemple 3.3.4 ou encore des contraintes dépendant de comportements infinis.

Les spécifications du système supervisé sont plus restreintes, ce sont des automates étendus appelés "coniques". Si on ne spécifie rien sur les comportements infinis, les automates coniques sont la conjonction de deux spécifications. Celle d'un langage requis (défini par l'automate minimal) et celle spécifiant un langage régulier clos par préfixes.

La deuxième restriction concerne le type de transitions du système supervisé. Celles-ci sont conditionnées par une certaine synchronisation entre les transitions des processus formant le système supervisé.

Dans le produit $P \times Q$ de deux processus, il peut avoir deux possibilités. Soit les événements que distinguent P (ceux qui ne sont pas indiscernables) sont aussi distingués par Q, soit, si Q n'observe pas un événement a observable par P, alors tous les événements dans le même cas (observables par P mais pas par Q) doivent être indiscernables.

Avec cette restriction, il est possible de définir une opération de quotient de deux automates étendus qui permet de trouver les solutions du problème **DCPIO** qui ont cette synchronisation. Ce cas décidable permet d'exprimer le cas d'un contrôle de type "pipeline" (voir exemple 6.2.6).

On peut penser qu'il est possible de regrouper ces deux restrictions en une seule. Plus précisément, on peut envisager de construire un automate "quotient" de deux automates tel que, depuis tout état, soit les modèles possibles sont bien synchronisés, ou soit les comportements possibles sont ceux définis par un automate conique. Ces restrictions semblent cependant très contraignantes. Il pourrait être intéressant de comparer ces restrictions avec celles envisagées dans d'autres contexes où le probème a été étudié (voir par exemple [GBZ04, MT01, MT02, MW03, PnRo90]).

Bibliographie

[AN01] A. Arnold and D. Niwinski. *Rudiments of $\mu-$calculus*. Volume 146 of Studies in Logic and the Foundations of Mathematics. Elsevier, 2001.

[A99] A. Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. RAIRO-Theorical Informatics and Applications 33(4/5) : 329-340, 1999.

[AVW03] A. Arnold, A. Vincent and I. Walukiewicz. Games for synthesis of controllers with partial observation. Theoretical Computer Science vol. 303(1) (2003) pp. 7-34.

[ABW03] A. Arnold, X. Briand and I. Walukiewicz, Synthesis of Decentralized Controllers : Decidable and Undecidable Cases. Proc. ATPN - Workshop on Discrete Event Systems Control, 24th International Conference on Application Theory of Petri Nets (ATPN 2003), Eindhoven, The Netherlands, pp.57-74, June 2003.

[BC05] M. Bojanczyk and T. Colcombet. Tree-walking automata do not recognize all regular languages. STOC, page 234-243, 2005.

[BSV03] H. Bjorklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In Symposium on Theoretical Aspects of Computer Science, STACS 2003, volume 2607 of LNCS, pages 663-674. Springer, 2003.

[Buc83] J.R. Büchi. State strategies for games in $F_{\sigma,\delta} \cap G_{\delta,\sigma}$. Journal of Symbolic Logic, 48 :1171-1198, 1983.

[CE81] E.M. Clarke and E.A. Emerson. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. In Proc. Workshop on Logics of Programs, Lecture Notes in Computer Science, 131 :52-71, Springer, Berlin, 1981.

[CDFV88] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. IEEE Trans. Automatic Control 33 (1988), 249-260.

[CL91] E. Chen and S. Lafortune. Dealing with blocking in supervisory control of discrete-event systems. IEEE Trans. Automatic Control, 36 :724-735, 1991.

[CL99] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, September 1999.

[EH86] E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited : On branching versus linear time temporal logic. Journal of the ACM, vol. 33, pp. 151-178, 1986.

[EJ88] E. A. Emerson, C. S. Jutla. The Complexity of Tree Automata and Logics of Programs (Extended Abstract) FOCS : 328-337, 1988

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In Proc. FOCS 91, 1991.

[ES89] R.S. Streett and E.A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. Information and Computation 81, no. 3, 249-264, 1989.

[GBZ04] Paul Gastin, Benjamin Lerman and Marc Zeitoun. Distributed Games with Causal Memory Are Decidable for Series-Parallel Systems. FSTTCS, pages 275-286, 2004.

[GH82] Y. Gurevich and L. Harrington. Trees, automata and games. In 14th ACM Symp. on Theory of Computations, pages 60-65, 1982.

[GR87] C. H. Golaszewski and P. J. Ramadge. Control of discrete event processes with forced events. Proc. 26th IEEE Conf. Decision and Control, Los Angeles, CA, 247-251, 1987.

[H90] M. Heymann. Concurrency and discrete event control. IEEE Control Systems Magazine. 10 :103-112, 1990.

[JW96] D. Janin, I. Walukiewicz. On the expressive completeness of the modal mu-calculus w.r.t. monadic second order logic. CONCUR'96, Lecture Notes in Comput. Sci., 1996, 1119 , pp. 263-277

[JPZ06] M. Jurdzinski., M. Paterson and U. Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, January 2006.

[KG95] R. Kumar and V.K.Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Pub.,1995.

[Koz83] D. Kozen. Results on the Propositional μ-calculus. TCS, 27 : 333-354, 1983.

[KS95] R. Kumar and M.A. Shayman.

[KS95] R. Kumar and M. A. Shayman. Supervisory Control of Nondeterministic Systems with Driven Events via Prioritized Synchronization and Trajectory Models, 1995.

[LY02] A General Architecture for Decentralized Supervisory Control of Discrete-Event Systems, invited to "WODES2000" issue of Journal of Discrete-Event Dynamical Systems : Theory and Applications, 2002.

[LW88] F.Lin and W.M. Wonham. On observability of discrete-event systems. Information Sciences, 44(3), pp. 173-198, 1988.

[MS87] D. Muller and P. Schupp. Alternating automata on infinite trees. Theoretical Computer Science, 54(2-3) :267-276, October 1987.

[MS95] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata : New results and new proofs of the theorems by Rabin, McNaughton and Safra. Theor. Comp. Sci., 141 :69-107, 1995.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems : Specification*. Springer-Verlag, 1992.

[MaSe05] Y. Matiyasevich and G. Sénizergues. Decision problems for semi-Thue systems with a few rules. Theor. Comp. Sci., 330(1) :145-169, 2005.

[MT01] P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specications. In ICALP 01, volume 2076 of LNCS. Springer, 2001.

[MT02] P. Madhusudan and P. S. Thiagarajan. A decidable class of asynchronous distributed controllers. In CONCUR 02, volume 2421 of LNCS. Springer, 2002.

[MW03] S. Mohalik and I. Walukiewicz. Distributed games. In FSTTCS 03, volume 2914 of LNCS, pages 338-351. Springer, 2003.

[N97] D. Niwinski. Fixed Point Characterization of Infinite Behavior of Finite-State Systems. Theor. Comput. Sci. 189(1-2) : 1-69, 1997.

[P05] G. Point. The Synthesis Toolbox - From modal automata to controller synthesis. Technical report RR-1342-05, LaBRI, 2005.

[PnRo90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthetize. In 31th IEEE Symp. FOCS, pages 746-757, 1990.

[PR05b] S. Pinchinat and S. Riedweg. On the Architectures in Decentralized Supervisory Control. 44th IEEE Conference on Decision and Control and European Control Conference, Seville, Spain, 12-15 December 2005.

[PR05] S. Pinchinat and S. Riedweg. You Can Always Compute Maximally Permissive Controllers Under Partial Observation When They Exist. 24th American Control Conference, Portland, Oregon, 8-10 June 2005.

[PAP94] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[R89] P.J.Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Buchi automata. IEEE Trans. on Automatic Control, 34(1) :10-19, January, 1989.

[RW87] P.Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. SIAM J. Control Optim. 25, 1, 206-230, 1987.

[RW88] W.M. Wonham and P.J. Ramadge. Modular supervisory control of discrete event systems. Mathematics of Control, Signals and Systems, 1(1) :13–30, 1988.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems, Proceedings of the IEEE, January, 1989.

[RuWo92] K. Rudie and W.M. Wonham. Think Globally, Act Locally : Decentralized Supervisory Control. IEEE Transactions on Automatic Control , Volume 37, Number 11, pp. 1692-1708, 1992.

[Rie03] S. Riedweg. Logiques pour le contrôle d'automatismes discrets thèse de l'université de Rennes 1, Decembre 2003, 122 pages.

[TW94] J.G. Thistle and W.M. Wonham. Control of Infinite Behaviour of Finite Automata. SIAM J. Control Optim., 32(4) : , July 1994.

[TW94b] J.G. Thistle and W.M. Wonham. Supervision of infinite behaviour of discrete-event systems. SIAM Journal on Control and Optimization, 32(4) :1098-1113, 1994.

[Tho96] W. Thomas. On the synthesis of strategies in infinite games. In STACS'95, volume 900, pages 1-13, Springer, 1995.

[Tho97] W.Thomas.Languages, automata, and logic. In G.Rosenberg and A.Salomaa, editors, *Hanbook of formal languages*, volume 3. Springer-Verlag, 1997.

[V98] M.Y. Vardi. Reasoning about The Past with Two-Way Automata. ICALP 1998 : 628-641

[VJ00] J. Vöge, M. Jurdzinski. A Discrete Strategy Improvement Algorithm for Solving Parity Games. CAV : 202-215, 2000.

[Wal01] I. Walukiewicz. Automata and logic. Notes from EEF Summer School'01, 2001.