ATP : une algebre pour la specification et l'analyse des systemes temps reel

Xavier Nicollin

THESE

présentée par
NICOLLIN Xavier

pour obtenir le grade de DOCTEUR
de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 23 novembre 1988)

(Spécialité : Informatique)

ATP : une algèbre pour la spécification et l'analyse des systèmes temps réel

---

Date de soutenance : 18 mai 1992

Composition du jury :	Président	J. Mossière
	Rapporteurs	G. Berry
		P. Cousot
	Examinateurs	A. Pnueli
		J. Sifakis

Thèse préparée au sein du Laboratoire de Génie Informatique

Remerciements

Je tiens à remercier

Monsieur Jacques Mossière, professeur à l'Institut National Polytechnique de Grenoble, pour m'avoir fait l'honneur de présider le jury de cette thèse, ainsi que pour son aide précieuse dans les ultimes phases administratives ;

Messieurs Gérard Berry, maître de recherches à l'École Nationale Supérieure des Mines de Paris, et Patrick Cousot, professeur à l'École Normale Supérieure de Paris, pour avoir accepté de juger ce travail et pour l'intérêt qu'ils lui ont accordé ;

Monsieur Amir Pnueli, professeur au Weizmann Institute de Rehovot, Israël, qui m'a fait le grand honneur d'être membre du jury ;

Monsieur Joseph Sifakis, directeur de recherches au C.N.R.S., qui est le directeur de cette thèse. C'est sous son impulsion que ce travail a commencé, et c'est grâce à nos nombreuses discussions, souvent passionnées mais toujours fructueuses, qu'il a pu aboutir. Je lui suis profondément reconnaissant de m'avoir ainsi donné le goût de la recherche ;

Tous les membres du projet SPECTRE, qui contribuent à l'établissement d'un environnement de travail hors du commun. Je suis particulièrement reconnaissant à Claire Loiseaux et Sergio Yovine, qui ont patiemment supporté l'atmosphère enfumée de notre bureau durant les derniers mois de ce travail.

Et Florence, pour tout.

Table des matières

Liste des figures

Introduction

Les systèmes temps réel : description et vérification

Un *système temps réel* est un ensemble de composantes qui interagissent de façon permanente avec leur environnement, afin de garantir un service satisfaisant des *contraintes temporelles*. On trouve par exemple de tels systèmes dans le domaine des transports (avionique, spatial, ...) et dans celui du contrôle de processus industriels (centrales nucléaires, ...)

Ce sont des systèmes hautement réactifs, dont les décisions peuvent avoir des conséquences *immédiates et irréversibles* sur leur environnement. En particulier, toute défaillance même intermittente peut entraîner des dégâts d'un coût prohibitif. Il s'ensuit qu'il est essentiel, avant de les mettre en service, de vérifier qu'il fonctionnent correctement *dans toutes les situations*. Pour que cette vérification fournisse des résultats fiables, il est indispensable de l'effectuer sur l'ensemble des comportements possibles du système.

Contraintes temporelles, temps multiforme

Les *contraintes temporelles* sont définies comme des relations entre occurrences d'événements, en particulier entre les actions du système et le comportement observé de l'environnement. Afin de satisfaire ces contraintes, le système a parfois la possibilité d'agir sur son environnement pour modifier le comportement de celui-ci. Il arrive cependant que certains événements externes soient incontrôlables. En particulier, l'écoulement du temps est toujours inéluctable. Il peut en aller de même pour d'autres événements externes.

Considérons par exemple un contrôleur de passage à niveau, qui ne peut pas ralentir ou arrêter un train. Le train envoie des impulsions tous les dix mètres (via des capteurs situés sur la voie). L'événement "le train a franchi dix mètres" est alors aussi inéluctable pour le contrôleur que le passage du temps. Ces deux types d'événements sont de même nature, bien qu'indépendants.

Le contrôleur doit satisfaire des contraintes, d'une part relatives au passage du temps ("la barrière est relevée au plus vingt secondes après le passage du train"), d'autre part relatives à l'avancée du train ("la barrière est baissée avant que le train soit à trois cents mètres du passage à niveau"). La description du système doit donc garantir que certaines actions ont lieu dans des "délais" précis. Ces délais peuvent être exprimés en unités de temps (par exemple les secondes), mais aussi en unités de distance (ici "dix mètres"). Les secondes et les dizaines de mètres sont pour le contrôleur des *événements temporels*.

Pour décrire aisément un tel système, il est nécessaire de disposer de langages de haut niveau qui prennent en compte cette notion de *temps multiforme*, c'est-à-dire qui permettent de spécifier

des délais de diverses natures. La vérification formelle exige de plus que la *sémantique formelle* de ces langages soit parfaitement définie, sans ambiguïté ni incohérence.

Les langages existants

Parmi les premiers langages permettant de décrire des systèmes temps réel, on peut citer ADA [Ada] et ESTELLE [ISO88a], qui comportent des constructions exprimant des délais. Ceux-ci sont définis relativement à une unique base de temps globale ; on ne dispose donc pas de notion de temps multiforme.

Ces langages présentent un défaut majeur : la sémantique des constructions temporelles est mal définie, et surtout la durée des autres instructions n'est pas précisée. Il est ainsi impossible de prédire le comportement d'un système et donc de le vérifier complètement. En effet, si, dans les systèmes classiques, la durée des instructions n'influe pas sur leur enchaînement, il n'en va pas de même dans un système temps réel : l'expiration d'un délai peut avoir des conséquences différentes selon l'état du système, qui ne dépend pas seulement de l'histoire de ses entrées, mais aussi des temps d'exécution.

Récemment, une nouvelle classe de langages dits *synchrones* a apporté une solution à ces problèmes sémantiques. Les principaux représentants en sont ESTEREL [BC84], SIGNAL [LBBG85], LUSTRE [CHPP87], les STATECHARTS [Har87] et ARGOS [Mar90]. Ils sont basés sur l'*hypothèse de synchronisme*, qui stipule que toutes les actions d'un système s'exécutent en un temps nul. La vitesse d'exécution du système est alors uniquement dictée par la fréquence de ses entrées. Cette hypothèse est la seule qui permette de résoudre un grand nombre de problèmes sémantiques. Toutes les réactions du système étant instantanées, on est assuré que la réponse à un événement sera terminée avant l'occurrence du suivant. Dans ces langages, le temps n'est qu'une entrée parmi d'autres. Ils adoptent donc implicitement une notion de temps multiforme.

L'hypothèse de synchronisme permet d'obtenir un modèle simple du comportement d'un programme. Il s'agit en général d'un *automate* ou *système de transitions étiquetées*. Les états de l'automate représentent les états du système ; une transition décrit le changement d'état provoqué par le ou les événements (entrées ou actions du système) désignés par son étiquette.

Lors de l'implantation sur une machine donnée, il faut bien sûr vérifier, en faisant des hypothèses sur le rythme des entrées, que le système a toujours le temps de réagir entre deux occurrences d'entrées quelconques. Si c'est le cas, toutes les propriétés prouvées avec l'hypothèse de synchronisme restent vraies lors de l'exécution.

Méthodes de vérification

La vérification formelle d'un programme parallèle consiste à garantir qu'il respecte sa *spécification* ; celle-ci définit les propriétés attendues du système. On distingue deux styles de description des spécifications :

- Les propriétés sont décrites dans le même formalisme que le programme. On parle alors de *spécifications comportementales*. Vérifier le programme revient à le comparer à sa spécification via une relation de préordre ou d'équivalence. En général, cette relation prend en compte un critère d'abstraction, qui permet de comparer une description détaillée (celle du système) et une description plus abstraite (celle de la spécification).

- Les propriétés sont décrites par des formules d'une logique ; la spécification est alors *déclarative*. Elle définit l'ensemble des programmes qui la satisfont. Vérifier le programme consiste dans ce cas à montrer qu'il appartient à cet ensemble. L'utilisation de *logiques temporelles* en tant que formalisme de spécification a été proposée par A. Pnueli [Pnu77]. Elles s'avèrent les mieux appropriées car elles permettent de raisonner globalement sur l'évolution du programme dans le temps. L'étude des logiques temporelles est un domaine de recherche particulièrement riche [MP81, EC82, OL82, Lam83, MW84, BKP86, CES86, Pnu86, MP89].

Ces deux styles de spécification sont complémentaires et nécessaires pour une expression directe et naturelle des propriétés.

Vérifier un programme, c'est donc établir une relation (d'équivalence ou de préordre d'un côté, de satisfaction de l'autre) entre le programme et sa spécification. Deux approches sont possibles pour résoudre ce problème :

- La première, appelée *axiomatique*, consiste à définir un système déductif décrivant la relation entre programmes et spécifications. La vérification devient alors une preuve dans ce système d'axiomes et de règles d'inférence. Dans le cas de spécifications comportementales, il faut pour cela axiomatiser complètement l'équivalence choisie (ou le préordre) entre les programmes. Si les spécifications sont logiques, il faut définir la relation de satisfaction par induction sur la structure des programmes. L'inconvénient principal de cette méthode réside dans le fait que la construction de preuve est la plupart du temps non automatisable, surtout dans le cas de spécifications déclaratives.

- La seconde approche consiste à *raisonner sur les modèles*. Nous nous situons ici uniquement dans le cas où la sémantique du langage de programmation associe au programme un système de transitions.

  Si les spécifications sont comportementales, on doit comparer deux modèles modulo une équivalence ou un préordre. Cette méthode est complètement automatisable pour un grand nombre de relations. Parmi les outils qui la mettent en œuvre, on peut citer les logiciels ALDÉBARAN [Fer88, Mou92], AUTO [Ver87] et le *Concurrency Workbench* [CPS89].

  Lorsqu'on a affaire à des spécifications logiques, il faut prouver que le modèle du programme est élément de l'ensemble des modèles de la spécification. On utilise alors des techniques de *model checking*, qui consistent à déterminer l'ensemble des états du modèle du système satisfaisant une formule [CE81, QS81, CES86, EL86, CS91]. La méthode fait intervenir des calculs de points fixes de fonctionnelles sur des ensembles d'états. L'intérêt du model checking réside dans le fait qu'il est complètement automatisable lorsque les points fixes sont calculables. Plusieurs outils performants de vérification appliquent cette méthode, parmi lesquels on peut citer les systèmes EMC [CES86] et XESAR [RRSV87].

La limitation essentielle des techniques de vérification par les modèles est due à la taille du modèle du système, qui croît exponentiellement en fonction du nombre de composantes parallèles. Diverses stratégies sont envisageables pour remédier à ce problème. On peut par exemple essayer de réduire le nombre d'états du modèle pendant ou après sa construction [Fer88, dSV89, GS90, CLM89, VT91]. Il est également envisageable de ne générer le modèle qu'au cours du processus de vérification, sans en mémoriser la totalité. On désigne cette approche sous l'appellation de vérification "à la volée" [JJ89, BFH90, CVWY90, JJ91, Mou92].

Une dernière méthode consiste, non pas à diminuer le nombre d'états, mais à adopter une représentation plus compacte du système de transitions, en manipulant des ensembles d'états sous la

forme de prédicats. La génération du modèle et la vérification font alors intervenir des transformateurs de prédicats et des procédures de décision de validité de prédicats.

Cette méthode théorique est à présent implémentable, essentiellement grâce aux *graphes binaires de décision* [Bry86], qui permettent de représenter des fonctions booléennes et d'effectuer des calculs sur ces fonctions de manière efficace. Elle peut être utilisée pour générer un modèle d'exécution du système destiné à être implanté, comme c'est le cas dans le compilateur LUSTRE-V3 [Ray91]. On peut également appliquer des techniques symboliques pour procéder à la vérification sans générer de système de transitions [CBM89, BCD+90, Rat92]. Pour cela, on calcule les ensembles d'états successeurs ou prédécesseurs d'un autre ensemble d'états au moyen de transformateurs de prédicats. Dans le cas de spécifications logiques, on parle alors de *model checking symbolique*.

En ce qui concerne les systèmes temps réel, la taille du modèle dépend également des valeurs des délais apparaissant dans la description. Par exemple, si le rôle d'une composante est de contrôler qu'un événement survient avant 500 millisecondes, le système de transitions doit comporter 500 arcs étiquetés par l'événement "milliseconde", car il faut pouvoir distinguer la $500^{\text{ème}}$ occurrence de toutes les précédentes. Il est par conséquent souhaitable de développer des méthodes de réduction de modèles ou de vérification symbolique spécifiques aux systèmes comportant des constructions temporelles, pour remédier à cette cause supplémentaire d'explosion.

Objectifs du travail

La mise au point de techniques de vérification adaptées aux systèmes temps réel nécessite une analyse approfondie de leurs modèles. Il faut pour cela étudier les opérateurs des langages, leur sémantique et leurs propriétés qui rendent compte de la structure des modèles.

Pour effectuer une telle analyse, on ne considère pas un langage de programmation dans toute sa généralité, mais on se restreint aux constructions qui sont en rapport avec le problème étudié. On se situe alors dans un cadre algébrique qui permet, d'une part, de développer plus facilement des systèmes de déduction et, d'autre part, d'associer plus simplement un modèle à un élément du langage (un terme de l'algèbre).

En définissant l'algèbre CCS [Mil80, Mil89], R. Milner a le premier isolé les concepts de base des *algèbres de processus communicants* et montré leur intérêt en ce qui concerne l'étude de la sémantique des systèmes parallèles. Cette approche est depuis lors fertile en développements ; les deux autres algèbres de processus les plus célèbres sont ACP [BK84] et TCSP [BHR84], cette dernière constituant la version algébrique du langage CSP [Hoa78].

Dans le cas des algèbres de processus, on utilise le terme de *spécification* de préférence à celui de *description* ou d'*implémentation*, car le niveau d'abstraction est plus élevé que celui des langages de programmation.

Les trois algèbres citées ci-dessus adoptent un principe de composition parallèle *asynchrone* : les processus peuvent évoluer de manière indépendante lorsqu'aucune communication n'est imposée. On peut à l'inverse définir des algèbres pour lesquelles la composition parallèle est *synchrone* : toute évolution du système résulte de l'évolution simultanée de ses composantes. On dit alors que les processus sont fortement synchronisés. Le représentant le plus connu des algèbres synchrones est SCCS [Mil83]. Il est possible de désynchroniser des processus dans une algèbre synchrone [Mil83], et inversement de les synchroniser fortement dans un cadre asynchrone si on

dispose d'un opérateur de *pilotage*, comme dans MEIJE [AB84].

Nous souhaitons définir une algèbre de processus comportant des opérateurs temporels. Ce travail prend sa source dans une proposition de J.-L. Richier, J. Sifakis et J. Voiron [RSV87], qui définissent un opérateur de composition parallèle mixte, c'est-à-dire qui se comporte comme une composition synchrone ou asynchrone selon les actions exécutées par les processus. Celles-ci sont partitionnées en *actions synchrones* et *actions asynchrones* ; l'exécution d'une action synchrone requiert la participation de toutes les composantes du système, alors qu'une action asynchrone peut être interne à un processus ou résulter d'une communication locale entre deux ou plusieurs processus. Les actions synchrones sont destinées à modéliser les événements temporels, qui doivent être perçus simultanément par toutes les composantes. Cette idée conduit dans [RSV87] à la définition d'une algèbre appelée ATP (*Algebra of Timed Processes*), dans laquelle n'apparaît qu'une seule action synchrone ; le temps n'est donc pas multiforme.

L'objectif initial d'ATP était d'apporter une sémantique formelle simple aux constructions temporelles du langage ESTELLE/R, variante d'ESTELLE utilisée dans l'outil XESAR de vérification de protocoles [RRSV87]. Si les idées générales ont pu être appliquées dans le système XESAR, il restait de nombreux problèmes à résoudre pour obtenir un ensemble cohérent, permettant d'étudier en détail les opérateurs temporels et la structure des modèles. Une tentative de clarification [Nic88] a permis d'isoler certains des principes à suivre pour construire une algèbre de processus temporisés, en particulier la nécessité d'assurer le déterminisme des événements temporels. Il s'agit ici d'appliquer ces principes et d'en dégager d'autres pour définir complètement, et de manière simple, une telle algèbre, afin de mettre en évidence les propriétés des opérateurs et des modèles.

Une fois résolus les problèmes de sémantique, on peut s'intéresser à la vérification des systèmes temps réel. L'analyse de la structure des modèles doit en effet permettre de dégager des méthodes de vérification adaptées à leurs spécificités.

Organisation du document

La **première partie** est intitulée *spécification* ; elle est dédiée à la construction et à l'étude d'une algèbre de processus temporisés et de ses extensions.

Le **chapitre 1** a pour but d'introduire les concepts de base des algèbres de processus. Nous présentons une algèbre de processus standard AUP, inspirée de CCS et ACP. Elle comporte tous les opérateurs non temporels utilisés par la suite. Nous explicitons en les appliquant à AUP les notions de *sémantique opérationnelle*, d'*équivalence forte* et d'*axiomatisation*.

Le **chapitre 2** est consacré à l'introduction du temps dans AUP au moyen d'opérateurs temporels. On obtient ainsi une algèbre de processus temporisés. Nous lui donnons le nom d'ATP, car elle emprunte à celle définie dans [RSV87] le principe de l'opérateur de composition parallèle mixte, ainsi que l'absence de temps multiforme. Elle s'en distingue par contre par la quasi-totalité des choix sémantiques concernant les autres opérateurs. Sa définition à partir d'AUP repose sur des principes généraux d'*extension conservative* d'une algèbre, explicités dans la première partie du chapitre. Nous présentons la sémantique opérationnelle d'ATP, ainsi qu'une axiomatisation complète de l'équivalence forte qui permet, d'une part de comparer deux spécifications sans construire leurs modèles, et d'autre part de mettre en évidence les propriétés des opérateurs. Nous définissons en fin de chapitre la notion de *processus*

bien temporisé, qui caractérise les systèmes implémentables.

L'algèbre définie au chapitre 2 comporte un nombre volontairement restreint d'opérateurs, afin d'en simplifier l'étude théorique. Nous présentons dans le **chapitre 3** un ensemble d'opérateurs qui permettent de spécifier de manière simple des contraintes temporelles sur les processus. Ces opérateurs n'enrichissent pas l'algèbre du point de vue de la classe des modèles, mais ils apportent une facilité de description de systèmes temps réel en ATP, mise en évidence par plusieurs exemples.

Nous introduisons dans le **chapitre 4** la distinction entre l'*unité de temps* $\mathtt{U_t}$, dans laquelle sont exprimées les contraintes temporelles, et la *base de temps* $\mathtt{B_t}$, qui définit la fréquence des événements temporels perçus par le système. La sémantique d'ATP est définie dans le chapitre 2 en considérant ces deux mesures du temps comme identiques. Nous présentons une classe d'algèbres de processus ATP_g, variantes d'ATP, dont la sémantique est paramétrée par la *granularité* $g = \mathtt{B_t}/\mathtt{U_t}$. Nous montrons que certaines propriétés des systèmes ne sont pas préservées lorsqu'on modifie la granularité. Cette constatation mène à la définition d'un ensemble d'algèbres encore plus générales, appelées ATP_D, paramétrées par un *domaine temporel* quelconque, discret ou dense. On montre alors que la validité d'une *propriété de sûreté* est garantie pour toute granularité si elle est prouvée pour un domaine temporel dense. La fin de ce chapitre est consacrée à une étude comparative de différentes algèbres de processus temporisés apparues dans la littérature depuis quelques années. Nous ne prétendons pas à l'exhaustivité, qui semble difficile à obtenir dans ce domaine de recherche qui, quoique récent, est déjà très productif.

La **seconde partie** est consacrée à la *vérification*. Nous nous intéressons en premier lieu au problème de l'explosion du nombre d'états des modèles due aux valeurs des délais, puis à celui de la vérification proprement dite.

Nous présentons dans le **chapitre 5** le formalisme des *graphes temporisés*, destinés à jouer le rôle de *modèles intermédiaires* entre les processus d'ATP_D et leurs modèles définis au chapitre 4. Les graphes temporisés sont inspirés des *timed graphs* de R. Alur, C. Courcoubetis et D. Dill [ACD90]. Ils présentent la particularité d'être de taille indépendante des valeurs de délais apparaissant dans la spécification. Nous en définissons la sémantique opérationnelle, qui leur associe un système de transitions étiquetées de même nature que ceux des termes d'ATP_D. Nous présentons ensuite les principes théoriques de traduction d'un processus en graphe temporisé, qui doit bien entendu préserver la sémantique.

Finalement, nous décrivons dans le **chapitre 6** un algorithme de *model checking symbolique* qui s'applique aux graphes temporisés. Nous définissons pour cela un *μ-calcul temporel* pour l'expression des propriétés. L'*ensemble caractéristique* d'une formule est l'ensemble des états du modèle du graphe temporisé qui la satisfont. Nous montrons que cet ensemble peut être représenté par un *prédicat d'état* dont la validité est décidable. Le calcul des ensembles caractéristiques se ramène alors à celui du point fixe d'une fonctionnelle sur les prédicats d'état. L'exposé théorique est simplifié par l'introduction de la notion de *programme temporisé à commandes gardées*, formalisme équivalent à celui des graphes temporisés. Nous montrons également comment appliquer l'algorithme pour vérifier des propriétés de la logique temporelle temps réel TCTL [Alu91]. Nous prouvons pour cela que, si les deux logiques sont incomparables dans le cas général, toute formule de TCTL est cependant équivalente à une formule du μ-calcul temporel dans le cas particulier des programmes temporisés à commandes gardées (ou des graphes temporisés). L'algorithme doit sa correction à la notion de *graphe*

de régions définie dans [ACD90], où est présenté un algorithme de model checking énumératif. Ce dernier présente l'inconvénient d'être appliqué sur un modèle de plus bas niveau que les graphes temporisés, pour lequel on retombe sur le problème d'explosion du nombre d'états causée par les valeurs des délais. Les techniques symboliques présentées ici permettent d'éviter cette explosion, puisque le modèle de bas niveau n'est jamais construit.

Les trois **annexes** sont consacrées à certaines preuves particulièrement longues de théorèmes et propositions des chapitres 2 et 5.

Partie I

Spécification

Chapitre 1

Une algèbre de processus standard

Nous présentons dans ce chapitre une algèbre de processus non temporisés AUP (*Algebra of Untimed Processes*). Elle ne comporte que des opérateurs non temporels ; l'évolution des processus est donc indépendante du passage du temps.

Nous obtiendrons au chapitre 2 l'algèbre ATP en ajoutant des opérateurs spécifiant des contraintes temporelles sur les processus. L'idée est d'obtenir ainsi une *extension conservative* d'AUP, c'est-à-dire de préserver les propriétés des processus d'AUP dans la théorie d'ATP.

Nous décrivons ici la syntaxe et la sémantique opérationnelle d'AUP, et nous en proposons une axiomatisation complète. La préservation de propriétés mentionnée ci-dessus s'exprime alors formellement par :

Les axiomes d'AUP sont valides dans ATP, *pour les processus d'*AUP.

En fait, nous verrons que ces axiomes restent valides *pour tous les processus d'*ATP.

Nous ne présentons pas les preuves des différentes propriétés d'AUP dans ce chapitre, car celles-ci ne sont que des cas particuliers de propositions et théorèmes énoncés à propos d'ATP dans le chapitre 2.

L'étude est menée sur une algèbre plus générale, dont AUP est une sous-algèbre.

AUP emprunte des éléments aux algèbres CCS de R. Milner [Mil80, Mil89] (pour les aspects séquentiels) et ACP de J. A. Bergstra et J .W. Klop [BK86] (pour les aspects parallèles).

1.1 Actions et communications

Les processus exécutent des *actions abstraites* (non interprétées). Une action peut être locale à un processus ou résulter d'une communication par rendez-vous entre plusieurs processus.

Définition 1.1 (actions)

L'ensemble des actions exécutables par les processus est noté $\mathcal{A}$.

$\mathcal{A}$ comporte un sous-ensemble particulier $\Xi \stackrel{\text{def}}{=} \{\xi^n, n \in \mathbb{N} - \{0\}\}$. Un élément ξ^n de Ξ est appelé *action d'échappement de niveau* n.

$\mathcal{A}$ possède de plus une *action interne*, notée τ ou ξ^0.

□

Une communication est le résultat de la composition d'actions individuelles de chacun des processus concernés. On spécifie les actions qui peuvent être ainsi composées, ainsi que l'action résultante, au moyen d'une fonction de communication :

Définition 1.2 (fonction de communication)
L'ensemble $\mathcal{A} \cup \{\bot\}$, où $\bot$ est un symbole n'appartenant pas à $\mathcal{A}$, est noté $\mathcal{A}_\bot$.

$\mathcal{A}_\bot$ est muni d'une loi de composition $|$, appelée *fonction de communication*, telle que $(\mathcal{A}_\bot, |)$ forme un semi-groupe commutatif, $\bot$ étant élément absorbant :

$$\forall a, b, c \in \mathcal{A}_\bot \ : \ a|(b|c) = (a|b)|c, \ \ a|b = b|a, \ \ \bot|a = \bot$$

Si a, b et c sont des actions, $a|b = c$ indique que c est l'action résultant de la communication entre a et b. Inversement, $a|b = \bot$ indique une impossibilité de communication entre a et b.

Les actions ξ^n (y compris τ) sont telles que $a|\xi^n = \bot$ pour tout a dans $\mathcal{A}_\bot$; elles ne peuvent donc communiquer avec aucune autre action.

□

Les actions sont *atomiques* : l'exécution d'une action ne peut commencer si une autre action est en cours. Cette hypothèse nous permet de les supposer *instantanées* : le début et la fin de l'exécution d'une action sont simultanés.

1.2 Syntaxe

Soit $\mathcal{V}$ un ensemble de variables, dont les éléments sont notés X, Y, Z, X_1, ...

L'algèbre $\mathcal{P}_u$, dont les termes sont notés P, Q, R, P_1, ..., est définie par la syntaxe suivante :

$$\begin{array}{rcl} P ::= & \delta \quad | \quad X \quad | \quad \widetilde{a}\, P \quad | \quad P + P \quad | \\ & P \| P \quad | \quad \partial_H(P) \quad | \quad P \bigtriangledown P \quad | \quad \mathtt{rec} X \cdot P \end{array}$$

où $X \in \mathcal{V}$, $a \in \mathcal{A}$, et $H \subseteq \mathcal{A}$.

La signification des différents opérateurs est la suivante :

- δ est un *processus terminé*. Il ne peut exécuter aucune action.
- X est un processus non déterminé, destiné à être lié par l'opérateur de récursion (voir ci-dessous).
- $\widetilde{a}\ \_$ est l'opérateur de *préfixage*. $\widetilde{a}P$ peut exécuter l'action a pour se comporter ensuite comme P.
- $\_ + \_$ est l'opérateur de *choix non déterministe*. $P + Q$ se comporte comme P ou comme Q. Par exemple, $\widetilde{a}\,\delta + \widetilde{b}\,\delta$ peut, soit exécuter l'action a avant de s'arrêter, soit exécuter l'action b avant de s'arrêter également.
- $\_\|\_$ est l'opérateur de *composition parallèle*. Si P peut exécuter l'action a et se comporte ensuite comme P', alors $P\|Q$ (resp. $Q\|P$) peut exécuter a pour se comporter ensuite comme $P'\|Q$ (resp. $Q\|P'$). Si de plus Q peut exécuter l'action b et se comporte ensuite comme Q', et si $a|b = c \neq \bot$, alors $P\|Q$ peut exécuter l'action c pour se comporter ensuite comme $P'\|Q'$ (communication entre P et Q).
- $\partial_H(\_)$ est l'opérateur d'*encapsulation*. $\partial_H(P)$ se comporte comme P, mais ne peut exécuter les actions de H.

- $\_\triangledown\_$ est l'opérateur de *continuation*. Le terme $R \stackrel{\text{def}}{=} P \triangledown Q$ se comporte comme P, tant que celui-ci n'exécute pas d'action d'échappement. Par contre, si P exécute une action ξ^n ($n > 0$), alors R exécute l'action ξ^{n-1} et se comporte ensuite comme Q. L'opérateur permet donc à P de transmettre le contrôle à Q via une action d'échappement.
- Finalement, $\mathtt{rec}X \cdot \_$ est l'opérateur de *récursion*. Intuitivement, $\mathtt{rec}X \cdot P$ se comporte comme P, où la variable X se comporte comme $\mathtt{rec}X \cdot P$. Par exemple, le terme $P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \tilde{a}\, X$ ne peut exécuter que l'action a, pour se comporter ensuite comme P. Ainsi, le comportement de P consiste à exécuter indéfiniment l'action a.

La terminaison correspond au *Nil* de CCS. Le préfixage et la récursion correspondent à ceux de CCS. Le choix non déterministe est présent dans CCS et ACP. La composition parallèle et l'encapsulation sont empruntées à ACP. L'opérateur de continuation est inspiré de la construction `trap-exit` d'Esterel [BC84, BCG87] ; il en diffère essentiellement par l'interdiction de communication entre actions d'échappement dans AUP.

Le choix des opérateurs est guidé par un souci de simplicité et d'expressivité. Le préfixage de CCS a été préféré à la composition séquentielle d'ACP, dont la sémantique est plus délicate à définir (voir remarque 1.2, page 32). En revanche, l'opérateur de composition parallèle d'ACP, associé à la fonction de communication des actions, offre une plus grande souplesse que celui de CCS, qui ne permet que le rendez-vous binaire.

Les notions de *variable libre* et de *terme fermé* sont identiques à celles de CCS :

Définition 1.3 (variable libre)
Une variable X est *libre* dans un terme P s'il existe une occurrence de X dans P non contenue dans un sous-terme $\mathtt{rec}X \cdot P'$ de P (*occurrence libre* de X).
□

Définition 1.4 (terme fermé)
Un terme P est *fermé* si aucune variable n'est libre dans P.
□

Nous présentons ici une définition supplémentaire, qui permet d'obtenir une première restriction de l'algèbre.

Définition 1.5 (terme régulier)
Un terme P est dit *régulier* si pour tous ses sous-termes $Q \| R$, $\partial_H(Q)$ et $Q \triangledown S$, Q et R sont fermés (rien n'est exigé sur S). On dit alors que P ne possède pas de récursion à travers les opérateurs de composition parallèle, d'encapsulation et de continuation (ou "pas de récursion à travers le parallèle, l'encapsulation ou la continuation" par abus de langage).

La sous-algèbre de $\mathcal{P}_u$ ne comportant que des termes réguliers est notée $\mathcal{P}_u^r$.
□

Exemple 1.1
X et Y sont libres dans $\tilde{a}\, X \;+\; \mathtt{rec}X \cdot (X + \tilde{b}\, Y)$.

Le terme $\tilde{a}\,(\mathtt{rec}X \cdot (\tilde{b}\, \partial_{\{b\}}(X) \| X))$ est fermé, mais pas régulier.

Par contre, le terme $((\mathtt{rec}X \cdot \tilde{a}\, X) \| \tilde{b}\, \delta) \triangledown Y$ est régulier, mais non fermé.

Finalement, le terme $(\tilde{a}\, \mathtt{rec}X \cdot \tilde{b}\, \delta + \xi^1 \delta \triangledown X) \| \partial_{\{c\}}(\mathtt{rec}Y \cdot (Y + \tilde{c}\, \delta))$ est à la fois fermé et régulier.
□

Les notions de *variable gardée* et de *terme bien gardé* sont définies de la manière suivante :

Définition 1.6 (variable gardée)
Une variable X est *gardée* dans P si toutes ses occurrences libres dans P sont, soit dans l'argument d'un opérateur de préfixage, soit dans le second argument d'un opérateur de continuation. On dit que le préfixage (resp. la continuation) est un *opérateur gardant son argument (resp. son second argument)*.
□

Définition 1.7 (terme rec-bien gardé)
Un terme récursif $\mathtt{rec}X \cdot P$ est *rec-bien gardé* si X est gardé dans P.
□

Définition 1.8 (terme bien gardé)
Un terme P de $\mathcal{P}_u$ est *bien gardé* si tout sous-terme récursif de P est rec-bien gardé.
□

Exemple 1.2
X est gardé dans $Y + \widetilde{a}\, X$ et dans $Y \bigtriangledown X$, mais pas Y.

Les termes $P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot (Y \| \widetilde{a}\, X)$ et $Q \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \mathtt{rec}Y \cdot (Y + \widetilde{a}\, X) \bigtriangledown X$ sont rec-bien gardés, mais pas le terme $\mathtt{rec}Y \cdot (Y + \widetilde{a}\, X)$.

Finalement, P est bien gardé, mais pas Q, car le sous-terme $\mathtt{rec}Y \cdot (Y + \widetilde{a}\, X)$ n'est pas rec-bien gardé.
□

Ces différentes notions nous permettent de définir l'algèbre AUP :

Définition 1.9 (AUP)
L'algèbre AUP, dont les éléments sont appelés *processus non temporisés*, est la sous-algèbre des termes *fermés et bien gardés* de $\mathcal{P}_u^r$, autrement dit des termes fermés, réguliers et bien gardés de $\mathcal{P}_u$.
□

Un processus non temporisé est donc un terme de $\mathcal{P}_u$ sans variable libre, sans récursion à travers le parallèle ou la continuation et dans lequel, pour tout sous-terme $\mathtt{rec}X \cdot P$, X est dans la portée d'un préfixage ou dans le second argument d'une continuation dans P.

Dans ce chapitre, nous utilisons l'appellation simplifiée *processus* pour les processus non temporisés.

1.3 Sémantique opérationnelle

Nous décrivons à présent la sémantique opérationnelle de $\mathcal{P}_u$, qui associe à un terme un *modèle*, sous la forme d'un système de transitions étiquetées.

Définition 1.10 (système de transitions étiquetées)
Un *système de transitions étiquetées* est un quadruplet

$$(S, \mathcal{L}, s_0, \mathcal{T})$$

constitué d'un ensemble d'*états* S, d'un ensemble d'*étiquettes* $\mathcal{L}$, d'un *état initial* s_0 et d'une *relation de transition* $\mathcal{T} \subseteq S \times \mathcal{L} \times S$.
□

La description des modèles se fait en deux étapes. Nous définissons tout d'abord une relation de transition $\rightarrow$ entre les éléments de $\mathcal{P}_u$. Une relation d'équivalence $\sim$ entre termes permet ensuite d'obtenir les modèles, dont les états sont des classes d'équivalence de termes, et dont la relation de transition est celle induite par $\rightarrow$ sur l'algèbre-quotient $\mathcal{P}_u/\!\sim$.

La relation d'équivalence spécifie les termes dont les comportements sont considérés indistinguables.

La relation de transition décrit ce qu'on peut observer du comportement d'un terme. Les "événements" observables sont les éléments ℓ de $\mathcal{L}$. $P \xrightarrow{\ell} Q$ se lit "*P a une transition par ℓ menant vers Q*", et signifie que dans le comportement de P, on peut observer initialement ℓ, et que le comportement ultérieur correspond à celui de Q. On dit que Q est un *successeur* (par ℓ) de P.

Essentiellement, le comportement d'un terme consiste à exécuter des actions. Nous supposons que toutes les actions sont observables ; ceci impose $\mathcal{A} \subset \mathcal{L}$.

Le comportement d'un terme composé $\mathtt{Op}(P_1, \ldots, P_n)$ doit se déduire des comportements de $P_1, \ldots, P_n$. Si P_1 est équivalent à un autre terme P_1' (indistinguable de P_1'), il doit alors en être de même pour $\mathtt{Op}(P_1, \ldots, P_n)$ et $\mathtt{Op}(P_1', \ldots, P_n)$. Autrement dit, le relation d'équivalence $\sim$ doit être une congruence pour les opérateurs de $\mathcal{P}_u$.

Il est indispensable de pouvoir détecter la présence d'une variable non gardée dans un terme. En effet, dans le cas contraire, il est impossible de distinguer les comportements des termes

$$P \stackrel{\text{def}}{=} \widetilde{a}\,\delta \qquad \text{et} \qquad Q \stackrel{\text{def}}{=} X + \widetilde{a}\,\delta$$

L'exigence de congruence pour $\sim$ implique alors

$$R \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \widetilde{b}\,P \;\sim\; S \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \widetilde{b}\,Q$$

Or, d'après la signification intuitive des comportements décrite plus haut, le terme R ne peut qu'exécuter l'action b pour se comporter ensuite comme P, qui ne peut exécuter que l'action a avant de se terminer. En revanche, le terme S, s'il ne peut également qu'exécuter initialement l'action b, se comporte ensuite comme $S + \widetilde{a}\,\delta$. De même que P, ce dernier peut exécuter a avant de s'arrêter ; mais il peut également exécuter les actions initiales de S, c'est-à-dire b. Les deux termes R et S ne peuvent donc pas être considérés comme équivalents.

Pour tenir compte de la présence des variables non gardées, nous ajoutons $\mathcal{V}$ à l'ensemble des étiquettes $\mathcal{L}$. Une transition $P \xrightarrow{X} Q$ signifie que X est non gardé dans P.

1.3.1 Relation de transition

Les remarques ci-dessus amènent à la définition suivante du domaine de la relation de transition :

$$\rightarrow \subseteq \mathcal{P}_u \times \mathcal{L} \times \mathcal{P}_u, \quad \text{avec } \mathcal{L} = \mathcal{A} \cup \mathcal{V}$$

Elle est décrite par induction structurelle, selon le style de G. Plotkin [Plo81], au moyen de règles de la forme

$$\frac{\text{prémisses}}{\text{conclusion}}$$

La conclusion est de la forme $\mathtt{Op}(P_1, \ldots, P_n) \xrightarrow{\ell} Q$, où n est l'arité de l'opérateur $\mathtt{Op}$; l'ensemble des prémisses (qui peut être vide) est de la forme

$$\{P_i \xrightarrow{\ell_i} P_i' \mid i \in I \subseteq [1,n]\} \cup \{P_j \xrightarrow{\ell_j} \not\rightarrow \mid j \in J \subseteq [1,n]\}$$

Une telle règle se lit :

Si, pour tout i dans I, P_i a une transition par ℓ_i vers P'_i, et pour tout j dans J, P_j n'a pas de transition par ℓ_j, alors $\mathtt{Op}(P_1, \ldots, P_n)$ a une transition par ℓ vers Q.

Ce mode de description de la relation de transition correspond au format de règles appelé GSOS, décrit dans [BIM88].

La relation de transition $\rightarrow$ est la plus petite relation satisfaisant les règles suivantes :

Règles d'action	**Règles de variable**
$[\widetilde{a}^a] : \dfrac{}{\widetilde{a}\,P \xrightarrow{a} P}$	$[X^X] : \dfrac{}{X \xrightarrow{X} \delta}$
$[+^a_l] : \dfrac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'}$ $\quad [+^a_r] : \dfrac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$	$[+^X_l] : \dfrac{P \xrightarrow{X} P'}{P+Q \xrightarrow{X} P'}$ $\quad [+^X_r] : \dfrac{Q \xrightarrow{X} Q'}{P+Q \xrightarrow{X} Q'}$
$[\|^a_l] : \dfrac{P \xrightarrow{a} P'}{P\|Q \xrightarrow{a} P'\|Q}$ $\quad [\|^a_r] : \dfrac{Q \xrightarrow{a} Q'}{P\|Q \xrightarrow{a} P\|Q'}$ $\quad [\|^a_c] : \dfrac{P \xrightarrow{a} P', Q \xrightarrow{b} Q'}{P\|Q \xrightarrow{a\|b} P'\|Q'}\ a\|b \neq \bot$	$[\|^X_l] : \dfrac{P \xrightarrow{X} P'}{P\|Q \xrightarrow{X} P'}$ $\quad [\|^X_r] : \dfrac{Q \xrightarrow{X} Q'}{P\|Q \xrightarrow{X} Q'}$
$[\partial^a] : \dfrac{P \xrightarrow{a} P'}{\partial_H(P) \xrightarrow{a} \partial_H(P')}\ a \notin H$	$[\partial^X] : \dfrac{P \xrightarrow{X} P'}{\partial_H(P) \xrightarrow{X} P'}$
$[\triangledown^a_a] : \dfrac{P \xrightarrow{a} P',\ a \notin \Xi}{P \triangledown Q \xrightarrow{a} P' \triangledown Q}$ $\quad [\triangledown^a_\xi] : \dfrac{P \xrightarrow{\xi^{n+1}} P'}{P \triangledown Q \xrightarrow{\xi^n} Q}$	$[\triangledown^X] : \dfrac{P \xrightarrow{X} P'}{P \triangledown Q \xrightarrow{X} P'}$
$[\mathtt{rec}^a] : \dfrac{P \xrightarrow{a} P'}{\mathtt{rec}X \cdot P \xrightarrow{a} P'[\mathtt{rec}X \cdot P/X]}$	$[\mathtt{rec}^X] : \dfrac{P \xrightarrow{Y} P'}{\mathtt{rec}X \cdot P \xrightarrow{Y} P'}\ Y \neq X$

La notation $P[Q/X]$ représente le terme P, dans lequel Q est substitué à toutes les occurrences *libres* de P (avec renommage éventuel des variables non libres de P) :

$\delta[Q/X]$	$\stackrel{\text{def}}{=}$	δ	$\partial_H(P)[Q/X]$	$\stackrel{\text{def}}{=}$	$\partial_H(P[Q/X])$
$X[Q/X]$	$\stackrel{\text{def}}{=}$	Q	$(P_1 \triangledown P_2)[Q/X]$	$\stackrel{\text{def}}{=}$	$P_1[Q/X] \triangledown P_2[Q/X]$
$Y[Q/X]$	$\stackrel{\text{def}}{=}$	Y si $Y \neq X$	$(\mathtt{rec}X \cdot P)[Q/X]$	$\stackrel{\text{def}}{=}$	$\mathtt{rec}X \cdot P$
$(\widetilde{a}\,P)[Q/X]$	$\stackrel{\text{def}}{=}$	$\widetilde{a}\,(P[Q/X])$	$(\mathtt{rec}Y \cdot P)[Q/X]$	$\stackrel{\text{def}}{=}$	$\mathtt{rec}Z \cdot P[Z/Y][Q/X]$
$(P_1 + P_2)[Q/X]$	$\stackrel{\text{def}}{=}$	$P_1[Q/X] + P_2[Q/X]$	$Y \neq X, Z \neq X$		
$(P_1 \| P_2)[Q/X]$	$\stackrel{\text{def}}{=}$	$P_1[Q/X] \| P_2[Q/X]$	avec Z pas libre dans $\mathtt{rec}Y \cdot P$ ni dans Q		

Les règles de sémantique en partie gauche décrivent les actions exécutées par les termes. On constate aisément que la sémantique des opérateurs, pour ce qui concerne les actions, correspond à leur signification intuitive énoncée dans la section 1.2. Les règles d'action sont suffisantes pour spécifier la sémantique d'AUP. Dans ce cas, la règle $[\mathtt{rec}^a]$ peut être remplacée par la règle de récursion plus classique

$$\frac{P[\mathtt{rec}X \cdot P/X] \xrightarrow{a} P'}{\mathtt{rec}X \cdot P \xrightarrow{a} P'}$$

En effet, il est montré dans [BD90] que les deux règles sont équivalentes en ce qui concerne les termes bien gardés.

Les règles de droite concernent les variables, et permettent de définir les notions sémantiques de variables *actives* et *activables* :

Définition 1.11 (variables actives et activables)

Une variable X est *active* dans un terme P s'il existe Q tel que $P \xrightarrow{X} Q$.

Une variable X est *activable* dans un terme P si X est active dans P, ou si X est activable dans un successeur de P.

□

La proposition suivante se montre aisément par induction sur la structure des termes :

Proposition 1.1

Pour tout terme P et toute variable X, si $P \xrightarrow{X} Q$, alors $Q = \delta$.

□

On relie les variables non gardées ou libres d'un terme (notions syntaxiques) avec ses variables actives ou activables (notions sémantiques) par la proposition suivante :

Proposition 1.2

(1) X est active dans P si et seulement si X est non gardée dans P.

(2) Si X est activable dans P, alors X est libre dans P.

□

Remarquer que la seconde partie n'est qu'une implication. En effet, une variable peut être libre sans être activable. Par exemple, X est libre dans $P \stackrel{\text{def}}{=} \partial_{\{a\}}(\tilde{a}\, X)$ mais n'est pas activable, car P n'a pas de transition, puisque l'encapsulation l'empêche d'exécuter l'action a.

1.3.2 Relation d'équivalence

Nous choisissons pour identifier les termes la relation d'*équivalence forte* de Milner [Mil80, Mil89], basée sur la notion de bisimulation de Park [Par80].

Définition 1.12 (bisimulation forte, équivalence forte)

Une relation $\mathcal{R} \subseteq \mathcal{P}_u \times \mathcal{P}_u$ est une *bisimulation forte* si pour tous P, Q dans $\mathcal{P}_u$,

$$P\ \mathcal{R}\ Q\ \Rightarrow\ \forall \ell\ :\ \begin{cases} P \xrightarrow{\ell} P'\ \Rightarrow\ \exists Q'\ :\ Q \xrightarrow{\ell} Q' \text{ et } P'\ \mathcal{R}\ Q' \\ Q \xrightarrow{\ell} Q'\ \Rightarrow\ \exists P'\ :\ P \xrightarrow{\ell} P' \text{ et } P'\ \mathcal{R}\ Q' \end{cases}$$

L'équivalence forte $\sim$ est la plus grande bisimulation forte.

□

La définition de l'équivalence forte suggère que, pour prouver $P \sim Q$, il suffit d'exhiber une bisimulation forte $\mathcal{R}$, telle que $P\ \mathcal{R}\ Q$. Par exemple, pour prouver que les deux processus

$$P \stackrel{\text{def}}{=} \texttt{rec}X \cdot \tilde{a}\, X + \tilde{a}\, \tilde{a}\, X \quad \text{et} \quad Q \stackrel{\text{def}}{=} \texttt{rec}Y \cdot \tilde{a}\, \tilde{a}\, \tilde{a}\, Y$$

sont fortement équivalents, on peut définir la relation

$$\mathcal{R} = \{(P, Q), (P, \tilde{a}\, \tilde{a}\, Q), (P, \tilde{a}\, Q), (\tilde{a}\, P, \tilde{a}\, \tilde{a}\, Q), (\tilde{a}\, P, \tilde{a}\, Q)\}$$

et montrer que $\mathcal{R}$ est une bisimulation forte, en utilisant les règles de sémantique opérationnelle.

Une autre technique de preuve utilise la notion de *bisimulation forte modulo* $\sim$ définie par Milner [Mil89] (strong bisimulation up to $\sim$) :

Définition 1.13

Une relation $\mathcal{R}$ est une *bisimulation forte modulo* $\sim$ si pour tous P, Q dans $\mathcal{P}_u$,

$$P\ \mathcal{R}\ Q\ \Rightarrow\ \forall \ell\ :\ \begin{cases} P \xrightarrow{\ell} P'\ \Rightarrow\ \exists Q'\ :\ Q \xrightarrow{\ell} Q' \text{ et } P' \sim \mathcal{R} \sim Q' \\ Q \xrightarrow{\ell} Q'\ \Rightarrow\ \exists P'\ :\ P \xrightarrow{\ell} P' \text{ et } P' \sim \mathcal{R} \sim Q' \end{cases}$$

□

La notation $\sim\mathcal{R}\sim$ décrit la composition de relations :

$$P \sim\mathcal{R}\sim Q \iff \exists P', Q' \,:\, P \sim P' \wedge P' \,\mathcal{R}\, Q' \wedge Q' \sim Q$$

Pour montrer $P \sim Q$, il suffit alors de prouver que (P, Q) appartient à une bisimulation forte modulo $\sim$.

Comme nous l'avons expliqué plus haut, il est important de vérifier que l'équivalence forte est une congruence. Il faut donc prouver la proposition suivante :

Proposition 1.3

$\forall P, Q, R \in \mathcal{P}_u$, $\forall a \in \mathcal{A}$, $\forall H \subseteq \mathcal{A}$, $\forall X \in \mathcal{V}$:

$$P \sim Q \;\Rightarrow\; \begin{cases} \tilde{a}\, P \sim \tilde{a}\, Q & \\ P + R \sim Q + R & \text{et} \quad R + P \sim R + Q \\ P \| R \sim Q \| R & \text{et} \quad R \| P \sim R \| Q \\ \partial_H(P) \sim \partial_H(Q) & \\ P \bigtriangledown R \sim Q \bigtriangledown R & \text{et} \quad R \bigtriangledown P \sim R \bigtriangledown Q \\ \mathtt{rec}X \cdot P \sim \mathtt{rec}X \cdot Q & \end{cases}$$

□

Preuve. Dans [Gro89], J.-F. Groote prouve un théorème qui stipule que si les règles de sémantique sont dans un format appelé *ntyft/ntyxt*, alors l'équivalence forte est une congruence. Le format GSOS adopté pour nos règles est un sous-ensemble du format *ntyft/ntyxt*, donc le théorème s'applique à la sémantique de $\mathcal{P}_u$. ■

1.3.3 Modèles

La sémantique opérationnelle induit un relation de transition $\rightarrow_{/\sim}$ sur l'algèbre-quotient $\mathcal{P}_u/\sim$ (dont les éléments sont notés $[P]$ pour la classe de P). Cette relation est définie, pour deux classes d'équivalence E et F, par :

$$E \xrightarrow{\ell}_{/\sim} F \iff \exists P \in E, \exists Q \in F, \,:\, P \xrightarrow{a} Q$$

Les modèles des éléments de $\mathcal{P}_u$ sont définis de la manière suivante :

Définition 1.14 (modèles)

Le modèle d'un terme P de $\mathcal{P}_u$, noté $\mathcal{M}_u(P)$, est le système de transitions étiquetées $(S, \mathcal{L}, s_0, \mathcal{T})$, où

- l'ensemble des états S est le sous-ensemble de $\mathcal{P}_u/\sim$ des classes E accessibles à partir de $[P]$, c'est-à-dire tels que $[P]\,(\rightarrow_{/\sim})^*\, E$;
- l'état initial s_0 est $[P]$;
- la relation de transition $\mathcal{T}$ est $\rightarrow_{/\sim}$.

□

Les modèles sont représentés sous la forme de graphes étiquetés orientés. Pour des questions de lisibilité, nous prenons parfois des libertés avec la définition ci-dessus, en donnant des graphes dont le nombre d'états n'est pas nécessairement minimal. Les modèles de quelques termes sont présentés figure 1.1.

Les deux propositions suivantes caractérisent les modèles des deux sous-algèbres $\mathcal{P}_u^r$ et AUP.

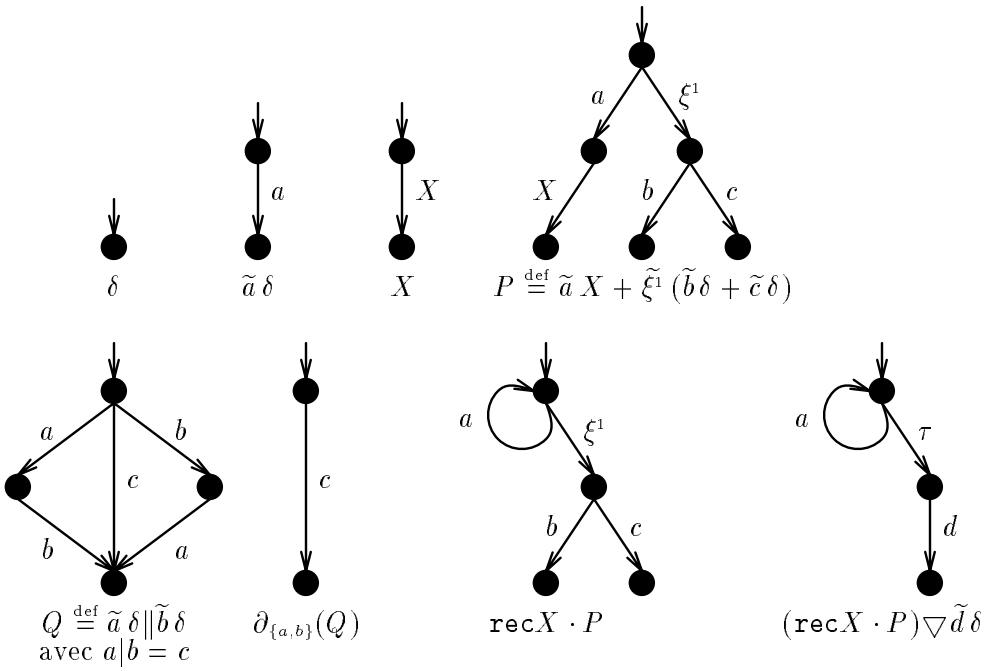


Figure 1.1 : modèles de termes de $\mathcal{P}_u$

Proposition 1.4 (modèles de $\mathcal{P}_u^r$)
Le modèle d'un terme de $\mathcal{P}_u^r$ est tel que

- le nombre d'états est fini ;
- le facteur de branchement en chaque état est fini ;
- l'état but d'une transition étiquetée par une variable est un puits (état sans successeur).

Inversement, tout système de transitions étiquetées par des éléments de $\mathcal{L}$, qui satisfait ces trois propriétés, est le modèle d'un élément de $\mathcal{P}_u^r$.

□

Proposition 1.5
Le modèle d'un processus d'AUP est tel que

- le nombre d'états est fini ;
- le facteur de branchement en chaque état est fini ;
- aucune transition n'est étiquetée par une variable.

De plus, tout système de transitions étiquetées fini, à branchement fini, et dont les étiquettes sont éléments de $\mathcal{A}$ est le modèle d'un processus d'AUP.

□

Interdire la récursion à travers le parallèle ou la continuation assure l'obtention de modèles finis (d'où l'appellation *régulier* pour une terme de $\mathcal{P}_u^r$), ce qui n'est pas le cas si de telles récursions sont permises. En ce qui concerne la récursion à travers l'encapsulation, voir la remarque 1.1, page 31. La figure 1.2 présente deux exemples de modèles infinis de termes de $\mathcal{P}_u$.

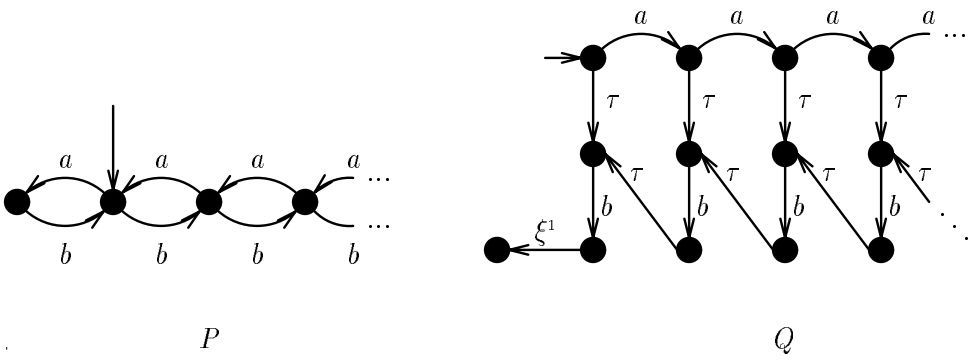


Figure 1.2 : deux modèles infinis : $P \stackrel{\text{def}}{=} \texttt{rec}X \cdot (\widetilde{a}\,\delta \| \widetilde{b}\,X)$
$Q \stackrel{\text{def}}{=} \texttt{rec}X \cdot ((\widetilde{a}\,X + \widetilde{\xi^1}\,\delta) \triangledown \widetilde{b}\,\widetilde{\xi^1}\,\delta)$

1.4 Axiomatisation

Nous présentons un système d'axiomes qui permet de décider de l'équivalence de deux termes de $\mathcal{P}_u^r$, sans avoir à construire leurs modèles. Cette axiomatisation est consistante et complète pour l'équivalence forte.

Comme pour l'algèbre ACP [BK86], l'axiomatisation de l'opérateur de composition parallèle nécessite l'emploi de deux opérateurs supplémentaires, appelés *composition à gauche* (noté $\lfloor\!\lfloor$) et *communication* (noté $|$). Ce sont des formes restrictives de la composition parallèle : la composition à gauche de P et Q doit exécuter initialement une action de P ; la communication entre P et Q doit exécuter en premier lieu une communication entre une action de P et une action de Q. La sémantique opérationnelle de ces deux opérateurs est décrite par les règles suivantes :

composition à gauche	**communication**
$[\lfloor\!\lfloor^a] : \dfrac{P \xrightarrow{a} P'}{P \lfloor\!\lfloor Q \xrightarrow{a} P' \| Q}$	$[\mid^a] : \dfrac{P \xrightarrow{a} P', Q \xrightarrow{b} Q'}{P \mid Q \xrightarrow{a \mid b} P' \| Q'}\ a \mid b \neq \perp$

Remarquer que ces règles sont aussi dans le format GSOS. Il n'est pas nécessaire de définir les transitions par une variable. En effet, l'axiomatisation concerne $\mathcal{P}_u^r$ dont tous les termes sont réguliers. Une composition parallèle ne s'applique donc qu'à des termes fermés.

L'axiomatisation définit une congruence $\equiv$. C'est la plus petite congruence sur $\mathcal{P}_u^r$ induite par les axiomes suivants :

$[+_u 1]$	$P + Q \equiv Q + P$	$[\mid_u 1]$	$P \mid Q \equiv Q \mid P$
$[+_u 2]$	$P + (Q + R) \equiv (P + Q) + R$	$[\mid_u 2]$	$(P + Q) \mid R \equiv P \mid R + Q \mid R$
$[+_u 3]$	$P + P \equiv P$	$[\mid_u 3]$	$\delta \mid P \equiv \delta$
$[+_u 4]$	$P + \delta \equiv P$	$[\mid_u 4]$	$\widetilde{a}\,P \mid \widetilde{b}\,Q \equiv \delta$ si $a \mid b = \bot$
$[\parallel_u]$	$P \parallel Q \equiv P \between Q + Q \between P + P \mid Q$	$[\mid_u 5]$	$\widetilde{a}\,P \mid \widetilde{b}\,Q \equiv \widetilde{a \mid b}\,(P \parallel Q)$ sinon
$[\between_u 1]$	$(P + Q) \between R \equiv P \between R + Q \between R$	$[\partial_u 1]$	$\partial_H(P + Q) \equiv \partial_H(P) + \partial_H(Q)$
$[\between_u 2]$	$\delta \between P \equiv \delta$	$[\partial_u 2]$	$\partial_H(\delta) \equiv \delta$
$[\between_u 3]$	$(\widetilde{a}\,P) \between Q \equiv \widetilde{a}\,(P \parallel Q)$	$[\partial_u 3]$	$\partial_H(\widetilde{a}\,P) \equiv \delta$ si $a \in H$
$[\triangledown_u 1]$	$(P + Q) \triangledown R \equiv P \triangledown R + Q \triangledown R$	$[\partial_u 4]$	$\partial_H(\widetilde{a}\,P) \equiv \widetilde{a}\,\partial_H(P)$ si $a \notin H$
$[\triangledown_u 2]$	$\delta \triangledown P \equiv \delta$	$[\mathtt{rec}_u 1]$	$\mathtt{rec}X \cdot P \equiv P[\mathtt{rec}X \cdot P/X]$
$[\triangledown_u 3]$	$(\widetilde{a}\,P) \triangledown Q \equiv \widetilde{a}\,(P \triangledown Q)$ si $a \notin \Xi$	$[\mathtt{rec}_u 3]$	si $P[Q/X] \equiv Q$ et X est gardé dans P alors $\mathtt{rec}X \cdot P \equiv Q$
$[\triangledown_u 4]$	$(\widetilde{\xi^{n+1}}\,P) \triangledown Q \equiv \widetilde{\xi^n}\,Q$	$[\mathtt{rec}_u 3]$	$\mathtt{rec}X \cdot (X + P) \equiv \mathtt{rec}X \cdot P$

Cette axiomatisation n'est pas valide pour l'algèbre $\mathcal{P}_u$. En effet, dans le cas général, l'équation $[\mathtt{rec}_u 1]$ n'est pas vérifiée. On peut le constater sur le terme

$$P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot X \parallel \widetilde{a}\,\delta$$

en supposant $a \mid a = b$, où b n'est ni a, ni $\bot$.

En suivant les règles de sémantique opérationnelle, l'unique transition initiale possible pour P est : $P \xrightarrow{a} P \parallel \delta$.

D'autre part, on constate que

$$Q \stackrel{\text{def}}{=} (X \parallel \widetilde{a}\,\delta)[P/X] \;=\; P \parallel \widetilde{a}\,\delta$$

(égalité syntaxique). Puisque P peut exécuter l'action a, le terme Q possède une transition par b (communication entre a et a). Les deux termes P et Q ne sont donc pas équivalents, ce qui contredit l'axiome $[\mathtt{rec}_u 1]$.

Cet axiome est clairement valide si et seulement si les termes $\mathtt{rec}X \cdot P$ et $P[\mathtt{rec}X \cdot P/X]$ ont les mêmes systèmes de transitions. On en déduit que, puisque l'axiomatisation est consistante pour tout $\mathcal{P}_u^r$, la proposition

$$\text{si } P[\mathtt{rec}X \cdot P/X] \xrightarrow{\ell} P' \text{ alors } \mathtt{rec}X \cdot P \xrightarrow{\ell} P'$$

est vraie dans $\mathcal{P}_u^r$. On peut même prouver que les règles de sémantique de l'opérateur de récursion sont *équivalentes* dans $\mathcal{P}_u^r$ à la règle

$$\frac{P[\mathtt{rec}X \cdot P/X] \xrightarrow{\ell} P'}{\mathtt{rec}X \cdot P \xrightarrow{\ell} P'}$$

Remarque 1.1 (la récursion à travers l'encapsulation)
En théorie, l'interdiction de la récursion à travers l'encapsulation n'est pas nécessaire pour

obtenir des modèles finis. Cependant, si on l'autorise, il faut définir la sémantique de manière beaucoup plus compliquée afin que la règle ci-dessus soit valide. Nous avons donc choisi de l'interdire car cela ne diminue en rien le pouvoir d'expression de $\mathcal{P}_u^r$ (en termes de classe de modèles).

□

La commutativité et l'associativité de l'opérateur + (axiomes $[+_u 1]$ et $[+_u 2]$) permettent d'utiliser la notation $\sum_{i\in I} P_i$, où I est un ensemble *fini* d'indices, pour désigner le choix non déterministe des P_i. L'axiome $[+_u 4]$ nous autorise de plus à identifier $\sum_{i\in\emptyset} P_i$ et δ.

1.4.1 Forme canonique

L'axiomatisation permet de définir une *forme canonique* pour les termes de $\mathcal{P}_u^r$:

Proposition 1.6 (forme canonique pour $\mathcal{P}_u^r$)
Soit P un terme de $\mathcal{P}_u^r$, dont les variables libres sont dans l'ensemble $\overline{Y}$. Il existe $n \geq 1$ et des termes $P_1, \ldots, P_n$ dans $\mathcal{P}_u^r$, dont les variables libres sont dans $\overline{Y}$, tels que $P \equiv P_1$, et chaque P_i est solution d'une équation de la forme :

$$P_i \equiv \sum_{j\in J_i} \widetilde{a_j} P_{j_i} + \sum_{k\in K_i} Y_k \qquad j_i \in [1,n], Y_k \in \overline{Y}$$

□

Un processus d'AUP n'ayant pas de variable libre, on en déduit l'énoncé suivant :

Proposition 1.7 (forme canonique pour AUP)
Soit P un processus d'AUP. Il existe $n \geq 1$ et des termes $P_1, \ldots, P_n$ dans AUP, tels que $P \equiv P_1$, et chaque P_i est solution d'une équation de la forme :

$$P_i \equiv \sum_{j\in J_i} \widetilde{a_j} P_{j_i} \qquad j_i \in [1,n]$$

□

Ainsi, il est toujours possible d'"éliminer" les opérateurs autres que δ, les variables, le préfixage, le choix non déterministe et la récursion.

Remarque 1.2 (composition séquentielle)
Nous avons préféré le préfixage à la composition séquentielle, car celle-ci est moins simple à définir. En effet, une axiomatisation de la composition séquentielle correspondant à l'intuition doit comporter les axiomes :

$$\begin{aligned}(P+Q);R &\equiv P;R + Q;R\\ (P;Q);R &\equiv P;(Q;R)\\ \texttt{fin};P &\equiv P\end{aligned}$$

où `fin` représente un processus terminé. Il n'est cependant pas possible de choisir $\texttt{fin} = \delta$.

Considérons en effet le terme $P \stackrel{\text{def}}{=} (\widetilde{a}\,\delta + \delta);\widetilde{b}\,\delta$. Nous obtenons, en utilisant les axiomes de la composition séquentielle :

$$\begin{aligned}P &\equiv ((\widetilde{a}\,\delta);\widetilde{b}\,\delta) + \delta;(\widetilde{b}\,\delta)\\ &\equiv \widetilde{a}\,\widetilde{b}\,\delta + \widetilde{b}\,\delta\end{aligned}$$

D'autre part, en tenant compte de l'axiome $P + \delta \equiv P$, nous obtenons :

$$\begin{aligned} P &\equiv (\widetilde{a}\,\delta);\widetilde{b}\,\delta \\ &\equiv \widetilde{a}\,\widetilde{b}\,\delta \end{aligned}$$

Il n'est certainement pas souhaitable d'identifier les deux termes obtenus.

La méthode correcte pour définir la composition séquentielle consiste à spécifier que dans $P;Q$, le contrôle est transmis à Q lorsque P *signale* sa terminaison.

Par exemple, la composition séquentielle est définie dans ACP au moyen du processus "terminé" ϵ, qui exécute une pseudo-action $\surd$. La sémantique de la composition séquentielle est alors donnée par les deux règles

$$\frac{P \xrightarrow{a} P'}{P;Q \xrightarrow{a} P';Q} \qquad \frac{P \xrightarrow{\surd} P', Q \xrightarrow{\ell} Q'}{P;Q \xrightarrow{\ell} Q'}$$

On constate que cette définition ressemble fortement à celle de l'opérateur de continuation d'AUP : la pseudo-action $\surd$ correspond à l'action d'échappement ξ^1. Nous définissons donc le processus `fin` par

$$\texttt{fin} \stackrel{\text{def}}{=} \widetilde{\xi^1}\,\delta$$

En fait, la continuation est plus générale que la composition séquentielle, puisqu'elle permet de choisir le processus auquel le contrôle est transmis en fonction du niveau d'échappement.

Il existe deux différences entre la continuation d'AUP et la composition séquentielle d'ACP :

- Dans AUP, une action interne τ est exécutée au moment de la transmission du contrôle, alors que celui-ci est instantané dans ACP.
- Si le premier membre d'une continuation est constitué d'une composition parallèle, le contrôle est transmis dès qu'une des composantes exécute une action d'échappement. Dans ACP, une composition parallèle est terminée lorsque toutes ses composantes le sont.

□

Chapitre 2

L'algèbre de processus temporisés ATP

Ce chapitre est consacré à l'étude de l'algèbre ATP [NRSV90, NS90], obtenue par extension d'AUP au moyen d'opérateurs exprimant des *contraintes temporelles* sur les processus.

L'objectif est d'obtenir ainsi des processus dont le comportement est conditionné par le passage du temps. Ce comportement étant représenté par le système de transitions étiquetées déduit de la sémantique opérationnelle, il s'ensuit que le passage du temps doit être explicite dans les systèmes de transitions.

Nous choisissons dans ce chapitre une approche extrêmement simple : le passage du temps est observé de manière *discrète*. Une *base de temps* $\mathtt{B_t}$ est définie comme une fraction de l'unité de temps $\mathtt{U_t}$ (cette dernière pouvant être la seconde, la milliseconde, ...) Nous ne pouvons observer que des intervalles de temps correspondant à des multiples de $\mathtt{B_t}$. Un événement χ correspond à la fin d'un intervalle de durée $\mathtt{B_t}$. Nous supposons dans ce chapitre que $\mathtt{B_t} = \mathtt{U_t}$. Nous identifions donc une durée $\mathtt{B_t}$ et une unité de temps.

L'événement χ est ajouté à l'ensemble des étiquettes de transitions $\mathcal{L}$. Le nouvel ensemble d'étiquettes $\mathcal{L} \cup \{\chi\}$ est noté $\mathcal{L}_t$. Un *modèle temporisé* est un système de transitions étiquetées par des éléments de $\mathcal{L}_t$.

Une transition $P \xrightarrow{\chi} Q$ signifie qu'au bout d'une durée $\mathtt{B_t}$ depuis le dernier événement χ (ou depuis le début du processus si aucun χ n'a encore été observé), un nouvel événement χ est observé, et le comportement du processus devient celui de Q.

Nous exigeons le *déterminisme* des transitions χ : un processus peut à tout moment exécuter au plus une transition χ. Le passage du temps ne peut donc modifier un comportement que d'une seule façon.

Plusieurs possibilités existent pour choisir les nouveaux opérateurs. L'approche choisie est sans doute la plus simple. Elle est suggérée par l'étude de la classe des modèles d'ATP que nous souhaitons obtenir.

Cette classe doit contenir tous les systèmes de transitions étiquetées par les éléments de $\mathcal{L}_t$, déterministes par rapport aux transitions χ.

En premier lieu, il faut des états sans transition χ. Un tel état, qui n'a que des transitions

étiquetées par des actions, illustre une situation d'*urgence* : une action doit être exécutée "immédiatement", c'est-à-dire avant la fin de l'intervalle de durée $\mathtt{B_t}$ courant. L'urgence est spécifiée par le biais d'un opérateur de *préfixage immédiat* par une action.

En second lieu, il faut des états comportant une transition χ menant vers un autre état. Cela correspond à une situation où la fin d'un intervalle de durée $\mathtt{B_t}$ provoque un changement de comportement. Un opérateur de *délai unitaire* permet de spécifier une telle contrainte.

Dans le chapitre 4, nous décrivons une algèbre dont le domaine temporel est quelconque (dense ou discret). Le préfixage immédiat peut être défini quel que soit le domaine temporel. Par contre, lorsque ce dernier est dense, la notion de base de temps disparaît, et avec elle celle de délai unitaire. Il faut alors adopter un autre opérateur.

2.1 Principes de l'extension d'AUP

Remarque 2.1

Pour plus de clarté, nous discutons ici des algèbres AUP et ATP, mais l'exposé est en fait valable pour $\mathcal{P}_u$ et son correspondant temporisé $\mathcal{P}_t$. Les principes généraux de l'extension d'une algèbre non temporisée par ajout d'opérateurs temporels sont présentés dans [NS91].

□

Formellement, nous considérons l'algèbre

$$\text{AUP} = (O_u, \mathcal{L}, R_{\mathcal{L}}^{O_u}, \overset{\mathcal{L}}{\sim})$$

où

- O_u est l'ensemble des opérateurs d'AUP ;
- $R_{\mathcal{L}}^{O_u}$ est l'ensemble des règles de sémantique des éléments de O_u définissant des transitions étiquetées par les éléments de $\mathcal{L}$;
- $\overset{\mathcal{L}}{\sim}$ est l'équivalence forte sur AUP (notée $\sim$ au chapitre 1).

ATP est définie à partir d'AUP par

$$\text{ATP} = (O_u \cup O_t, \mathcal{L}_t, R_{\mathcal{L}_t}^{O_u \cup O_t}, \overset{\mathcal{L}_t}{\sim})$$

où

- O_t est un ensemble d'opérateurs décrivant des contraintes temporelles ;
- $R_{\mathcal{L}_t}^{O_u \cup O_t}$ décrit la sémantique des opérateurs d'ATP au moyen de transitions étiquetées par $\mathcal{L}_t$;
- $\overset{\mathcal{L}_t}{\sim}$ est l'équivalence forte sur ATP.

Dans AUP, le passage du temps n'a pas à être représenté dans les modèles, puisque par définition il n'a pas d'influence sur le comportement d'un processus non temporisé. La première étape de la construction d'ATP consiste donc à redéfinir la sémantique d'AUP en intégrant l'événement χ. Cela revient à associer à chaque modèle non temporisé d'AUP un *équivalent temporisé*.

Cette extension de la sémantique doit bien entendu préserver les propriétés des processus non temporisés. Pour cela, il faut satisfaire aux exigences suivantes :

Conservation de la sémantique : un modèle non temporisé et son équivalent temporisé doivent définir les mêmes comportement, quand seuls les éléments de $\mathcal{L}$ sont observés. Plus formellement, un modèle non temporisé doit être *observationnellement équivalent* [Mil80, Mil89] à son équivalent temporisé, dans lequel les éléments de $\mathcal{L}_t - \mathcal{L}$ sont non visibles. Cette propriété est garantie si on exige, d'une part que les règles de $R_{\mathcal{L}}^{O_u}$ restent valides dans ATP, *lorsqu'elles sont appliquées aux termes d'*AUP, et d'autre part qu'une transition par χ ne change pas le comportement des termes d'AUP.

Isomorphisme : pour tous processus P et Q définis avec les opérateurs d'AUP, P et Q doivent être fortement équivalents dans ATP si et seulement si ils sont fortement équivalents dans AUP, c'est-à-dire :

$$P \overset{\mathcal{L}}{\sim} Q \iff P \overset{\mathcal{L}_t}{\sim} Q$$

Ceci garantit que la théorie de processus d'AUP est isomorphe à celle de la restriction d'ATP aux opérateurs d'AUP. Par conséquent, tous les développements théoriques concernant les processus non temporisés effectués dans AUP restent valides dans ATP, et vice-versa.

La section suivante décrit la méthode suivie pour étendre AUP et obtenir l'algèbre des processus temporisés ATP, en appliquant ces principes. La restriction à AUP (remarque 2.1) ne s'applique plus : nous devons considérer les algèbres générales $\mathcal{P}_u$ et $\mathcal{P}_t$. Nous ne cherchons cependant pas à satisfaire les deux exigences ci-dessus pour toute l'algèbre $\mathcal{P}_u$: l'exigence d'isomorphisme s'applique à $\mathcal{P}_u^r$, tandis que celle de conservation de la sémantique n'est valable que pour AUP. Ces restrictions sont sans réelle conséquence : d'un point de vue pratique, seuls les modèles des processus nous intéressent. Les axiomes doivent eux être préservés pour $\mathcal{P}_u^r$, si on souhaite que le théorème de complétude du chapitre précédent reste valide.

2.2 De l'algèbre $\mathcal{P}_u$ vers l'algèbre $\mathcal{P}_t$

2.2.1 Sémantique opérationnelle des termes de $\mathcal{P}_u$

Nous décrivons en premier lieu l'introduction du temps dans les processus non temporisés, en prenant en compte les deux exigences d'isomorphisme (pour $\mathcal{P}_u^r$) et de conservation de la sémantique (pour AUP).

2.2.1.1 Isomorphisme

La condition d'isomorphisme est trivialement satisfaite pour tout $\mathcal{P}_u$, si on impose que les termes de $\mathcal{P}_u$ peuvent toujours laisser passer le temps, sans changer de comportement. Cela revient à ajouter une transition bouclante χ en chaque état des systèmes de transitions étiquetées : tout terme P possède une transition $P \xrightarrow{\chi} P$. De telles transitions sont définies par la règle de sémantique

$$[{\mathcal{P}_u}^{\chi}] : \frac{P \in \mathcal{P}_u}{P \xrightarrow{\chi} P}$$

Étant donné qu'il est possible de déterminer syntaxiquement qu'un terme est construit en utilisant seulement les opérateurs de $\mathcal{P}_u$, cette règle permet effectivement d'associer une transition bouclante χ à chaque état du modèle d'un terme de $\mathcal{P}_u$. Deux considérations nous empêchent cependant de l'accepter sous cette forme dans la sémantique de $\mathcal{P}_u$:

- Afin de garantir que $\overset{\mathcal{L}_t}{\approx}$ est une congruence sur $\mathcal{P}_t$, nous souhaitons obtenir des règles de sémantique au format GSOS, ce qui n'est pas le cas ici.
- Rien ne permet de supposer qu'il n'est pas possible de définir dans $\mathcal{P}_t$ (en utilisant aussi les opérateurs de O_t) des termes dont le comportement intuitif est identique à celui de termes de $\mathcal{P}_u$. La sémantique de $\mathcal{P}_t$ doit donc produire pour ces termes des modèles dont chaque état comporte une boucle χ. Il est donc nécessaire d'avoir des règles (au format GSOS) permettant d'obtenir de tels modèles dans $\mathcal{P}_t$. La règle $[{\mathcal{P}_u}^{\chi}]$ fait alors double emploi avec ces dernières.

Au lieu de $[{\mathcal{P}_u}^{\chi}]$, nous introduisons donc les huit règles suivantes, qui prennent en compte ces considérations :

Règles temporelles des opérateurs de $\mathcal{P}_u$		
$[\delta^{\chi}] : \dfrac{}{\delta \xrightarrow{\chi} \delta}$	$[\widetilde{a}^{\chi}] : \dfrac{}{\widetilde{a}\,P \xrightarrow{\chi} \widetilde{a}\,P}$	$[X^{\chi}] : \dfrac{}{X \xrightarrow{\chi} \delta}$
$[+^{\chi}] : \dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P+Q \xrightarrow{\chi} P'+Q'}$	$[\|^{\chi}] : \dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P\|Q \xrightarrow{\chi} P'\|Q'}$	$[\triangledown^{\chi}] : \dfrac{P \xrightarrow{\chi} P'}{P \triangledown Q \xrightarrow{\chi} P' \triangledown Q}$
$[\partial^{\chi}] : \dfrac{P \xrightarrow{\chi} P'}{\partial_H(P) \xrightarrow{\chi} \partial_H(P')}$	$[\mathtt{rec}^{\chi}] : \dfrac{P \xrightarrow{\chi} P'}{\mathtt{rec}X \cdot P \xrightarrow{\chi} P'[\mathtt{rec}X \cdot P/X]}$	

On constate aisément que ces règles sont dans le format GSOS, et que dans le cadre général de $\mathcal{P}_t$, elles respectent l'hypothèse de déterminisme des transitions χ.

La règle $[X^{\chi}]$ peut paraître surprenante. Elle s'explique par des considérations techniques exposées plus loin. La règle $X \xrightarrow{\chi} X$ n'est pas adéquate pour les termes temporisés. Si on l'avait adoptée, l'exigence d'isomorphisme aurait été satisfaite pour $\mathcal{P}_u$ tout entière. Celle que nous adoptons garantit clairement l'isomorphisme pour AUP, puisque ses termes n'ont pas de variable libre. En fait, l'isomorphisme reste également valable pour $\mathcal{P}_u^r$. Nous n'en donnons pas ici la preuve ; il suffit de vérifier que les axiomes de $\mathcal{P}_u^r$ sont valides dans $\mathcal{P}_t^r$, ce qui est fait à la section 2.4.

Par contre, on peut vérifier que l'équivalence forte n'est pas préservée pour $\mathcal{P}_u$, en considérant les deux termes

$$P \overset{\text{def}}{=} \mathtt{rec}X \cdot X \| \widetilde{a}\,\delta \quad \text{et} \quad Q \overset{\text{def}}{=} \mathtt{rec}Y \cdot \widetilde{a}\,Y$$

Ils sont équivalents dans $\mathcal{P}_u$, leurs modèles étant réduits à un état avec une transition bouclante par a.

Par contre, leurs modèles dans $\mathcal{P}_t$ ne sont pas fortement équivalents. Celui de Q est obtenu en ajoutant une boucle χ à son modèle dans $\mathcal{P}_u$. Il n'en va pas de même pour P. En effet, puisque $X \xrightarrow{\chi} \delta$, le terme $X\|\widetilde{a}\,\delta$ possède une transition par χ menant vers $\delta\|\widetilde{a}\,\delta$, qui est clairement équivalent à $\widetilde{a}\,\delta$. P possède donc une transition χ menant vers un état où une seule action a reste possible. On en déduit que P et Q ne sont pas équivalents dans $\mathcal{P}_u$.

Nous adoptons cependant sans restriction les règles ci-dessus pour la sémantique de $\mathcal{P}_t$, puisqu'elles suffisent à garantir l'exigence d'isomorphisme sur $\mathcal{P}_u^r$.

Les axiomes sont préservés, mais l'exigence de conservation de la sémantique n'est pas satisfaite pour $\mathcal{P}_u^r$. En effet, le terme $P \overset{\text{def}}{=} X + \widetilde{a}\,\delta$ possède une transition par X vers δ, une transition par a vers δ, mais également une transition par χ vers $\delta + \widetilde{a}\,\delta$. Ce dernier terme, contrairement à P, ne possède pas de transition par X. La transition χ modifie donc le "comportement" du terme

P. Cet effet ne survient que lorsque le terme comporte des variables libres ; il n'apparaît donc pas dans le cas d'AUP.

2.2.1.2 Conservation de la sémantique

Il s'agit à présent d'assurer la préservation, dans la sémantique de $\mathcal{P}_t$, des règles de $\mathcal{P}_u$ pour les termes de AUP. Il faut donc avoir des règles permettant de déduire, par exemple :

$$\frac{P, P', Q \in \mathcal{P}_u, P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$$

De même que pour $[\mathcal{P}_u{}^{\chi}]$, des règles de cette forme ne sont pas acceptables pour la sémantique de $\mathcal{P}_t$. Afin d'assurer la conservation de la sémantique en restant dans le format GSOS, nous adoptons pour $\mathcal{P}_t$ toutes les règles de $\mathcal{P}_u$ (section 1.3.1, page 25).

La sémantique opérationnelle dans $\mathcal{P}_t$ des termes de $\mathcal{P}_u$ est à présent complètement définie. À titre d'exemple, la figure 2.1 présente les modèles temporisés des termes dont les modèles dans $\mathcal{P}_u$ se trouvent figure 1.1 (page 29). En comparant les deux figures, on vérifie que les modèles temporisés sont obtenus en ajoutant une boucle χ en chaque état, sauf lorsqu'une variable est présente.

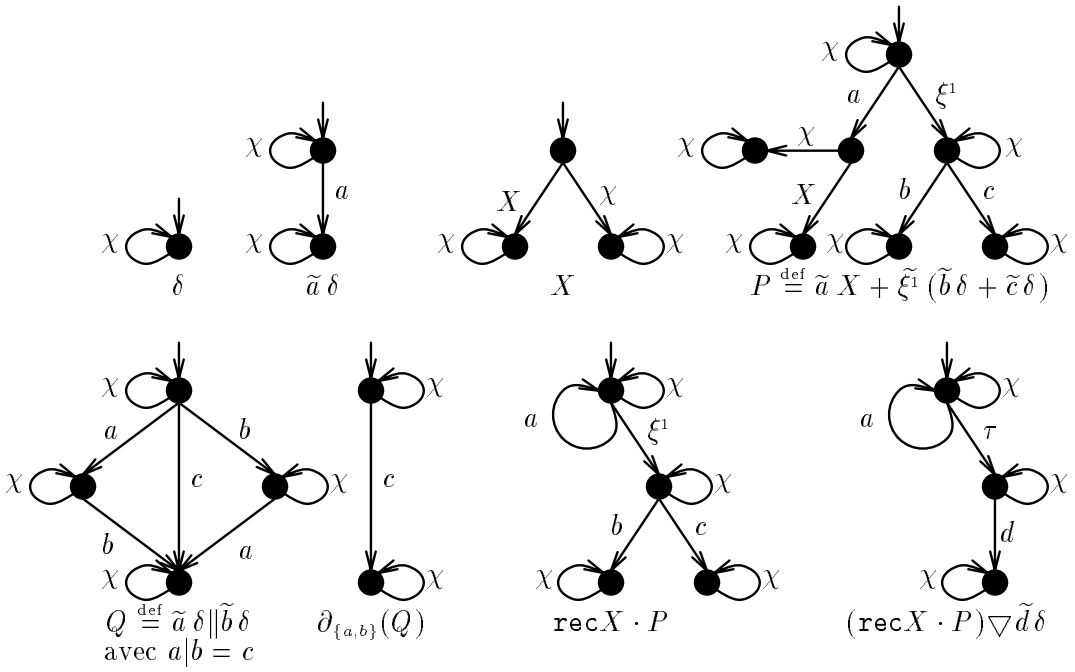


Figure 2.1 : modèles temporisés de termes de $\mathcal{P}_u$

2.2.2 Faut-il d'autres règles temporelles pour les opérateurs de $\mathcal{P}_u$?

Les règles temporelles des opérateurs $+, \|, \triangledown, \partial$ et `rec` définissent les transitions par χ d'un terme composé lorsque ses composants possèdent une transition χ. La question se pose à présent de

savoir s'il faut ajouter d'autres règles lorsque les composants sont en situation d'urgence (pas de transition χ).

Nous examinons successivement chacun de ces opérateurs.

Composition parallèle. Il est clairement hors de question d'accepter d'autres règles que celle déjà présentée. En effet, l'hypothèse principale de l'ensemble de cette étude stipule que la notion de temps est commune à toutes les composantes parallèles d'un système. Le passage du temps doit donc s'effectuer de manière synchrone. Or, la règle $[\|^{\chi}]$ se lit :

> "une durée unité de temps est écoulée dans une composition parallèle lorsqu'elle est écoulée dans chacune de ses composantes"

Elle traduit donc exactement l'hypothèse de temps global. Ajouter une règle de la forme

$$\frac{P \xrightarrow{\chi} P', Q \not\xrightarrow{\chi}}{P\|Q \xrightarrow{\chi} P'\|Q}$$

reviendrait à contredire cette hypothèse.

Continuation. Le terme $P \triangledown Q$ se comporte comme P tant que celui-ci n'exécute pas d'action d'échappement. L'introduction du temps ne modifie en rien cette définition. Une situation d'urgence pour P conduit à une situation d'urgence pour $P \triangledown Q$. Il n'y a donc pas de règle particulière à introduire dans ce cas.

Récursion. Un argument similaire est valable pour un terme $\mathtt{rec}X \cdot P$. Nous n'ajoutons donc aucune nouvelle règle.

Choix non déterministe. On peut supposer qu'une situation d'urgence de l'un des opérandes est supprimée si l'autre opérande peut laisser passer le temps. Il y a deux façons d'exprimer ce "relâchement" de l'urgence :

- La première se traduit par les règles :

$$\frac{P \xrightarrow{\chi} P', Q \not\xrightarrow{\chi}}{P + Q \xrightarrow{\chi} P'} \qquad \frac{Q \xrightarrow{\chi} Q', P \not\xrightarrow{\chi}}{P + Q \xrightarrow{\chi} Q'}$$

  Après un χ, le comportement global correspond donc à celui de l'opérande non urgent. Cette sémantique définit un *choix faible* [MT90]. Son inconvénient est qu'elle invalide l'axiome [+4] d'AUP ($P + \delta \equiv P$) : si P n'a pas de transition χ, alors $P + \delta$ en a une, le terme résultant étant δ.

- La seconde est définie par les règles :

$$\frac{P \xrightarrow{\chi} P', Q \not\xrightarrow{\chi}}{P + Q \xrightarrow{\chi} P' + Q} \qquad \frac{Q \xrightarrow{\chi} Q', P \not\xrightarrow{\chi}}{P + Q \xrightarrow{\chi} P + Q'}$$

  Cela correspond à un relâchement de l'urgence, non seulement sur le terme composé, mais aussi sur la composante urgente. L'axiome [+4] reste cette fois valide. Cette sémantique introduit cependant une fonctionnalité du choix non déterministe qu'on peut considérer comme parasite. Il est préférable d'exprimer le relâchement de l'urgence d'un processus en utilisant un opérateur temporel spécifique (voir le chapitre suivant).

Ces choix de sémantique sont donc fortement discutables, ils introduisent des règles supplémentaires qui n'apportent rien à l'expressivité de l'algèbre. Nous nous en tenons donc à la règle $[+^{\chi}]$.

Encapsulation. D'après la sémantique décrite au chapitre précédent, l'encapsulation par un ensemble d'action H, d'un processus P qui ne peut exécuter que des actions de H, "bloque" P. Le processus résultant est alors équivalent à δ : la terminaison et le blocage sont indistinguables. Cela reste vrai dans $\mathcal{P}_t$ pour les termes de $\mathcal{P}_u$. Si on souhaite qu'il en aille de même pour tous les termes de $\mathcal{P}_t$, il faut introduire la règle

$$\frac{\forall \ell \notin H,\ P \stackrel{\ell}{\not\longrightarrow}}{\partial_H(P) \stackrel{\chi}{\longrightarrow} \delta}$$

Cette règle correspond au format GSOS : il y a une instance de règle par ensemble H, et le nombre de prémisses est le cardinal de $\mathcal{A} - H$ (qui peut être infini).

Si nous ne l'ajoutons pas, alors l'encapsulation d'un terme qui n'a que des transitions étiquetées par des éléments de H (en particulier qui n'a pas de transition χ) résulte en un comportement sans aucune transition. Cela permet de distinguer de manière simple la terminaison du blocage : un processus bloqué ne peut, ni exécuter une action, *ni laisser passer le temps*, alors qu'un processus terminé *ne peut que laisser passer le temps indéfiniment.* En d'autres termes, la capacité d'observer le comportement temporel d'un processus permet d'effectuer la distinction entre terminaison et blocage.

Le blocage ne survient que lorsqu'un processus *urgent* est encapsulé : il *doit* exécuter immédiatement une action, et on l'en empêche. En revanche, un processus non urgent auquel on interdit d'exécuter ses actions a toujours la possibilité de laisser passer le temps : il n'est pas bloqué.

Nous n'incorporons donc pas la règle ci-dessus à la sémantique de $\mathcal{P}_t$, ce qui augmente l'expressivité de l'algèbre.

Il est important de remarquer que le blocage ainsi défini représente un comportement non implémentable, puisqu'il bloque le temps indéfiniment. C'est également le cas d'autres processus d'ATP. Avant d'implanter un processus d'ATP, il faut vérifier que de telles situations (qui sont de toutes façons anormales pour un système) ne peuvent survenir. Ce problème est discuté en détail dans la section 2.5 et au chapitre 6.

En conclusion, il ressort qu'il n'est pas nécessaire d'ajouter de règle temporelle supplémentaire pour les opérateurs de $\mathcal{P}_u$.

2.2.3 Opérateurs de $\mathcal{P}_t$

Comme nous l'avons expliqué dans l'introduction à ce chapitre, l'algèbre $\mathcal{P}_t$ est obtenue en ajoutant deux nouveaux opérateurs : le *préfixage immédiat* et le *délai unitaire.* Nous donnons à présent leur syntaxe et leur sémantique opérationnelle. Celle du délai unitaire nous conduit ensuite à redéfinir les transitions par une variable en distinguant deux cas.

2.2.3.1 Préfixage immédiat

Le préfixage immédiat d'un terme P par une action a est noté $a\,P$. Sa sémantique est donnée par l'unique règle :

$$[a^a] : \frac{}{a\,P \stackrel{a}{\longrightarrow} P}$$

Le terme $a\,P$ ne peut donc qu'exécuter l'action a, et il doit le faire immédiatement, n'ayant pas la possibilité de laisser passer le temps.

La substitution est définie sur un tel terme par $(a\,P)[Q/X] \stackrel{\text{def}}{=} a\,(P[Q/X])$.

Le processus `fin` est redéfini pour signaler immédiatement la terminaison :

$$\texttt{fin} \stackrel{\text{def}}{=} \xi^1\,\delta$$

Définition 2.1 (processus bloqué)
L'opérateur de préfixage immédiat permet de définir le *processus bloqué* **0** par :

$$\mathbf{0} \stackrel{\text{def}}{=} \partial_{\{a\}}(a\,P)$$

où a est une action quelconque, et P un terme quelconque de $\mathcal{P}_t$. Quels que soient a et P, on constate que le processus **0** n'a aucune transition.
□

2.2.3.2 Délai unitaire

Le *délai unitaire de corps P et d'exception Q* est noté $\lfloor P \rfloor Q$. Nous étendons la notion de variable gardée (définition 1.6, page 24), en déclarant que le préfixage immédiat (resp. le délai unitaire) garde son argument (resp. son second argument).

En ce qui concerne les actions et le passage du temps, sa sémantique est donnée par les trois règles suivantes :

Règle d'action	Règle temporelle	Règle de variable
$[\lfloor\rfloor^a] : \dfrac{P \xrightarrow{a} P'}{\lfloor P \rfloor Q \xrightarrow{a} P'}$	$[\lfloor\rfloor^\chi] : \dfrac{}{\lfloor P \rfloor Q \xrightarrow{\chi} Q}$	$[\lfloor\rfloor^X] : \dfrac{P \xrightarrow{X} P'}{\lfloor P \rfloor Q \xrightarrow{X} P'}$

Le terme $\lfloor P \rfloor Q$ se comporte donc comme P si celui-ci exécute une action avant une unité de temps. Dans le cas contraire, il se comporte au bout d'une unité de temps comme Q.

Le délai unitaire permet donc de décrire des termes dont le comportement peut varier en fonction du passage du temps.

La substitution est définie sur le délai unitaire par :

$$(\lfloor P \rfloor Q)[R/X] \stackrel{\text{def}}{=} \lfloor P[R/X] \rfloor Q[R/X]$$

On constate aisément que les deux propriétés 1.1 (le but d'une transition par X est δ) et 1.2 (reliant variables actives et non gardées, et variables activables et libres), page 27, restent vérifiées après introduction de ces opérateurs.

On peut définir le préfixage de $\mathcal{P}_u$ comme un opérateur dérivé, à l'aide du délai unitaire, du préfixage immédiat et de la récursion :

$$\forall a, P,\ \widetilde{a}\,P \stackrel{\text{def}}{=} \texttt{rec}X \cdot \lfloor a\,P \rfloor X$$

La terminaison δ est également un opérateur dérivé :

$$\delta \stackrel{\text{def}}{=} \texttt{rec}X \cdot \lfloor \mathbf{0} \rfloor X$$

En fait, il en allait de même dans le chapitre précédent : la terminaison peut être définie dans $\mathcal{P}_u$ par $\delta \stackrel{\text{def}}{=} \partial_{\{a\}}(\widetilde{a}\,P)$. Cette définition est équivalente dans $\mathcal{P}_t$ à celle donnée ci-dessus.

Les définitions formelles de **0** et δ sont en accord avec la discussion informelle de la page 41 concernant le blocage et la terminaison.

2.2.4 Les transitions par une variable

Une propriété essentielle de l'équivalence forte est qu'elle doit être préservée par substitution, c'est-à-dire, si $P \sim Q$, alors $P[R/X] \sim Q[R/X]$ pour tous R et X. En d'autres termes, il faut pouvoir définir sémantiquement la substitution au moyen de règles qui préservent la propriété de congruence de $\sim$, et montrer que les notions syntaxique et sémantique sont équivalentes. Ceci est en particulier nécessaire pour prouver la consistance d'un système d'axiomes.

Il n'y a pas de problème pour $\mathcal{P}_u^r$; ce n'est plus le cas une fois introduit le délai unitaire. En effet, considérons les deux termes

$$P \stackrel{\text{def}}{=} X + \lfloor \mathbf{0} \rfloor a\,\delta \quad \text{et} \quad Q \stackrel{\text{def}}{=} \lfloor X \rfloor a\,\delta$$

Ils sont fortement équivalents : ils ont tous les deux une transition par X vers δ, et ils ont une transition par χ, vers $\delta + a\,\delta$ pour P, et vers $a\,\delta$ pour Q, et ces deux termes sont équivalents.

Considérons à présent le terme $R \stackrel{\text{def}}{=} \lfloor \mathbf{0} \rfloor b\,\delta$. Nous avons alors :

$$P_1 \stackrel{\text{def}}{=} P[R/X] = \lfloor \mathbf{0} \rfloor b\,\delta + \lfloor \mathbf{0} \rfloor a\,\delta \quad \text{et} \quad Q_1 \stackrel{\text{def}}{=} Q[R/X] = \lfloor \lfloor \mathbf{0} \rfloor b\,\delta \rfloor a\,\delta$$

Les règles de sémantique donnent, pour P_1 une transition par χ vers $b\,\delta + a\,\delta$, et pour Q_1 une transition par χ vers $a\,\delta$. L'équivalence forte n'est donc pas préservée.

Si on observe cet exemple, on constate que les variables non gardées d'un terme de $\mathcal{P}_u^r$ peuvent être de deux sortes :

- elles apparaissent en dehors d'un corps de délai dans un terme qui a une transition par χ, comme c'est le cas pour P. Elles "participent" alors à la transition par χ de ce terme, en ce sens qu'une transition χ du terme qu'on leur substitue (ici R) intervient dans cette dernière ;
- elles n'apparaissent que dans un corps de délai, comme dans Q, ou dans un terme sans transition par χ, auquel cas la transition χ de R ne joue aucun rôle dans celle de Q_1.

Noter qu'un terme tel que $X + \lfloor X \rfloor S$ entre dans la première catégorie : il suffit pour cela que X apparaisse une fois en dehors d'un délai (et que le terme ait une transition par χ).

Cette différence de comportement de la substitution doit se traduire dans la sémantique : il faut distinguer différentes sortes de variables non gardées. D'où l'idée de définir deux catégories de transitions par une variable : $\xrightarrow{X^+}$ et $\xrightarrow{X^-}$, afin de distinguer les deux termes P et Q : P a alors une transition par X^+, et Q par X^-.

Il faut donc redéfinir la sémantique opérationnelle en tenant compte de ces deux catégories d'étiquettes.

Dans le cas de $\mathcal{P}_u$, toutes les transitions par une variable sont des transitions par X^+.

2.3 Syntaxe et sémantique opérationnelle

Nous résumons à présent les définitions de $\mathcal{P}_t$ et nous présentons sa sémantique opérationnelle en distinguant les deux cas de transition par une variable.

Nous n'incluons dans les termes de base, ni le préfixage d'AUP, ni la terminaison, puisqu'ils s'obtiennent comme opérateurs dérivés.

2.3.1 Syntaxe

L'algèbre $\mathcal{P}_t$ est définie par la syntaxe suivante :

$$\begin{array}{rcccccccc} P ::= & X & | & a\,P & | & P+P & | & \lfloor P \rfloor P & | \\ & P\|P & | & \partial_H(P) & | & P \bigtriangledown P & | & \mathtt{rec}X \cdot P & \end{array}$$

Les opérateurs dérivés sont :

$$\begin{array}{l} \text{Blocage : } \mathbf{0} \stackrel{\text{def}}{=} \partial_{\{a\}}(a\,P) \\ \text{Terminaison : } \delta \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \lfloor \mathbf{0} \rfloor X \\ \text{Préfixage : } \widetilde{a}\,P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot \lfloor a\,P \rfloor X \end{array}$$

Comme pour l'algèbre non temporisée, les deux restrictions de $\mathcal{P}_t$ sont définies par :

Définition 2.2 (les algèbres $\mathcal{P}_t^r$ et ATP)

(1) $\mathcal{P}_t^r$ est la sous-algèbre des termes réguliers de $\mathcal{P}_t$ (cf. définition 1.5, page 23).

(2) ATP est la sous-algèbre des termes fermés et bien gardés de $\mathcal{P}_t^r$.

□

Les termes d'ATP sont appelés *processus temporisés*, ou plus simplement *processus* lorsque aucune confusion n'est possible.

2.3.2 Sémantique opérationnelle

La sémantique opérationnelle décrit les modèles des termes de $\mathcal{P}_t$ en leur associant un système de transitions étiquetées par les éléments de $\mathcal{L}_t \stackrel{\text{def}}{=} \mathcal{A} \cup \{\chi\} \cup \mathcal{V}^+ \cup \mathcal{V}^-$, où

$$\mathcal{V}^+ \stackrel{\text{def}}{=} \{X^+, X \in \mathcal{V}\} \quad \text{et} \quad \mathcal{V}^- \stackrel{\text{def}}{=} \{X^-, X \in \mathcal{V}\}$$

Nous notons $X^?$ pour désigner indifféremment X^+ ou X^-. Lorsque $X^?$ apparaît dans les prémisses et la conclusion d'une règle, ils dénotent la même étiquette.

La relation d'équivalence des comportements $\overset{\mathcal{L}_t}{\sim}$ est l'équivalence forte sur $\mathcal{P}_t$. Nous la notons ici aussi $\sim$.

Le domaine de la relation de transition $\rightarrow$ est donc :

$$\rightarrow \subseteq \mathcal{P}_t \times \mathcal{L}_t \times \mathcal{P}_t$$

Elle est définie par les règles suivantes :

Règles d'action	
$[a^a] : \dfrac{}{a\,P \xrightarrow{a} P}$	$[+^a_l] : \dfrac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'} \quad [+^a_r] : \dfrac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$
$[\lfloor\rfloor^a] : \dfrac{P \xrightarrow{a} P'}{\lfloor P \rfloor Q \xrightarrow{a} P'}$	$[\|\|^a_l] : \dfrac{P \xrightarrow{a} P'}{P\|Q \xrightarrow{a} P'\|Q} \quad [\|\|^a_r] : \dfrac{Q \xrightarrow{a} Q'}{P\|Q \xrightarrow{a} P\|Q'}$
$[\partial^a] : \dfrac{P \xrightarrow{a} P',\ a \notin H}{\partial_H(P) \xrightarrow{a} \partial_H(P')}$	$[\|\|^a_c] : \dfrac{P \xrightarrow{a} P', Q \xrightarrow{b} Q',\ a\|b \neq \bot}{P\|Q \xrightarrow{a\|b} P'\|Q'}$
$[\bigtriangledown^a_a] : \dfrac{P \xrightarrow{a} P',\ a \notin \Xi}{P \bigtriangledown Q \xrightarrow{a} P' \bigtriangledown Q} \quad [\bigtriangledown^a_\xi] : \dfrac{P \xrightarrow{\xi^{n+1}} P'}{P \bigtriangledown Q \xrightarrow{\xi^n} Q}$	$[\mathtt{rec}^a] : \dfrac{P \xrightarrow{a} P'}{\mathtt{rec}X \cdot P \xrightarrow{a} P'[\mathtt{rec}X \cdot P/X]}$

Règles temporelles

$$[X^\chi] : \frac{}{X \xrightarrow{\chi} \delta} \qquad [+^\chi] : \frac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P + Q \xrightarrow{\chi} P' + Q'} \qquad [\lfloor\rfloor^\chi] : \frac{}{\lfloor P \rfloor Q \xrightarrow{\chi} Q} \qquad [\|^\chi] : \frac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P \| Q \xrightarrow{\chi} P' \| Q'}$$

$$[\partial^\chi] : \frac{P \xrightarrow{\chi} P'}{\partial_H(P) \xrightarrow{\chi} \partial_H(P')} \qquad [\triangledown^\chi] : \frac{P \xrightarrow{\chi} P'}{P \triangledown Q \xrightarrow{\chi} P' \triangledown Q} \qquad [\mathtt{rec}^\chi] : \frac{P \xrightarrow{\chi} P'}{\mathtt{rec}X \cdot P \xrightarrow{\chi} P'[\mathtt{rec}X \cdot P/X]}$$

Règles de variable

$$[X^{X^+}] : \frac{}{X \xrightarrow{X^+} \delta} \qquad [+_l^{X^+}] : \frac{P \xrightarrow{X^+} P', Q \xrightarrow{\chi} Q'}{P + Q \xrightarrow{X^+} P'} \qquad [+_r^{X^+}] : \frac{Q \xrightarrow{X^+} Q', P \xrightarrow{\chi} P'}{P + Q \xrightarrow{X^+} Q'}$$

$$[\lfloor\rfloor^{X^-}] : \frac{P \xrightarrow{X^?} P'}{\lfloor P \rfloor Q \xrightarrow{X^-} P'} \qquad [+_l^{X^-}1] : \frac{P \xrightarrow{X^+} P', Q \not\xrightarrow{\chi}}{P + Q \xrightarrow{X^-} P'} \qquad [+_r^{X^-}1] : \frac{Q \xrightarrow{X^+} Q', P \not\xrightarrow{\chi}}{P + Q \xrightarrow{X^-} Q'}$$

$$[\partial^X] : \frac{P \xrightarrow{X^?} P'}{\partial_H(P) \xrightarrow{X^?} P'} \qquad [+_l^{X^-}2] : \frac{P \xrightarrow{X^-} P', Q \not\xrightarrow{X^+}}{P + Q \xrightarrow{X^-} P'} \qquad [+_r^{X^-}2] : \frac{Q \xrightarrow{X^-} Q', P \not\xrightarrow{X^+}}{P + Q \xrightarrow{X^-} Q'}$$

$$[\|_l^{X^+}] : \frac{P \xrightarrow{X^+} P', Q \xrightarrow{\chi} Q'}{P \| Q \xrightarrow{X^+} P'} \qquad [\|_l^{X^-}1] : \frac{P \xrightarrow{X^+} P', Q \not\xrightarrow{\chi}}{P \| Q \xrightarrow{X^-} P'} \qquad [\|_l^{X^-}2] : \frac{P \xrightarrow{X^-} P', Q \not\xrightarrow{X^+}}{P \| Q \xrightarrow{X^-} P'}$$

$$[\|_r^{X^+}] : \frac{Q \xrightarrow{X^+} Q', P \xrightarrow{\chi} P'}{P \| Q \xrightarrow{X^+} Q'} \qquad [\|_r^{X^-}1] : \frac{P \xrightarrow{X^+} P', Q \not\xrightarrow{\chi}}{P \| Q \xrightarrow{X^-} P'} \qquad [\|_r^{X^-}2] : \frac{Q \xrightarrow{X^-} Q', P \not\xrightarrow{X^+}}{P \| Q \xrightarrow{X^-} Q'}$$

$$[\triangledown^X] : \frac{P \xrightarrow{X^?} P'}{P \triangledown Q \xrightarrow{X^?} P'} \qquad [\mathtt{rec}^X] : \frac{P \xrightarrow{Y^?} P'}{\mathtt{rec}X \cdot P \xrightarrow{Y^?} P'} \; Y \neq X$$

Toutes ces règles sont dans le format GSOS. Comme pour les termes non temporisés, la proposition suivante est donc vraie en application du théorème énoncé dans [Gro89].

Proposition 2.1

L'équivalence forte est une congruence pour les opérateurs de $\mathcal{P}_t$.

□

Les propositions 1.1 et 1.2 (page 27) restent valides dans $\mathcal{P}_t$. Elles se montrent aisément par induction sur la structure des termes. Il faut bien sûr les considérer avec $X^?$ au lieu de X pour les transitions.

En outre, on peut facilement prouver la proposition suivante :

Proposition 2.2

Pour tout terme P et toute variable X,

(1) $P \xrightarrow{X^+} \delta \;\Rightarrow\; P \not\xrightarrow{X^-}$

(2) $P \xrightarrow{X^+} \delta \;\Rightarrow\; \exists R,\ P \xrightarrow{\chi} R$

□

Les modèles des termes de $\mathcal{P}_t$ sont définis de la même manière que ceux de $\mathcal{P}_u$, comme des systèmes de transitions étiquetées par des éléments de $\mathcal{L}_t$, dont les états sont des classes d'équivalence de $\sim$, et la relation de transition est celle induite par $\rightarrow$ sur l'algèbre-quotient $\mathcal{P}_t/\sim$. À titre d'exemple, les modèles (finis) de quelques termes sont présentés sur la figure 2.2. On constate sur le modèle du terme $\lfloor P \rfloor (Y + d\,\mathbf{0})$ que lorsque deux délais unitaires sont imbriqués, l'expiration des délais (transition χ) déclenche l'exception la plus externe.

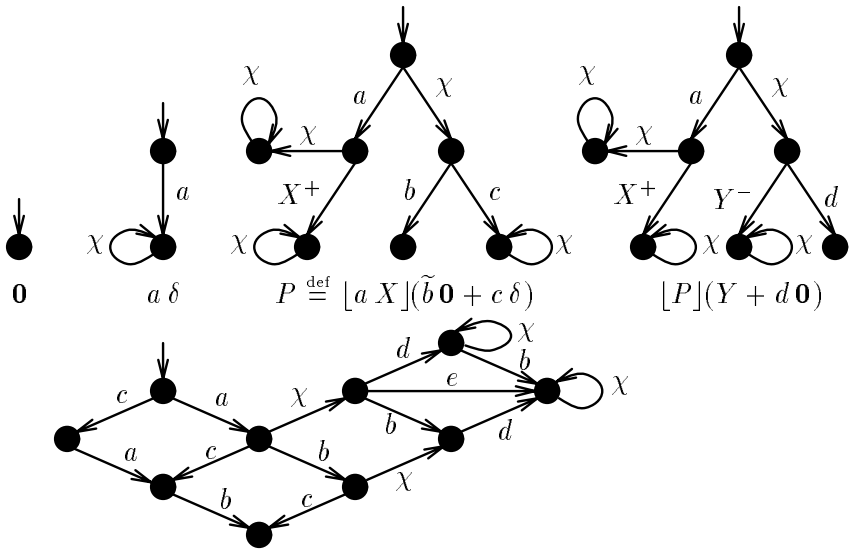


Figure 2.2 : modèles de termes de $\mathcal{P}_t$

L'exemple de la composition parallèle illustre les divers cas de figure pouvant se présenter lors de la composition de termes urgents et non urgents.

On peut montrer que la classe des modèles de $\mathcal{P}_t^r$ est l'ensemble des systèmes de transitions finis, à branchement fini, déterministes par rapport aux transitions χ, et dont les états buts des transitions par une variable comportent uniquement une boucle χ. Pour ATP, la classe des modèles est le sous-ensemble des modèles de $\mathcal{P}_t^r$ qui ne comportent pas de transition par une variable. La preuve de ces propositions est présentée à la fin de la section suivante, en utilisant la forme canonique des termes.

2.4 Axiomatisation de $\mathcal{P}_t^r$

Nous présentons à présent un ensemble d'axiomes définissant une congruence sur $\mathcal{P}_t^r$, et nous prouvons que cette congruence est identique à l'équivalence forte sur $\mathcal{P}_t^r$, c'est-à-dire que l'axiomatisation est complète. Les différentes preuves de cette section sont inspirées de [Mil84].

L'axiomatisation de la composition parallèle nécessite une fois encore l'utilisation des deux opérateurs de composition à gauche et de communication. Leur sémantique est définie par les règles suivantes (nous reprenons les règles d'action de $\mathcal{P}_u^r$; les règles de variable sont inutiles, les termes de $\mathcal{P}_t^r$ étant réguliers) :

Règles d'action	Règles temporelles
$[\lfloor\!\lfloor^a] : \dfrac{P \xrightarrow{a} P'}{P \lfloor\!\lfloor Q \xrightarrow{a} P' \lfloor\!\lfloor Q}$	$[\lfloor\!\lfloor^\chi] : \dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P \lfloor\!\lfloor Q \xrightarrow{\chi} P' \lfloor\!\lfloor Q'}$
$[\mid^a] : \dfrac{P \xrightarrow{a} P', Q \xrightarrow{b} Q'}{P \mid Q \xrightarrow{a\mid b} P' \| Q'} a\mid b \neq \perp$	$[\mid^\chi] : \dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P \mid Q \xrightarrow{\chi} P' \mid Q'}$

La composition à gauche doit donc exécuter une action de son premier argument avant toute action du second. Le passage du temps ne modifie pas cette contrainte. De même, la communication doit exécuter en premier lieu une communication entre ses deux arguments, indépendamment du passage du temps.

La relation $\equiv \subseteq \mathcal{P}_t^r \times \mathcal{P}_t^r$ est définie comme la plus petite congruence induite par les axiomes suivants :

[+1]	$P + Q \equiv Q + P$	[$\lfloor\rfloor$]	$\lfloor\lfloor P\rfloor Q\rfloor R \equiv \lfloor P\rfloor R$
[+2]	$P + (Q + R) \equiv (P + Q) + R$	[|1]	$P\|Q \equiv Q\|P$
[+3]	$P + P \equiv P$	[|2]	$(P + Q)\|R \equiv P\|R + Q\|R$
[+4]	$P + \delta \equiv P$	[|3]	$\mathbf{0}\|P \equiv \mathbf{0}$
[+5]	$\mathbf{0} + a\,P \equiv a\,P$	[|4]	$(a\,P)\|(b\,Q) \equiv \mathbf{0}$ si $a\|b = \bot$
[+6]	$\mathbf{0} + \lfloor P\rfloor Q \equiv \mathbf{0} + P$	[|5]	$(a\,P)\|(b\,Q) \equiv a\|b\,(P \parallel Q)$ sinon
[+7]	$\lfloor P_1\rfloor Q_1 + \lfloor P_2\rfloor Q_2 \equiv \lfloor P_1 + P_2\rfloor(Q_1 + Q_2)$	[|6]	$(a\,P)\|\lfloor Q\rfloor R \equiv (a\,P)\|Q$
[+8]	$P + \lfloor P\rfloor Q \equiv P + \lfloor\mathbf{0}\rfloor Q$	[|7]	$\lfloor P_1\rfloor Q_1 \mid \lfloor P_2\rfloor Q_2 \equiv \lfloor P_1\|P_2\rfloor(Q_1\|Q_2)$
[$\|\|$]	$P\|\|Q \equiv P\mathbin{\lfloor\!\lfloor} Q + Q\mathbin{\lfloor\!\lfloor} P + P\|Q$	[$\triangledown$1]	$(P + Q)\triangledown R \equiv P\triangledown R + Q\triangledown R$
[$\lfloor\!\lfloor$1]	$(P + Q)\mathbin{\lfloor\!\lfloor} R \equiv P\mathbin{\lfloor\!\lfloor} R + Q\mathbin{\lfloor\!\lfloor} R$	[$\triangledown$2]	$\mathbf{0}\triangledown P \equiv \mathbf{0}$
[$\lfloor\!\lfloor$2]	$\mathbf{0}\mathbin{\lfloor\!\lfloor} P \equiv \mathbf{0}$	[$\triangledown$3]	$(a\,P)\triangledown Q \equiv a\,(P\triangledown Q)$ si $a \notin \Xi$
[$\lfloor\!\lfloor$3]	$(a\,P)\mathbin{\lfloor\!\lfloor} Q \equiv a\,(P\|\|Q)$	[$\triangledown$4]	$(\xi^{n+1}\,P)\triangledown Q \equiv \xi^n\,Q$
[$\lfloor\!\lfloor$4]	$\lfloor P\rfloor Q \mathbin{\lfloor\!\lfloor}(\mathbf{0} + R) \equiv P\mathbin{\lfloor\!\lfloor}(\mathbf{0} + R)$	[$\triangledown$5]	$(\lfloor P\rfloor Q)\triangledown R \equiv \lfloor P\triangledown R\rfloor(Q\triangledown R)$
[$\lfloor\!\lfloor$5]	$\lfloor P_1\rfloor Q_1 \mathbin{\lfloor\!\lfloor} \lfloor P_2\rfloor Q_2 \equiv \lfloor P_1\mathbin{\lfloor\!\lfloor} P_1\rfloor(Q_1\mathbin{\lfloor\!\lfloor} Q_2)$	[rec1]	$\mathtt{rec}X \cdot P \equiv P[\mathtt{rec}X \cdot P/X]$
[∂1]	$\partial_H(P + Q) \equiv \partial_H(P) + \partial_H(Q)$	[rec2]	si $P[Q/X] \equiv Q$
[∂2]	$\partial_H(\mathbf{0}) \equiv \mathbf{0}$		et X est gardé dans P
[∂3]	$\partial_H(a\,P) \equiv \mathbf{0}$ si $a \in H$		alors $\mathtt{rec}X \cdot P \equiv Q$
[∂4]	$\partial_H(a\,P) \equiv a\,\partial_H(P)$ si $a \notin H$	[rec3]	$\mathtt{rec}X \cdot (X + P) \equiv \mathtt{rec}X \cdot P$
[∂5]	$\partial_H(\lfloor P\rfloor Q) \equiv \lfloor\partial_H(P)\rfloor\partial_H(Q)$	[rec4]	$\mathtt{rec}X \cdot \lfloor X + P\rfloor Q \equiv \mathtt{rec}X \cdot \lfloor P\rfloor Q$

Les axiomes de $\mathcal{P}_u$ qui ne se trouvent pas dans cette liste peuvent tous être déduits de ceux-ci.

Ces équations résument l'essence des propriétés des opérateurs :

- Les axiomes du choix non déterministe décrivent la structure de monoïde de $(\mathcal{P}_t^r, +)$, ainsi que l'idempotence de $+$. L'axiome [+6] (en combinaison avec [+5]) indique de plus la priorité de l'urgence sur la non urgence. [+7] décrit le déterminisme temporel. L'axiome [+8] ne peut être déduit des autres ; il est essentiel pour obtenir une forme canonique dans le cas où P est une variable (voir annexe B).
- L'axiome de la composition parallèle permet de la décomposer en composition à gauche et communication. $P\|Q$ doit exécuter initialement, soit une action de P, soit une action de Q, soit une action de communication entre P et Q (ceci est décrit par les axiomes de $\mathbin{\lfloor\!\lfloor}$ et $|$).
- Les axiomes de l'encapsulation et de la continuation se passent de commentaires ; ils correspondent au comportement intuitif de ces opérateurs.

- [rec1] décrit le développement d'une récursion. [rec2] est une axiome d'unicité (modulo $\equiv$) de la solution d'une équation récursive gardée. Enfin les axiomes [rec3] et [rec4] permettent d'éliminer les instances non gardées de la variable de récursion, en spécifiant qu'elles n'ont aucune influence sur le comportement.

Le théorème suivant affirme la consistance des axiomes. Sa preuve est donnée en annexe A

Théorème 2.3 (consistance de l'axiomatisation de $\mathcal{P}_t^r$)
Soit P et Q deux termes de $\mathcal{P}_t^r$. Si $P \equiv Q$, alors $P \sim Q$.
□

La commutativité et l'associativité du choix non déterministe nous permettent à nouveau d'employer la notation $\sum_{i\in I} P_i$, où I est un ensemble fini d'indices. Nous pouvons en particulier écrire $\sum_{i\in I} a_i\, P_i$; dans ce cas, le terme est identifié au processus **0** lorsque I est vide.

2.4.1 Systèmes d'équations

L'opérateur de récursion est utile pour les preuves des théorèmes, mais est peu agréable d'emploi pour décrire des processus. Il est souhaitable de pouvoir définir un système au moyen d'un système d'équations mutuellement récursives, par exemple :

$$\begin{array}{lcl} P, \text{ où} & & \\ P & \stackrel{\text{def}}{=} & \lfloor a\, Q \rfloor P \;+\; \lfloor b\, P \rfloor Q \\ Q & \stackrel{\text{def}}{=} & c\, P \;+\; d\, Q \end{array}$$

Un tel système d'équations signifie : "le processus P, où il existe Q tel que P et Q satisfont ces équations modulo $\equiv$".

La proposition suivante garantit que de tels systèmes ont une solution unique modulo $\equiv$.

Proposition 2.4
Soit $\overline{X} = X_1, \ldots, X_m$ et $\overline{Y} = Y_1, \ldots, Y_n$ des éléments distincts de $\mathcal{V}$. Soit $\overline{Q} = Q_1, \ldots, Q_m$ des termes de $\mathcal{P}_t^r$, à variables libres dans $\overline{X}$ et $\overline{Y}$, dans lesquels les X_i sont gardés. Alors il existe $\overline{P} = P_1, \ldots, P_m$ éléments de $\mathcal{P}_t^r$, à variables libres dans $\overline{Y}$, tels que

$$P_i \equiv Q_i[\overline{P}/\overline{X}] \qquad \forall i \in [1, m]$$

De plus, les P_i sont uniques modulo $\equiv$.
□

Remarque 2.2
Le terme $Q_i[\overline{P}/\overline{X}]$ représente la substitution simultanée des P_i aux X_i dans Q. Nous ne l'avons pas définie. Nous ne l'employons cependant que dans deux cas particuliers :

- lorsqu'aucun X_i n'est libre dans aucun P_j, on a trivialement

$$Q[P_1/X_1, \ldots, P_m/X_m] = Q[P_1/X_1] \ldots [P_m/X_m]$$

- l'autre situation consiste à réécrire le terme $Q[R/X][\overline{P}/\overline{X}]$ en $Q\left[R[\overline{P}/\overline{X}]/X\ ,\ \overline{P}/\overline{X}\right]$, lorsque X n'est pas dans $\overline{X}$. Il est facile de constater que les deux termes sont syntaxiquement identiques

□

Preuve. Pour prouver la proposition, nous procédons par récurrence sur la valeur de m.

- $m = 1$. Soit $P_1 \stackrel{\text{def}}{=} \texttt{rec}X_1 \cdot Q_1$. Les variables libres de P_1 sont dans $\overline{Y}$; de plus, par l'axiome [rec1], $P_1 \equiv Q_1[P_1/X]$. L'unicité modulo $\equiv$ est immédiate en utilisant [rec2], puisque X_1 est gardé dans Q_1.
- Supposons la propriété vraie pour m. Soit $\overline{Q} = Q_1, \ldots, Q_m$ dans $\mathcal{P}_t^r$, et $\overline{X} = X_1, \ldots, X_m$ dans $\mathcal{V}$. Soit de plus $Q_{m+1} \in \mathcal{P}_t^r$, et $X_{m+1} \in \mathcal{V}$. On suppose que les variables libres des Q_i ($i \in [1, m+1]$) sont dans $\overline{X}, \overline{Y}, X_{m+1}$.
  Nous définissons

$$\begin{array}{rcll} R_{m+1} & \stackrel{\text{def}}{=} & \texttt{rec}X_{m+1} \cdot Q_{m+1} & \\ R_i & \stackrel{\text{def}}{=} & Q_i[R_{m+1}/X_{m+1}] & i \in [1, m] \end{array}$$

  Pour $i \leq m$, les variables libres de R_i sont dans $\overline{X}$,$\overline{Y}$. Par hypothèse de récurrence, il existe $\overline{P} = P_1, \ldots, P_m$, dont les variables libres sont dans $\overline{Y}$, tels que $P_i \equiv R_i[\overline{P}/\overline{X}]$ pour $i \in [1, m]$.
  À présent, soit $P_{m+1} \stackrel{\text{def}}{=} R_{m+1}[\overline{P}/\overline{X}]$. Nous obtenons alors, pour chaque $i \leq m$,

$$\begin{array}{rcl} P_i & \equiv & Q_i[R_{m+1}/X_{m+1}][\overline{P}/\overline{X}] \\ & \equiv & Q_i[\overline{P}/\overline{X}\ ,\ (R_{m+1}[\overline{P}/\overline{X}])/X_{m+1}] \\ & \equiv & Q_i[\overline{P}/\overline{X}\ ,\ P_{m+1}/X_{m+1}] \end{array}$$

  De plus, comme X_{m+1} n'est pas libre dans $\overline{P}$, on a aussi

$$\begin{array}{rcll} P_{m+1} & = & \texttt{rec}X_{m+1} \cdot (Q_{m+1}[\overline{P}/\overline{X}]) & \\ & \equiv & Q_{m+1}[\overline{P}/\overline{X}\ ,\ P_{m+1}/X_{m+1}] & \text{(axiome [rec1])} \end{array}$$

  L'existence des P_i est donc montrée. Pour prouver l'unicité modulo $\equiv$, supposons que le même résultat peut être obtenu pour $\overline{P'} = P'_1, \ldots, P'_m$ et P'_{m+1}, dont les variables libres sont dans $\overline{Y}$. D'une part, nous avons

$$\begin{array}{rcll} P'_{m+1} & \equiv & Q_{m+1}[\overline{P'}/\overline{X}\ ,\ P'_{m+1}/X_{m+1}] & \\ & \equiv & Q_{m+1}[\overline{P'}/\overline{X}][P'_{m+1}/X_{m+1}] & \\ & \equiv & \texttt{rec}X_{m+1} \cdot (Q_{m+1}[\overline{P'}/\overline{X}]) & \text{(axiome [rec2])} \end{array}$$

  Puisque X_{m+1} n'est pas libre dans $\overline{P'}$, nous obtenons

$$P'_{m+1} \equiv R_{m+1}[\overline{P'}/\overline{X}] \qquad (1)$$

  D'autre part, cette dernière équation est utilisée en conjonction avec la propriété de $\overline{P'}$ pour obtenir, pour $i \leq m$:

$$\begin{array}{rcl} P'_i & \equiv & Q_i[\overline{P'}/\overline{X}\ ,\ (R_{m+1}[\overline{P'}/\overline{X}])/X_{m+1}] \\ & \equiv & Q_i[R_{m+1}/X_{m+1}][\overline{P'}/\overline{X}] \\ & \equiv & R_i[\overline{P'}/\overline{X}] \qquad \text{(définition de } R_i) \end{array}$$

  Comme nous avons aussi $P_i \equiv R_i[\overline{P}/\overline{X}]$, nous pouvons utiliser l'hypothèse de récurrence pour conclure que pour tout $i \leq m$, $P'_i \equiv P_i$. En injectant ces équivalences dans l'équation (1), nous obtenons

$$\begin{array}{rcl} P'_{m+1} & \equiv & R_{m+1}[\overline{P}/\overline{X}] \\ & \equiv & P_{m+1} \end{array}$$

  ce qui termine la preuve de la proposition. ∎

Cette preuve nous indique par ailleurs que le processus défini par le système d'équations donné en exemple ci-dessus est équivalent au terme

$$\mathtt{rec}X \cdot (\lfloor a\, \mathtt{rec}Y \cdot (c\, X \ + \ d\, Y)\rfloor X \ + \ \lfloor b\, X\rfloor \mathtt{rec}Y \cdot (c\, X \ + \ d\, Y))$$

qui correspond bien à ce que nous espérions.

2.4.2 Complétude de l'axiomatisation

Nous en arrivons à présent à la preuve de complétude du système d'axiomes Du fait de sa longueur, elle est présentée en annexe B.

Cette démonstration est basée sur la *forme canonique* des termes de $\mathcal{P}_t^r$ et d'ATP. Celle-ci est spécifiée en termes d'un système d'équations par la proposition 2.5 ci-dessous. Pour des raisons de lisibilité, nous nous restreignons ici aux processus d'ATP. Dans l'annexe B, nous présentons et prouvons la proposition correspondante pour $\mathcal{P}_t^r$.

Proposition 2.5 (forme canonique des processus d'ATP)
Soit P un terme d'ATP. Il existe $n \geq 1$ et des termes $P_1, \ldots, P_n$ d'ATP, tels que $P \equiv P_1$ et chaque P_i est solution d'une équation dans une et une seule des deux formes [1] et [2] définies comme suit :

$$[1] \quad P_i \ \equiv \ \sum_{j \in J_i} a_j\, P_{j_i} \qquad\qquad j_i \in [1,n]$$

$$[2] \quad P_i \ \equiv \ \left\lfloor \sum_{j \in J_i} a_j\, P_{j_i} \right\rfloor (P_{k_i}) \qquad j_i, k_i \in [1,n]$$

□

En utilisant cette proposition, il devient trivial de montrer que le modèle d'un processus d'ATP est fini, à branchement fini et déterministe par rapport aux transitions χ. La finitude découle du nombre fini des P_i ; le branchement fini se déduit de la finitude des J_i ; enfin, le déterminisme du passage du temps provient du fait que si un P_i est dans la forme [1], il n'a pas de transition χ, et s'il est dans la forme [2], il en a une seule.

Inversement, on peut prouver que tout système de transitions de cette forme est modèle d'un terme d'ATP, en associant un P_i à chacun de ses états.

Le même genre d'argument peut être utilisé pour caractériser les modèles des termes de $\mathcal{P}_t^r$, en utilisant la forme canonique décrite dans l'annexe B.

Nous présentons finalement le théorème de complétude du système d'axiomes de $\mathcal{P}_t^r$, qui constitue avec celui de consistance le résultat principal de ce chapitre.

Théorème 2.6 (complétude de l'axiomatisation de $\mathcal{P}_t^r$)
Soit P et Q deux termes de $\mathcal{P}_t^r$. Si $P \sim Q$, alors $P \equiv Q$.

□

Sa démonstration est présentée en annexe B. Grâce à ce théorème, il est possible de prouver l'équivalence forte de deux processus au moyen des seuls axiomes, sans passer par les systèmes de transitions.

2.5 Processus bien temporisés

Cette dernière section est consacrée au problème de la détermination de la "réalisabilité" d'un processus décrit en ATP, au sens où son comportement peut être celui d'un système réel exécuté sur une machine.

2.5.1 Processus fortement bien temporisés

Il est incontestable qu'un tel processus doit garantir que l'écoulement du temps ne peut être bloqué. Si on considère que l'exécution du processus commence à l'instant 0, cela signifie que chacun de ses comportements possibles permet d'atteindre tout instant $d > 0$.

Définition 2.3 (processus fortement bien temporisé)
Nous appelons *fortement bien temporisé* (en abrégé *fbt*) un processus d'ATP qui satisfait cette contrainte, c'est-à-dire dont toutes les exécutions (séquences de transitions du modèle) maximales comportent une infinité de transitions χ.
□

Une séquence de transitions est maximale si elle est de longueur infinie, ou si elle ne peut être prolongée par aucune transition. Il est clair que si une séquence de la seconde sorte existe, alors le processus n'est pas *fbt*. Un tel cas correspond à une possibilité de blocage.

Exemple 2.1
Les processus $P_1 = a\,\mathbf{0}$ et $P_2 = \mathtt{rec}X \cdot a\,\mathbf{0} + b\,X$ ne sont pas *fbt* ; en revanche, le processus $P_3 = \mathtt{rec}X \cdot \lfloor a\,\lfloor \mathbf{0} \rfloor X \rfloor X$ est *fbt* (voir figure 2.3). P_1 se bloque inévitablement ; P_2 peut que se bloquer ou exécuter une infinité d'actions sans laisser passer le temps ; par contre, le comportement de P_3 consiste à exécuter au plus une action a par unité de temps.

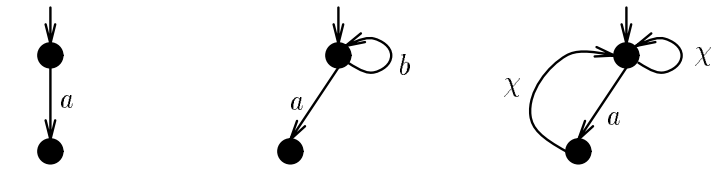


Figure 2.3 : processus non *fbt* (P_1 et P_2) et *fbt* (P_3)

□

Il est difficile, voire impossible, de déterminer syntaxiquement si un processus est *fbt*. Le problème principal provient de la combinaison de la composition parallèle et de l'encapsulation. Le premier opérateur peut faire "apparaître" des actions (de communication), alors que le second peut "couper" des chemins dans le modèle. La conséquence peut être le blocage d'un processus *fbt*, ou inversement la suppression de boucles d'actions qui rendaient un processus non *fbt*.

À l'inverse, il est facile de déterminer si un processus est *fbt*, une fois construite sa forme canonique représentée sous la forme d'un système d'équations (prop. 2.5) : pour tout processus P il existe des termes $P_1, \ldots, P_n$ tels que $P \equiv P_1$, et chaque P_i est solution d'un équation dans une des deux

formes

$$P_i \equiv \sum_{j \in J_i} a_j \, P_{j_i} \qquad\qquad P_i \equiv \left\lfloor \sum_{j \in J_i} a_j \, P_{j_i} \right\rfloor P_{k_i}$$

Les règles suivantes définissent un prédicat FBT qui s'applique à un processus d'ATP dont on connaît la forme canonique.

On constate aisément que $\texttt{FBT}(P)$ est vrai (tt) si et seulement si P est *fbt*. Le principe de la définition est de "parcourir" les P_i en partant de P_1. FBT est défini en fonctions de prédicats indicés $\texttt{FBT}_{V,B}$, où V et B sont des sous-ensembles de $[1,n]$. Lors de l'évaluation de $\texttt{FBT}_{V,B}(P_i)$, V contient les indices des P_j déjà visités, et $B \subseteq V$ les indices des P_j depuis la visite desquels on a rencontré une transition χ.

$\texttt{FBT}(P) = \texttt{FBT}_{\emptyset,\emptyset}(P_1)$

(1) **si** $i \in V$ **et** $i \notin B$ **alors** $\texttt{FBT}_{V,B}(P_i) = \texttt{ff}$

(2) **si** $i \in V$ **et** $i \in B$ **alors** $\texttt{FBT}_{V,B}(P_i) = \texttt{tt}$

(3) **si** $i \notin V$ **alors**

si $P_i \equiv 0$ **alors** $\texttt{FBT}_{V,B}(P_i) = \texttt{ff}$

si $P_i \equiv \sum_{j \in J_i \neq \emptyset} a_j \, P_{j_i}$ **alors** $\texttt{FBT}_{V,B}(P_i) = \bigwedge_{j \in J_i} \texttt{FBT}_{V \cup \{i\},B}(P_{j_i})$

si $P_i \equiv \left\lfloor \sum_{j \in J_i} a_j \, P_{j_i} \right\rfloor P_{k_i}$ **alors** $\texttt{FBT}_{V,B}(P_i) = \bigwedge_{j \in J_i} \texttt{FBT}_{V \cup \{i\},B}(P_{j_i}) \wedge \texttt{FBT}_{V \cup \{i\},V \cup \{i\}}(P_{k_i})$

L'algorithme se termine toujours, car

- les P_i sont en nombre fini ;
- si i est dans V, le calcul de $\texttt{FBT}_{V,B}(P_i)$ s'arrête (cas (1) ou (2)) ;
- si i n'est pas dans V, $\texttt{FBT}_{V,B}(P_i)$ est défini en fonction de plusieurs $\texttt{FBT}_{V',B'}(P_j)$ où V' est strictement plus grand que V. On retombe donc toujours sur les cas (1) ou (2).

$\texttt{FBT}_{V,B}(P_i)$ est trivialement faux (cas (1)) lorsque dans le chemin parcouru pour arriver à P_i, on a déjà visité P_i ($i \in V$), et on n'est pas passé par une transition χ depuis ($i \notin B$). On a donc trouvé un cycle de transitions d'actions de P_i à P_i.

Au contraire, lorsque $i \in V$ et $i \in B$, on est passé par une transition χ depuis la dernière visite de P_i ; pour le chemin choisi, $\texttt{FBT}_{V,B}(P_i)$ est donc trivialement vrai (cas (2)) puisqu'on a un cycle où le temps passe.

Le chemin choisi est mémorisé par V et B. L'ordre exact des termes visités précédemment n'a pas à être connu pour déterminer par le cas (1) ou (2) le statut de $\texttt{FBT}_{V,B}(P_i)$; seule importe la connaissance des termes visités avant (B) et après ($V - B$) la dernière transition χ.

En fait, les P_i représentent les états du modèle de P. L'algorithme décrit consiste en un parcours "en avant" des chemins du graphe, jusqu'à l'obtention d'un puits ($\mathbf{0}$) ou d'un cycle. Si le cycle comporte une transition χ, alors le chemin est temporellement "correct" ; dans le cas contraire, il représente un comportement où le temps est bloqué. Il suffit de trouver un seul chemin du second type pour que tout le système ne soit pas *fbt*.

On peut effectuer le même calcul "en arrière", en vérifiant par model checking sur le graphe une propriété de logique temporelle qui s'exprime en CTL [CES86] par

$$\forall\Box(\neg\text{puits} \wedge \forall\Diamond\text{after}(\chi))$$

C'est-à-dire : pour tout état P_i du modèle, P_i n'est pas un puits et tout chemin issu de P_i contient une transition χ.

Nous ne décrivons pas ici la méthode du model checking, qui n'est pas l'objet de ce chapitre.

2.5.2 Processus bien temporisés

La classe de processus décrite ci-dessus est trop restrictive. En effet, considérons un système extrêmement simple dont le comportement cyclique consiste à émettre immédiatement une sortie (action b) chaque fois qu'il reçoit une entrée (action a). Aucune hypothèse n'est faite sur la fréquence des entrées. Un tel système est représenté en ATP par le processus $P \stackrel{\text{def}}{=} \texttt{rec}X \cdot \tilde{a}\, b\, X$. Son modèle est présenté figure 2.4. On constate qu'il n'est pas fortement bien temporisé. En

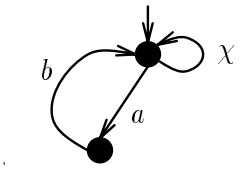


Figure 2.4 : un processus non *fbt*

effet, parmi ses comportements possibles, il existe des séquences infinies qui bloquent le temps : toutes celles qui après un nombre fini de transitions, se prolongent par une alternance infinie de a et de b.

Il paraît cependant surprenant de rejeter un tel système. En effet, si la fréquence des entrées est inconnue, donc non bornée, elle est de toutes façons finie. Il est donc inévitable qu'une unité de temps s'écoule après plusieurs entrées. Or les séquences d'exécution qui font de ce processus un système non *fbt* sont telles qu'au bout d'un certain nombre de transitions, une transition χ est infiniment souvent possible sans jamais être exécutée : elles ne sont pas *équitables* du point de vue des transitions χ. Lorsqu'on implante le système, ces séquences d'exécution ne peuvent jamais survenir. Les autres séquences (les "bonnes" séquences) satisfont bien à l'exigence de progression du temps. De plus, toute séquence de transitions finie à partir de l'état initial du système peut toujours être prolongée par une "bonne" séquence. En d'autres termes, le système ne peut jamais se retrouver dans une situation où le temps devient indéfiniment bloqué.

Définition 2.4 (processus bien temporisé)

Nous appelons *bien temporisés* (en abrégé *bt*) les processus d'ATP qui satisfont ces contraintes, c'est-à-dire dont toutes les séquences d'exécution finies peuvent être prolongées en des séquences d'exécution infinies comportant une infinité de transitions χ

□

En termes de modèles, cela signifie que de tout état il est possible d'atteindre un état d'où est issue une transition χ. En logique temporelle CTL, tous les états doivent vérifier la propriété $\exists\Diamond\text{enable}(\chi)$ (propriété `P1`).

Comme pour les processus *fbt*, nous définissons un prédicat $\mathtt{BT}(P)$ qui est vrai si P est bien temporisé. Nous utilisons à nouveau les termes P_i définissant la forme canonique de P. $\mathtt{BT}(P)$ est calculé à partir des prédicats $\mathtt{BT}_{V,B}(P_i)$. V contient toujours les indices des états déjà visités dans le chemin courant ; B est à présent l'ensemble des indices des états du chemin courant qui satisfont `P1`.

Le calcul de $\mathtt{BT}(P)$ et des $\mathtt{BT}_{V,B}(P_i)$ est plus compliqué, mais obéit aux mêmes principes que celui de $\mathtt{FBT}(P)$ et $\mathtt{FBT}_{V,B}(P_i)$: on parcourt en avant le graphe en mémorisant les états déjà visités.

L'algorithme est défini par les règles suivantes. Il est facile de vérifier qu'il s'arrête toujours.

$$\mathtt{BT}(P) = \mathtt{BT}_{\emptyset,\emptyset}(P_1)$$

(1) **si** $i \in V$ **et** $i \notin B$ **alors** $\quad \mathtt{BT}_{V,B}(P_i) = \mathtt{ff}$

(2) **si** $i \in V$ **et** $i \in B$ **alors** $\quad \mathtt{BT}_{V,B}(P_i) = \mathtt{tt}$

(3) **si** $i \notin V$ **alors**

(3.1) **si** $P_i \equiv \sum_{j \in J_i t} a_j P_{j_i}$ **alors**

$$\mathtt{BT}_{V,B}(P_i) = \textbf{si } \bigvee_{j \in J_i} \mathtt{BT}_{V\cup\{i\},B}(P_{j_i}) \textbf{ alors } \bigwedge_{j \in J_i} \mathtt{BT}_{V\cup\{i\},V\cup\{i\}}(P_{j_i}) \textbf{ sinon } \mathtt{ff}$$

(3.2) **si** $P_i \equiv \left\lfloor \sum_{j \in J_i} a_j P_{j_i} \right\rfloor P_{k_i}$ **alors**

$$\mathtt{BT}_{V,B}(P_i) = \bigwedge_{j \in J_i} \mathtt{BT}_{V\cup\{i\},V\cup\{i\}}(P_{j_i}) \ \wedge \ \mathtt{BT}_{V\cup\{i\},V\cup\{i\}}(P_{k_i})$$

On vérifie facilement que lors du calcul de $\mathtt{BT}_{V,B}(P_i)$, B contient les indices des états du chemin déjà parcouru qui vérifient `P1`. Dès qu'un état satisfait `P1`, tous ses prédécesseurs dans le chemin la satisfont aussi. Dans le cas (3.2), P_i satisfait trivialement `P1`. Il suffit alors de vérifier que c'est aussi le cas de ses successeurs. Dans le cas (3.2), il faut au préalable vérifier qu'*un* des successeurs de P_i satisfait `P1`. On en déduit alors que P_i et tous ses prédécesseurs dans le chemin la satisfont, et il reste à vérifier en utilisant cette information que c'est également le cas de *tous* ses successeurs.

Remarque 2.3 (variabilité finie et bornée)

Les processus bien temporisés sont également dits *à variabilité finie* et les processus fortement bien temporisés *à variabilité bornée*. En effet, dans le premier cas, les "bonnes" séquences d'exécution comportent un nombre fini mais non nécessairement borné d'actions entre deux transitions χ ; dans le second cas, ce nombre est borné par une constante calculable pour l'ensemble du système.

□

Dans le chapitre 6, nous présentons une méthode de model checking symbolique de propriétés de logique temporelle temps réel sur des systèmes dans le cas d'un domaine temporel dense. Elle ne donne un résultat correct que pour de systèmes bien temporisés, et son principe restreint automatiquement la vérification aux seules "bonnes" séquences. Nous expliquons également comment déterminer par la même méthode si un processus est bien temporisé (pour ce calcul particulier, le résultat doit bien entendu être correct même si le système n'est pas *bt*).

Chapitre 3

Opérateurs temporels et exemples

Comme nous l'avons vu au chapitre précédent, ATP est basée sur deux opérateurs contraignant le comportement temporel d'un processus : le préfixage immédiat et le délai unitaire.

Ces opérateurs permettent d'obtenir une classe de modèles suffisamment large. Celle-ci contient tous les comportements spécifiables à partir d'une notion de temps discret déterministe, pourvu qu'il soient d'état fini et à branchement fini. Le préfixage immédiat décrit une situation d'urgence (états sans transition χ), et le délai unitaire une situation où le passage d'une unité de temps (transition χ) peut modifier radicalement le comportement du processus.

En théorie, on peut donc se contenter des opérateurs d'ATP pour définir des processus temporisés. En pratique, il est nécessaire de disposer de moyens de spécification et de description "de haut niveau", reflétant la *structure* d'un système. Par exemple, il est classique dans les systèmes temps-réel de contrôler l'exécution d'un processus P au moyen d'un *chien de garde* : si P n'a pas exécuté une action donnée avant un certain temps, il est interrompu et un traitement d'exception est déclenché. Décrire un tel système au moyen du seul délai unitaire peut être très compliqué, car il est nécessaire pour cela de connaître la structure du modèle de P.

Ce chapitre est consacré à l'introduction d'opérateurs temporels de "haut niveau" dans le cadre d'ATP. Ces opérateurs entrent dans une catégorie générale de définition d'*observateurs*. Comme son nom l'indique, un observateur est un processus qui observe le comportement d'un autre processus relativement à une *contrainte* que celui-ci doit satisfaire. Nous ne considérons qu'un cas particulier de contraintes, qui stipulent que le processus doit exécuter certaines actions dans un temps donné.

Nous distinguons deux catégories d'opérateurs :

- ceux qui définissent des observateurs *passifs* : le processus s'exécute normalement jusqu'à ce qu'il viole la contrainte ; il est alors interrompu définitivement et une exception est déclenchée ;
- ceux qui définissent des observateurs *actifs* : ceux-ci restreignent le comportement du processus de telle sorte qu'il ne puisse violer la contrainte ; on peut dans ce cas obtenir une situation de blocage, si le processus se trouve dans l'impossibilité de la satisfaire. En ce sens, l'opérateur d'encapsulation peut être vu comme définissant un observateur actif, la contrainte étant que le processus ne doit exécuter aucune action de l'ensemble paramétrant l'encapsulation.

Nous définissons ci-dessous divers opérateurs pour chacune de ces catégories. Nous décrivons leur sémantique observationnelle et nous en proposons une axiomatisation. Leur utilisation est illustrée par plusieurs exemples.

Dans tout ce chapitre, nous nous situons dans le cadre de la définition d'ATP : les processus sont fermés, réguliers et bien gardés. La sémantique ne concerne donc que les règles d'action et les règles temporelles. Nous indiquons lorsqu'un opérateur garde certains de ses arguments.

3.1 Observateurs passifs

Les observateurs passifs sont définis au moyen d'un unique opérateur : le *chien de garde* ou *watchdog*. Les cas particuliers du chien de garde permettent de dériver quatre autres opérateurs : le *délai de commencement*, le *retard*, le *délai d'exécution* et le *délai de terminaison*.

3.1.1 Chien de garde

Le *chien de garde* est un opérateur à deux arguments ; il est paramétré par un entier strictement positif d et un ensemble d'actions H.

Le chien de garde de *corps* P, d'*exception* Q, de *durée* d et d'*actions attendues* H est noté $\lceil P \rceil_H^d Q$. Ce processus se comporte initialement comme P. Si celui-ci exécute une des actions de H avant d unités de temps, alors l'exécution de P continue normalement. Dans le cas contraire, *le délai expire à l'instant* d : P est interrompu définitivement et l'exception Q est exécutée. En d'autres termes, un chien de garde surveille si un processus exécute certaines actions dans un délai donné, et déclenche une exception si ce n'est pas le cas. Cet opérateur garde son second argument (l'exception).

La sémantique opérationnelle du chien de garde est décrite par quatre règles :

actions	temps
$\dfrac{P \xrightarrow{a} P'}{\lceil P \rceil_H^d Q \xrightarrow{a} \lceil P' \rceil_H^d Q}\ a \notin H$	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P \rceil_H^{d+1} Q \xrightarrow{\chi} \lceil P' \rceil_H^d Q}$
$\dfrac{P \xrightarrow{a} P'}{\lceil P \rceil_H^d Q \xrightarrow{a} P'}\ a \in H$	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P \rceil_H^1 Q \xrightarrow{\chi} Q}$

On peut axiomatiser complètement le chien de garde au moyen de la liste (non nécessairement minimale) d'axiomes suivante :

$$
\begin{array}{ll}
[\lceil\rceil 1] & \lceil P + Q \rceil_H^d R \equiv \lceil P \rceil_H^d R + \lceil Q \rceil_H^d R \\
[\lceil\rceil 2] & \lceil \mathbf{0} \rceil_H^d Q \equiv \mathbf{0} \\
[\lceil\rceil 3] & \lceil a\,P \rceil_H^d Q \equiv a \lceil P \rceil_H^d Q \quad \text{si } a \notin H \\
[\lceil\rceil 4] & \lceil a\,P \rceil_H^d Q \equiv a\,P \quad \text{si } a \in H \\
[\lceil\rceil 5] & \lceil \lfloor P \rfloor Q \rceil_H^1 R \equiv \lfloor \lceil P \rceil_H^1 R \rfloor R \\
[\lceil\rceil 6] & \lceil \lfloor P \rfloor Q \rceil_H^{d+1} R \equiv \lfloor \lceil P \rceil_H^{d+1} R \rfloor \lceil Q \rceil_H^d R \\
[\lceil\rceil 7] & \lceil \lceil P \rceil_H^d Q \rceil_H^{d'} R \equiv \lceil P \rceil_H^{d'} R \qquad \text{si } d \geq d' \\
[\lceil\rceil 8] & \lceil \lceil P \rceil_H^d Q \rceil_H^{d'} R \equiv \lceil P \rceil_H^d \lceil Q \rceil_H^{d'-d} R \qquad \text{si } d < d'
\end{array}
$$

À l'aide de l'axiome $[\lceil\rceil 7]$, on peut montrer que tout terme fermé, régulier et bien gardé comportant des opérateurs de chien de garde est équivalent à un processus d'ATP. Cela est vrai

même lorsqu'un terme comporte une récursion "à travers" un chien de garde. L'exemple suivant illustre un tel cas.

Exemple 3.1

Soit le processus P défini par

$$P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot (a \lceil \mathtt{rec}Y \cdot \lfloor c\,X + d\,X \rfloor Y \rceil^2_H (\mathtt{rec}Z \cdot \lfloor e\,X \rfloor Z) \;+\; b\,X)$$

où $H = \{c\}$. L'axiome [rec1] de la récursion nous permet d'écrire

$$\begin{array}{rcl} P & \equiv & a\,\lceil Q_1 \rceil^2_H Q_2 \;+\; b\,P \\ Q_1 & \equiv & \lfloor c\,P + d\,P \rfloor Q_1 \\ Q_2 & \equiv & \lfloor e\,P \rfloor Q_2 \end{array}$$

La séquence d'équivalences suivante se déduit des axiomes du chien de garde.

$$\begin{array}{rcll} \lceil Q_1 \rceil^2_H Q_2 & \equiv & \lceil \lfloor c\,P + d\,P \rfloor Q_1 \rceil^2_H Q_2 & \\ & \equiv & \lfloor \lceil c\,P + d\,P \rceil^2_H Q_2 \rfloor \lceil Q_1 \rceil^1_H Q_2 & \\ & \equiv & \lfloor c\,P + d\,\lceil P \rceil^2_H Q_2 \rfloor \lceil Q_1 \rceil^1_H Q_2 & \\ \lceil Q_1 \rceil^1_H Q_2 & \equiv & \lceil \lfloor c\,P + d\,P \rfloor Q_1 \rceil^1_H Q_2 & \\ & \equiv & \lfloor \lceil c\,P + d\,P \rceil^1_H Q_2 \rfloor Q_2 & \\ & \equiv & \lfloor c\,P + d\,\lceil P \rceil^1_H Q_2 \rfloor Q_2 & \\ \lceil P \rceil^2_H Q_2 & \equiv & \lceil a\,\lceil Q_1 \rceil^2_H Q_2 \;+\; b\,P \rceil^2_H Q_2 & \\ & \equiv & a\,\lceil \lceil Q_1 \rceil^2_H Q_2 \rceil^2_H Q_2 \;+\; b\,\lceil P \rceil^2_H Q_2 & \\ & \equiv & a\,\lceil Q_1 \rceil^2_H Q_2 \;+\; b\,\lceil P \rceil^2_H Q_2 & \text{(axiome [}\lceil\,\rceil 7\text{])} \\ \lceil P \rceil^1_H Q_2 & \equiv & \lceil a\,\lceil Q_1 \rceil^2_H Q_2 \;+\; b\,P \rceil^1_H Q_2 & \\ & \equiv & a\,\lceil \lceil Q_1 \rceil^2_H Q_2 \rceil^1_H Q_2 \;+\; b\,\lceil P \rceil^1_H Q_2 & \\ & \equiv & a\,\lceil Q_1 \rceil^1_H Q_2 \;+\; b\,\lceil P \rceil^1_H Q_2 & \text{(axiome [}\lceil\,\rceil 7\text{])} \end{array}$$

Considérons les six termes d'ATP définis par

$$\begin{array}{rcl} R_1 & \stackrel{\text{def}}{=} & a\,R_2 \;+\; b\,R_1 \\ R_2 & \stackrel{\text{def}}{=} & \lfloor c\,R_1 + d\,R_4 \rfloor R_3 \\ R_3 & \stackrel{\text{def}}{=} & \lfloor c\,R_1 + d\,R_5 \rfloor R_6 \\ R_4 & \stackrel{\text{def}}{=} & a\,R_2 \;+\; b\,R_4 \\ R_5 & \stackrel{\text{def}}{=} & a\,R_3 \;+\; b\,R_5 \\ R_6 & \stackrel{\text{def}}{=} & \lfloor e\,R_1 \rfloor R_6 \end{array}$$

La proposition 2.4 (page 48) assure que ces six processus existent dans ATP. Or P, $\lceil Q_1 \rceil^2_H Q_2$, $\lceil Q_1 \rceil^1_H Q_2$, $\lceil P \rceil^2_H Q_2$, $\lceil P \rceil^1_H Q_2$ et Q_2 satisfont ces six équations. Ils sont donc respectivement équivalents aux R_i $(i = 1..6)$.

□

On peut de plus montrer que tout terme comportant une récursion à travers le corps d'un chien de garde est équivalent à un processus où ces récursions sont gardées par des actions attendues (des éléments du paramètre H du chien de garde). Cette propriété est illustrée en reprenant le terme P de l'exemple 3.1, qu'on prolonge comme suit, afin d'éliminer l'appel récursif "$d\,X$".

Exemple 3.2

On définit le terme

$$P' \stackrel{\text{def}}{=} \texttt{rec}X \cdot (a \lceil \texttt{rec}Y \cdot (\lfloor c\, X + d\, \texttt{rec}U \cdot (a\, Y \ + \ b\, U)\rfloor Y)\rceil^2_H \texttt{rec}Z \cdot (\lfloor e\, X\rfloor Z) \ + \ b\, X)$$

L'axiome [rec1] nous permet à nouveau d'écrire

$$\begin{aligned} P' &\equiv a \lceil Q'_1\rceil^2_H Q'_2 \ + \ b\, P' \\ Q'_1 &\equiv \lfloor c\, P' + d\, P''\rfloor Q'_1 \\ Q'_2 &\equiv \lfloor e\, P'\rfloor Q'_2 \\ P'' &\equiv a\, Q'_1 \ + \ b\, P'' \end{aligned}$$

P' diffère de P par le fait que dans le corps du chien de garde (ici Q'_1), l'appel récursif de P non gardé par c est remplacé par un terme P'' ; celui-ci a la même structure que P', sauf que le chien de garde ($\lceil Q'_1\rceil^2_H Q'_2$) est remplacé par son corps. On remarque que P' est un terme sans récursion à travers le premier argument du chien de garde.

Les axiomes du chien de garde entraînent alors la série d'équivalences suivante :

$$\begin{aligned} \lceil Q'_1\rceil^2_H Q'_2 &\equiv \lceil \lfloor c\, P' + d\, P''\rfloor Q'_1\rceil^2_H Q'_2 \\ &\equiv \left\lfloor \lceil c\, P' + d\, P''\rceil^2_H Q'_2 \right\rfloor \lceil Q'_1\rceil^1_H Q'_2 \\ &\equiv \left\lfloor c \lceil P''\rceil^2_H Q'_2 \right\rfloor \lceil Q'_1\rceil^1_H Q'_2 \\ \lceil Q'_1\rceil^1_H Q'_2 &\equiv \lceil \lfloor c\, P' + d\, P''\rfloor Q'_1\rceil^1_H Q'_2 \\ &\equiv \left\lfloor \lceil c\, P' + d\, P''\rceil^1_H Q'_2 \right\rfloor Q'_2 \\ &\equiv \left\lfloor c\, P' + d \lceil P''\rceil^1_H Q'_2 \right\rfloor Q'_2 \\ \lceil P''\rceil^2_H Q'_2 &\equiv \lceil a\, Q'_1 \ + \ b\, P''\rceil^2_H Q'_2 \\ &\equiv a \lceil Q'_1\rceil^2_H Q'_2 \ + \ b \lceil P''\rceil^2_H Q'_2 \\ \lceil P''\rceil^1_H Q'_2 &\equiv \lceil a\, Q'_1 \ + \ b\, P''\rceil^1_H Q'_2 \\ &\equiv a \lceil Q'_1\rceil^1_H Q'_2 \ + \ b \lceil P''\rceil^1_H Q'_2 \end{aligned}$$

On constate que P', $\lceil Q'_1\rceil^2_H Q'_2$, $\lceil Q'_1\rceil^1_H Q'_2$, $\lceil P''\rceil^2_H Q'_2$, $\lceil P''\rceil^1_H Q'_2$ et Q'_2 satisfont eux aussi les six équations des R_i présentées dans l'exemple 3.1. L'énoncé d'unicité de la proposition 2.4 entraîne alors $P \equiv P'$. Cela peut se comprendre si on remarque que lorsque Q_1 fait un appel récursif à P sans avoir exécuté d'action de H, on obtient deux chiens de garde imbriqués, le plus externe expirant toujours avant le plus profond. On peut donc remplacer ce dernier par son corps.

□

Formellement, cette équivalence se traduit de la manière suivante : si aucune occurrence de Y n'est gardée par un élément de H dans Q, alors

$$\texttt{rec}X \cdot (P\,[\lceil Q[X/Y]\rceil^d_H R/Z]) \equiv \texttt{rec}X \cdot (P\,[\lceil Q[\texttt{rec}V \cdot (P[V/X][Q[V/Y]/Z])/Y]\rceil^d_H R/Z])$$

3.1.2 Délai de commencement

Le *délai de commencement* ou *timeout* est un cas particulier de chien de garde, dans lequel l'ensemble H est constitué de $\mathcal{A}$ tout entier. $\lceil P\rceil^d_{\mathcal{A}} Q$ est noté $P \stackrel{d}{\triangleright} Q$ (la notation est empruntée à timed CSP [DS89]). L'opérateur est associatif à droite : $P \stackrel{d}{\triangleright} Q \stackrel{d'}{\triangleright} R = P \stackrel{d}{\triangleright} (Q \stackrel{d'}{\triangleright} R)$.

Un délai de commencement de corps P, de durée d et d'exception Q définit donc un observateur qui contrôle que P exécute sa première action avant l'instant d.

Les règles de sémantique du chien de garde permettent d'obtenir facilement celles du délai de commencement :

actions	temps	
$\dfrac{P \xrightarrow{a} P'}{P \overset{d}{\triangleright} Q \xrightarrow{a} P'}$	$\dfrac{P \xrightarrow{\chi} P'}{P \overset{d+1}{\triangleright} Q \xrightarrow{\chi} P' \overset{d}{\triangleright} Q}$	$\dfrac{P \xrightarrow{\chi} P'}{P \overset{1}{\triangleright} Q \xrightarrow{\chi} Q}$

De même, la restriction du chien de garde au cas $H = \mathcal{A}$ nous donne les axiomes suivants :

$$
\begin{array}{ll}
[\triangleright 1] & (P + Q) \overset{d}{\triangleright} R \equiv P \overset{d}{\triangleright} R + Q \overset{d}{\triangleright} R \\
[\triangleright 2] & \mathbf{0} \overset{d}{\triangleright} Q \equiv \mathbf{0} \\
[\triangleright 3] & a\,P \overset{d}{\triangleright} Q \equiv a\,P \\
[\triangleright 4] & \lfloor P \rfloor Q \overset{1}{\triangleright} R \equiv \lfloor P \rfloor R \\
[\triangleright 5] & \lfloor P \rfloor Q \overset{d+1}{\triangleright} R \equiv \lfloor P \rfloor (Q \overset{d}{\triangleright} R) \\
[\triangleright 6] & (P \overset{d}{\triangleright} Q) \overset{d'}{\triangleright} R \equiv P \overset{d'}{\triangleright} R \qquad \text{si } d \geq d' \\
[\triangleright 7] & (P \overset{d}{\triangleright} Q) \overset{d'}{\triangleright} R \equiv P \overset{d}{\triangleright} Q \overset{d'-d}{\triangleright} R \qquad \text{si } d < d'
\end{array}
$$

Les trois premiers axiomes peuvent être remplacés par

$$(\mathbf{0} + P) \overset{d}{\triangleright} Q \equiv \mathbf{0} + P$$

3.1.3 Retard

Lorsque le corps d'un délai de commencement de durée d ne fait que laisser passer le temps, l'effet obtenu est de différer de d unités de temps l'exécution de l'exception. Nous définissons l'opérateur de *retard de d unités de temps* d'un processus P par :

$$(d)P \overset{\text{def}}{=} \delta \overset{d}{\triangleright} P \quad \text{ou} \quad (d)P \overset{\text{def}}{=} \lceil \delta \rceil_H^d P$$

On peut également le définir au moyen du délai unitaire :

$$(1)P \overset{\text{def}}{=} \lfloor \mathbf{0} \rfloor P \qquad (d+1)P \overset{\text{def}}{=} \lfloor \mathbf{0} \rfloor (d)P = \underbrace{\lfloor \mathbf{0} \rfloor \lfloor \mathbf{0} \rfloor \ldots \lfloor \mathbf{0} \rfloor}_{d+1 \text{ fois}} P$$

Cet opérateur est considéré comme primitif dans de nombreuses algèbres [BL91, DS89, HR91, MT90, Wan91]. L'algèbre est alors définie, soit sans préfixage immédiat dans [BL91, DS89, HR91, Wan91], soit avec un choix non déterministe faible (cf. section 2.2.2, page 40), dans [MT90]. Nous préférons utiliser le délai unitaire comme opérateur de base d'ATP, car il se conjugue parfaitement avec le préfixage immédiat et le choix fort pour construire une théorie simple d'une algèbre dont le pouvoir d'expression est suffisamment élevé.

La sémantique du retard n'est composée que de règles temporelles sans prémisse :

temps	
$\dfrac{}{(d+1)P \xrightarrow{\chi} (d)P}$	$\dfrac{}{(1)P \xrightarrow{\chi} P}$

3.1.4 Délai d'exécution

Le troisième cas particulier du chien de garde consiste à choisir $H = \emptyset$. Nous obtenons alors le *délai d'exécution*, noté $\lceil P \rceil^d Q$. Aucune action n'étant attendue, son effet se limite à interrompre P et à exécuter Q au bout de d unités de temps.

Le délai d'exécution rappelle la construction `upto` d'ESTEREL [BCG87]: $\lceil P \rceil^d Q$ correspond à

$$\texttt{do}\ P\ \texttt{upto}\ d\chi\ ;\ Q$$

La sémantique opérationnelle du délai d'exécution se déduit de celle du chien de garde :

actions	temps	
$\dfrac{P \xrightarrow{a} P'}{\lceil P \rceil^d Q \xrightarrow{a} \lceil P' \rceil^d Q}$	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P \rceil^{d+1} Q \xrightarrow{\chi} \lceil P \rceil^d Q}$	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P \rceil^1 Q \xrightarrow{\chi} Q}$

Les axiomes du délai d'exécution sont identiques à ceux du chien de garde où on supprime l'ensemble H. L'axiome [⌈ ⌉4] disparaît évidemment.

3.1.5 Délai de terminaison

Le dernier cas particulier de chien de garde est appelé *délai de terminaison* et est noté $\langle P \rangle^d Q$. Cet opérateur définit un observateur qui lance l'exception Q si le corps P *n'a pas signalé sa terminaison* avant l'instant d. Dans le cas contraire, $\langle P \rangle^d Q$ est terminé et le signale au même instant que P.

Nous rappelons (cf. section 2.2.3.1, page 41) qu'un processus peut signaler sa terminaison au moyen du terme $\texttt{fin} \stackrel{\text{def}}{=} \xi^1\ \delta$. La terminaison est donc ici prise au sens de l'exécution de l'échappement ξ^1.

Le délai de terminaison est donc défini par $\langle P \rangle^d Q \stackrel{\text{def}}{=} \lceil P \rceil^d_{\{\xi^1\}} Q$.

Sa sémantique opérationnelle est donnée par les règles suivantes :

actions	temps
$\dfrac{P \xrightarrow{a} P'}{\langle P \rangle^d Q \xrightarrow{a} \langle P' \rangle^d Q}\ a \neq \xi^1$	$\dfrac{P \xrightarrow{\chi} P'}{\langle P \rangle^{d+1} Q \xrightarrow{\chi} \langle P' \rangle^d Q}$
$\dfrac{P \xrightarrow{\xi^1} P'}{\langle P \rangle^d Q \xrightarrow{a} P'}$	$\dfrac{P \xrightarrow{\chi} P'}{\langle P \rangle^1 Q \xrightarrow{\chi} Q}$

Les axiomes du délai de terminaison sont les mêmes que ceux du chien de garde, en prenant $H = \{\xi^1\}$.

3.2 Observateurs actifs

Nous décrivons deux opérateurs définissant des observateurs actifs. Le premier impose un caractère d'urgence à un processus lorsqu'il peut exécuter certaines actions. Le second exige qu'un processus exécute certaines actions dans un délai donné.

3.2.1 Opérateur d'urgence

Soit H un sous-ensemble de $\mathcal{A}$, et P un processus. L'opérateur d'*urgence* paramétré par H et appliqué à P est noté $\mathtt{U}_H(P)$. Il se lit "H est urgent dans P". Le processus $\mathtt{U}_H(P)$ se comporte comme P, sauf lorsque celui-ci peut exécuter une action de H ; dans ce cas, il doit exécuter *immédiatement* une des actions possibles de P (pas nécessairement dans H). En d'autres termes, $\mathtt{U}_H(P)$ ne peut pas attendre lorsqu'il peut exécuter une des actions de H. Cet opérateur a été proposé dans U-LOTOS [BL91], où il se nomme "as soon as possible".

La sémantique de $\mathtt{U}_H(P)$ est décrite par les deux règles suivantes :

actions	temps
$\dfrac{P \xrightarrow{a} P'}{\mathtt{U}_H(P) \xrightarrow{a} \mathtt{U}_H(P')}$	$\dfrac{P \xrightarrow{\chi} P', \forall a \in H,\ P \overset{a}{\not\longrightarrow}}{\mathtt{U}_H(P) \xrightarrow{\chi} \mathtt{U}_H(P')}$

Il est facilement axiomatisable :

$$
\begin{array}{ll}
[\mathtt{U1}] & \mathtt{U}_H(P+Q) \equiv \mathtt{U}_H(P) + \mathtt{U}_H(Q) \\
[\mathtt{U2}] & \mathtt{U}_H(\mathbf{0}) \equiv \mathbf{0} \\
[\mathtt{U3}] & \mathtt{U}_H(a\,P) \equiv a\,\mathtt{U}_H(P) \\
[\mathtt{U4}] & \mathtt{U}_H(\lfloor \mathbf{0} \rfloor P) \equiv \lfloor \mathbf{0} \rfloor \mathtt{U}_H(P) \\
[\mathtt{U5}] & \mathtt{U}_H(\lfloor a\,P \rfloor Q) \equiv a\,\mathtt{U}_H(P) \qquad \text{si } a \in H \\
[\mathtt{U6}] & \mathtt{U}_H(\lfloor a\,P \rfloor Q) \equiv \lfloor a\,\mathtt{U}_H(P) \rfloor \mathtt{U}_H(Q) \qquad \text{sia} \notin \mathrm{H}
\end{array}
$$

On constate facilement que $\mathtt{U}_{\{a\}\cup H}(P) \equiv \mathtt{U}_{\{a\}}(\mathtt{U}_H(P))$. Donc, si on n'autorise que des paramètres H finis, on peut se contenter d'un opérateur d'urgence paramétré par une seule action.

L'opérateur d'urgence est utile lorsqu'il s'applique à une composition parallèle, comme le montre l'exemple suivant.

Exemple 3.3

Soit deux processus P et Q en parallèle. Initialement, P peut exécuter l'action a, et Q l'action b, n'importe quand. On veut de plus que *dès que P et Q ont exécuté a et b*, les deux processus se synchronisent (par une action de communication $r = r_P | r_Q$) avant de continuer leur exécution. Si on définit

$$P \overset{\text{def}}{=} \widetilde{a}\,\widetilde{r_P}\,P' \qquad \text{et} \qquad Q \overset{\text{def}}{=} \widetilde{b}\,\widetilde{r_Q}\,Q'$$

on obtient pour $\partial_{\{r_P, r_Q\}}(P \| Q)$ le comportement présenté sur la figure 3.1 (on suppose que ni P' ni Q' ne peuvent exécuter r_P, r_Q ou r). On constate que le rendez-vous n'a pas nécessairement lieu dès que possible.

Il faut utiliser l'opérateur d'urgence, en déclarant que r est urgent dans le processus global. On définit donc le terme $\mathtt{U}_{\{r\}}(\partial_{\{r_P, r_Q\}}(P \| Q))$, pour lequel la synchronisation se produit dès que P et Q ont respectivement exécuté a et b (voir figure 3.2).

□

3.2.2 Restriction temporelle

Le dernier opérateur de ce chapitre restreint le comportement d'un processus de telle sorte qu'il ne puisse progresser au-delà d'un instant donné tant qu'il n'a pas exécuté certaines actions. Cet

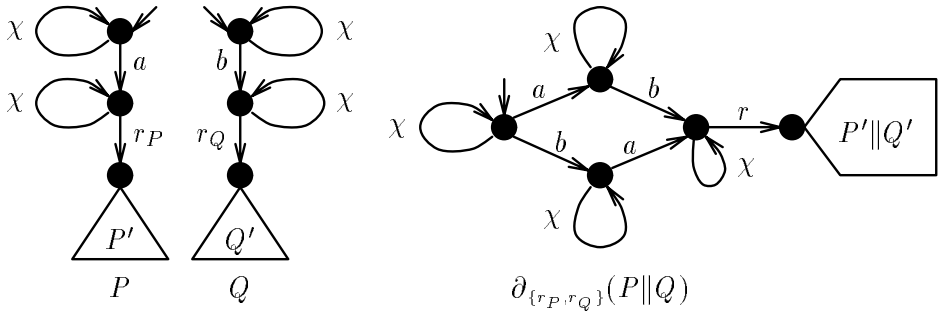


Figure 3.1 : un rendez-vous non immédiat

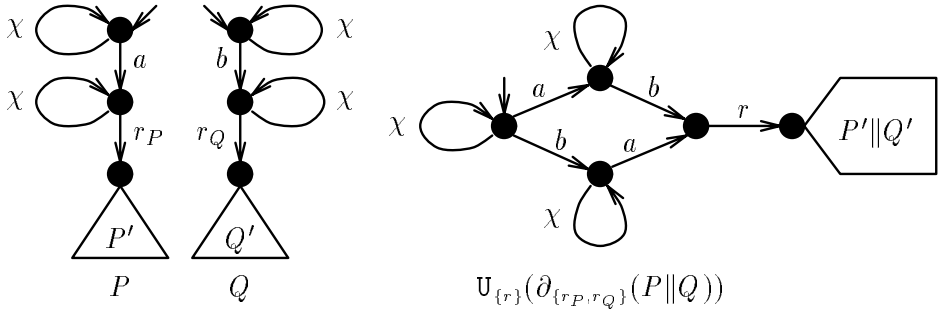


Figure 3.2 : un rendez-vous dès que possible

opérateur, appelé *restriction temporelle*, est paramétré par un délai d et un ensemble d'actions H ; appliqué à un terme P, il est noté $P\Downarrow_H^d$. Ce processus se comporte comme P, mais il ne peut atteindre l'instant d que s'il a exécuté une des actions de H.

La sémantique de la restriction temporelle est décrite par les règles suivantes :

actions	temps
$\dfrac{P \xrightarrow{a} P'}{P\Downarrow_H^d \xrightarrow{a} P'\Downarrow_H^d}\, a \notin H$ $\quad$ $\dfrac{P \xrightarrow{a} P'}{P\Downarrow_H^d \xrightarrow{a} P'}\, a \in H$	$\dfrac{P \xrightarrow{\chi} P'}{P\Downarrow_H^{d+1} \xrightarrow{\chi} P\Downarrow_H^d}$

Cet opérateur s'axiomatise très naturellement :

$$
\begin{array}{ll}
[\Downarrow 1] & (P+Q)\Downarrow_H^d \;\equiv\; P\Downarrow_H^d \;+\; Q\Downarrow_H^d \\
[\Downarrow 2] & \mathbf{0}\Downarrow_H^d \;\equiv\; \mathbf{0} \\
[\Downarrow 3] & a\,P\Downarrow_H^d \;\equiv\; a\,P \qquad \text{si } a \in H \\
[\Downarrow 4] & a\,P\Downarrow_H^d \;\equiv\; a\,(P\Downarrow_H^d) \qquad \text{si } a \notin H \\
[\Downarrow 5] & (\lfloor P \rfloor Q)\Downarrow_H^1 \;\equiv\; P\Downarrow_H^1 \\
[\Downarrow 6] & (\lfloor P \rfloor Q)\Downarrow_H^{d+1} \;\equiv\; \lfloor P\Downarrow_H^{d+1} \rfloor (Q\Downarrow_H^d)
\end{array}
$$

3.3 Exemples

Nous présentons quatre exemples d'utilisation des opérateurs temporels : un protocole de communication, une version améliorée de l'émetteur de ce protocole, une procédure de connexion à

un terminal et un contrôleur de passage à niveau.

3.3.1 Protocole du bit alterné

Le premier exemple est une version simplifiée du protocole du bit alterné, où les lignes de communication unidirectionnelles se comportent comme des tampons de taille 1 avec possibilité de perte.

3.3.1.1 Description du protocole

Le système est composé d'un émetteur E, d'un récepteur R, et de deux lignes L et LL, la première transmettant de E vers R, la seconde de R vers E (figure 3.3).

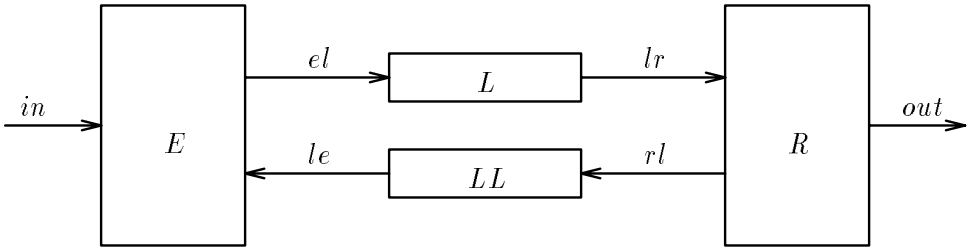


Figure 3.3 : composantes du protocole du bit alterné

Lorsqu'il en reçoit la demande (in), l'émetteur transmet un message à la ligne L (action el). S'il reçoit ensuite un accusé de réception dans un délai de, il peut passer au message suivant ; sinon il réémet le message courant.

Le récepteur attend un message sur la ligne L (action lr). Si ce message n'est pas une duplication d'un message déjà reçu, il le transmet immédiatement à l'extérieur (out). Dans tous les cas, il envoie ensuite un accusé de réception (action rl) via la ligne LL.

La ligne L (resp. LL) peut perdre le message (resp. l'accusé de réception) qu'elle transporte (actions lp et llp), ou le transmettre correctement (actions lr et le) après un délai de dl (resp. dll) unités de temps.

Afin que l'émetteur ne réémette pas inutilement un message dont l'accusé de réception n'a de toutes façons pas eu le temps d'arriver, on suppose $de > dl + dll$.

Pour traiter correctement la perte des messages, un *bit de contrôle* 0 ou 1 est ajouté aux messages et accusés de réception. Il reste le même si l'information est dupliquée, et il change lorsqu'elle correspond à l'envoi d'un nouveau message ou de son accusé de réception.

3.3.1.2 Modélisation en ATP

L'émetteur. Le délai avant réémission est modélisé par un délai de commencement : il est annulé dès qu'un accusé de réception arrive, que celui-ci soit celui attendu ou non. Dans le second cas, le message est réémis et le délai est de nouveau armé.

L'émetteur est décrit par le système d'équations suivant.

$$\begin{aligned}
E_0 &\stackrel{\text{def}}{=} \widetilde{in}\, E_1 \\
E_1 &\stackrel{\text{def}}{=} \widetilde{el_0}\, \left((\widetilde{le_0}\, E_2 \;+\; \widetilde{le_1}\, E_1) \stackrel{de}{\triangleright} E_1 \right) \\
E_2 &\stackrel{\text{def}}{=} \widetilde{in}\, E_3 \\
E_3 &\stackrel{\text{def}}{=} \widetilde{el_1}\, \left((\widetilde{le_0}\, E_3 \;+\; \widetilde{le_1}\, E_0) \stackrel{de}{\triangleright} E_3 \right)
\end{aligned}$$

Le modèle de l'émetteur dans le cas où $de = 3$ est présenté sur la figure 3.4.

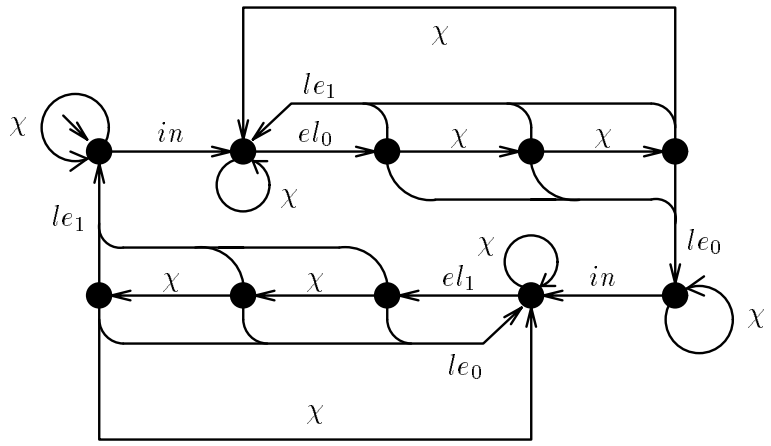


Figure 3.4 : modèle de l'émetteur

Les lignes. Leurs descriptions sont similaires, elles comportent un opérateur de retard. Celle de L est donnée par les équations suivantes :

$$\begin{aligned}
L_0 &\stackrel{\text{def}}{=} \widetilde{el_0}\, L_1 \;+\; \widetilde{el_1}\, L_2 \\
L_1 &\stackrel{\text{def}}{=} \widetilde{lp}\, L_0 \;+\; (dl)\widetilde{lr_0}\, L_0 \\
&\equiv \widetilde{lp}\, L_0 \stackrel{dl}{\triangleright} (\widetilde{lp}\, L_0 + \widetilde{lr_0}\, L_0) \\
L_2 &\stackrel{\text{def}}{=} \widetilde{lp}\, L_0 \;+\; (dl)\widetilde{lr_1}\, L_0 \\
&\equiv \widetilde{lp}\, L_0 \stackrel{dl}{\triangleright} (\widetilde{lp}\, L_0 + \widetilde{lr_1}\, L_0)
\end{aligned}$$

Les équations pour LL sont :

$$\begin{aligned}
LL_0 &\stackrel{\text{def}}{=} \widetilde{rl_0}\, LL_1 \;+\; \widetilde{rl_1}\, LL_2 \\
LL_1 &\stackrel{\text{def}}{=} \widetilde{llp}\, LL_0 \;+\; (dll)\widetilde{le_0}\, LL_0 \\
&\equiv \widetilde{\tau}\, LL_0 \stackrel{dll}{\triangleright} (\widetilde{llp}\, LL_0 + \widetilde{le_0}\, LL_0) \\
LL_2 &\stackrel{\text{def}}{=} \widetilde{llp}\, LL_0 \;+\; (dll)\widetilde{le_1}\, LL_0 \\
&\equiv \widetilde{llp}\, LL_0 \stackrel{dll}{\triangleright} (\widetilde{llp}\, LL_0 + \widetilde{le_1}\, LL_0)
\end{aligned}$$

Le modèle de L pour $dl = 1$ est présenté sur la figure 3.5.

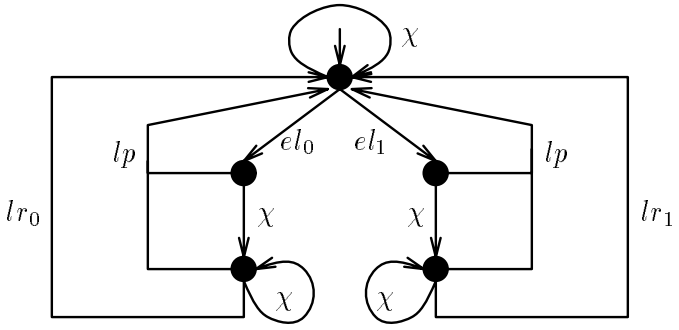


Figure 3.5 : modèle de la ligne L

Le récepteur. Sa description est très simple :

$$\begin{aligned}
R_0 &\stackrel{\text{def}}{=} \widetilde{lr_0}\, R_1 \;+\; \widetilde{lr_1}\, R_3 \\
R_1 &\stackrel{\text{def}}{=} out\, R_2 \\
R_2 &\stackrel{\text{def}}{=} \widetilde{rl_0}\, R_3 \\
R_3 &\stackrel{\text{def}}{=} \widetilde{lr_0}\, R_2 \;+\; \widetilde{lr_1}\, R_4 \\
R_4 &\stackrel{\text{def}}{=} out\, R_5 \\
R_5 &\stackrel{\text{def}}{=} \widetilde{rl_1}\, R_6 \\
R_6 &\stackrel{\text{def}}{=} \widetilde{lr_0}\, R_1 \;+\; \widetilde{lr_1}\, R_5
\end{aligned}$$

Son modèle est dessiné sur la figure 3.6.

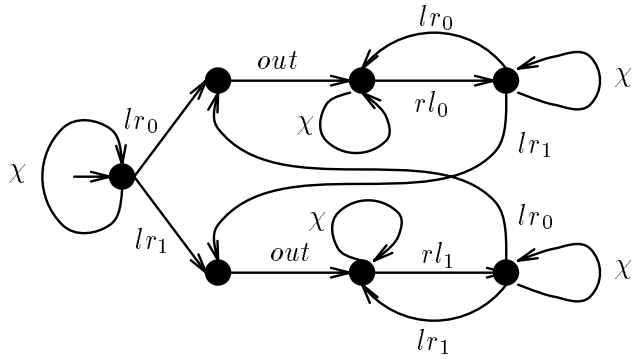


Figure 3.6 : modèle du récepteur

Le protocole complet. Il est constitué de la mise en parallèle des quatre composantes décrites ci-dessus. La fonction de communication est définie par :

$$el_0|el_0 = el_1|el_1 = el \quad lr_0|lr_0 = lr_1|lr_1 = lr \quad rl_0|rl_0 = rl_1|rl_1 = rl \quad le_0|le_0 = le_1|le_1 = le$$

toutes les autres communications étant impossibles.

On force les communications en encapsulant la mise en parallèle par l'ensemble

$$H \stackrel{\text{def}}{=} \{el_0, el_1, lr_0, lr_1, rl_0, rl_1, le_0, le_1\}$$

Finalement, on impose que ces communications aient lieu dès que possible en utilisant l'opérateur d'urgence, paramétré par

$$H' \stackrel{\text{def}}{=} \{el, lr, rl, le\}$$

Le protocole complet est alors défini par

$$AB \stackrel{\text{def}}{=} \mathtt{U}_{H'}(\partial_H(E_0 \| L_0 \| LL_0 \| R_0))$$

Son modèle est présenté sur la figure 3.7.

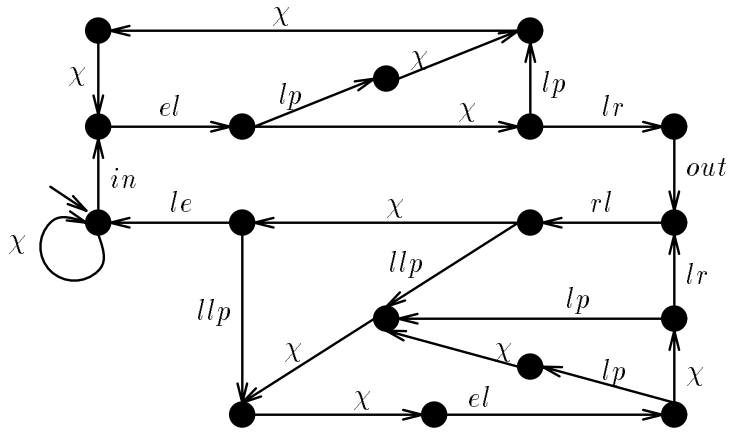


Figure 3.7 : modèle du protocole de bit alterné

3.3.2 Extension pour l'émetteur

Nous ajoutons une contrainte à l'émetteur du protocole du bit alterné, qui stipule que la phase de transmission d'un message (entre la demande d'émission in et la réception de l'accusé de réception correspondant) ne doit pas excéder une durée da. dans le cas contraire, on considère que la connexion est coupée ; l'émetteur entre alors dans une phase de déconnexion D (non spécifiée). On suppose da plusieurs fois supérieur à de, pour autoriser plusieurs essais de transmission.

On utilise alors un chien de garde par bit de contrôle. Ils sont activés lors d'une demande d'émission correspondant à leurs bits respectifs, et désactivés sur réception de le_0, l'autre sur réception de le_1. S'ils expirent, le contrôle est transmis à D.

L'émetteur ainsi spécifié est décrit en ATP par les équations suivantes :

$$\begin{aligned}
E_0 &\stackrel{\text{def}}{=} \widetilde{in}\, \lceil E_1 \rceil^{da}_{\{le_0\}} D \\
E_1 &\stackrel{\text{def}}{=} \widetilde{el_0}\, \left((\widetilde{le_0}\, E_2 \;+\; \widetilde{le_1}\, E_1) \stackrel{de}{\triangleright} E_1 \right) \\
E_2 &\stackrel{\text{def}}{=} \widetilde{in}\, \lceil E_3 \rceil^{da}_{\{le_1\}} D \\
E_3 &\stackrel{\text{def}}{=} \widetilde{el_1}\, \left((\widetilde{le_0}\, E_3 \;+\; \widetilde{le_1}\, E_0) \stackrel{de}{\triangleright} E_3 \right)
\end{aligned}$$

Son modèle est quelque peu compliqué ; nous le présentons au chapitre 5, sous la forme d'un graphe temporisé.

3.3.3 Procédure de connexion à un terminal

3.3.3.1 Description informelle

Nous considérons à présent une procédure P de connexion à un terminal d'ordinateur spécifiée de la façon suivante.

Initialement, le système affiche à l'écran un message (action m), et attend une réponse de l'utilisateur. Celle-ci peut être valide (v) ou invalide (i). Si la réponse est valide, le système entre dans la phase de session S. Si elle est invalide, ou si aucune réponse n'est donnée dans un délai dr, le système recommence en affichant un nouveau message.

On souhaite de plus contrôler la durée totale de la procédure. Celle-ci ne doit pas excéder dp unités de temps. Si ce délai expire, la procédure est annulée et une phase d'exception E est lancée.

3.3.3.2 Modélisation en ATP

Grâce au chien de garde et au délai de commencement, cette spécification est décrite très facilement. Le processus P est défini par deux équations :

$$\begin{aligned} P &\stackrel{\text{def}}{=} \lceil L \rceil_{\{v\}}^{dp} E \\ L &\stackrel{\text{def}}{=} m\ ((\widetilde{v}\, S\ +\ \widetilde{\imath}\, L) \overset{dr}{\triangleright} L) \end{aligned}$$

La figure 3.8 présente le modèle de L dans le cas où $dr = 3$.

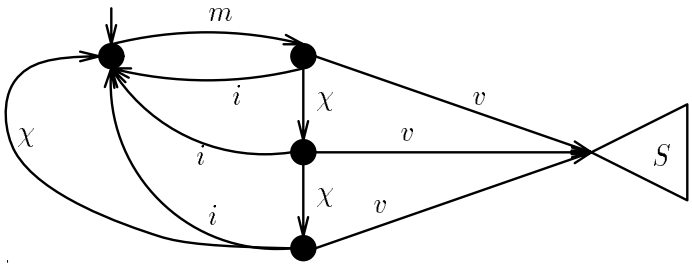


Figure 3.8 : procédure de connexion, modèle de L

Pour obtenir le modèle de la procédure complète P, il faut "dérouler" celui de L sur la durée dp et "coller" le modèle de E après le dernier χ. L'expansion du modèle s'arrête au-delà des transitions v. On obtient le système de transitions de la figure 3.9 dans le cas où $dr = 2$ et $dp = 7$.

3.3.3.3 Variante

Une autre possibilité est de considérer que la procédure se termine après que l'utilisateur a donné une réponse valide. Le chien de garde devient alors un délai de terminaison et le contrôle est transmis à S dès que P est terminé.

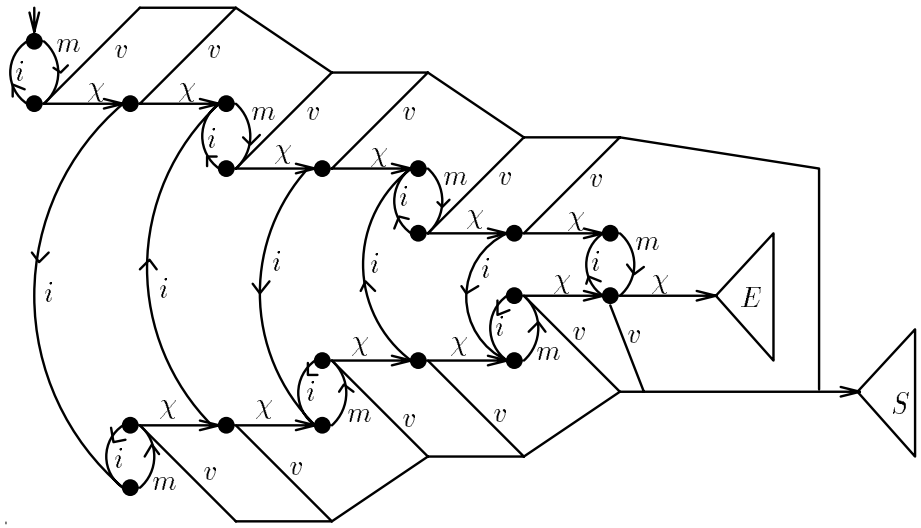


Figure 3.9 : procédure de connexion, modèle complet

Cette spécification se traduit en ATP par les équations suivantes.

$$\begin{array}{lrcl} \text{Pour la procédure :} & P & \stackrel{\text{def}}{=} & \langle L \rangle^{dp} E \\ & L & \stackrel{\text{def}}{=} & m\ ((\widetilde{v}\,\texttt{fin} + \widetilde{\imath}\,L) \stackrel{dr}{\triangleright} L) \\ \text{Pour le système complet :} & SYS & \stackrel{\text{def}}{=} & P \bigtriangledown S \end{array}$$

Le modèle de SYS est presque identique à celui de la figure 3.9 : une transition τ doit être ajoutée entre les transitions par v et le modèle de S.

3.3.4 Contrôleur de passage à niveau

Le dernier exemple de ce chapitre est tiré de [Alu91]. Il s'agit d'un contrôleur de la barrière d'un passage à niveau. Il communique avec un train et avec la barrière. Le comportement de ceux-ci n'est pas complètement décrit, on spécifie uniquement les intervalles minimaux ou maximaux séparant leurs actions.

Le train envoie au contrôleur deux signaux, indiquant respectivement l'approche (a_t) et la sortie (s_t) de la zone du passage à niveau. Après le signal d'approche, le train met au moins trois unités de temps avant d'entrer dans la zone (action i). Le signal s est envoyé après la sortie effective (action o). On suppose de plus que moins de six unités de temps s'écoulent entre a_t et s_t.

En ATP, le train est décrit par le processus T suivant :

$$\begin{array}{rcl} T & \stackrel{\text{def}}{=} & \widetilde{a_t}\,(T_1 \Downarrow^6_{\{s_t\}}) \\ T_1 & \stackrel{\text{def}}{=} & (3)\widetilde{\imath}\,\widetilde{o}\,\widetilde{s_t}\,T \end{array}$$

Son modèle est présenté sur la figure 3.10.

La barrière reçoit du contrôleur deux signaux : descendre (d_b) et monter (m_b). Elle met moins d'une unité de temps pour se baisser ; l'action b correspond à la fin de l'abaissement. Lorsqu'elle

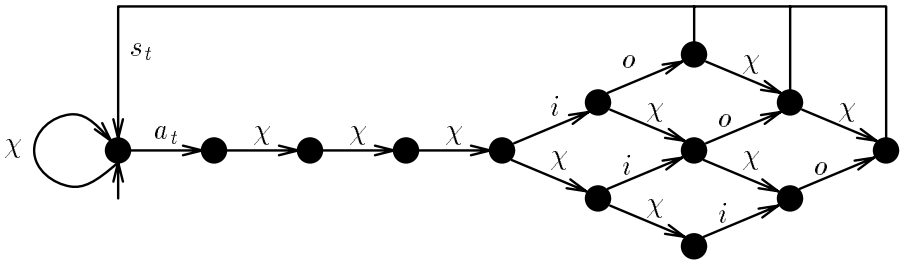


Figure 3.10 : modèle du train

reçoit le signal m_b, elle met au moins une et au plus deux unités de temps pour se relever, en terminant par l'action h. La description en ATP de la barrière est :

$$\begin{aligned} B &\stackrel{\text{def}}{=} \widetilde{d_b}\,(B_1 \Downarrow^1_{\{b\}}) \\ B_1 &\stackrel{\text{def}}{=} \widetilde{b}\,\widetilde{m_b}\,(B_2 \Downarrow^2_{\{h\}}) \\ B_2 &\stackrel{\text{def}}{=} (1)\widetilde{h}\,B \end{aligned}$$

La figure 3.11 présente son modèle.

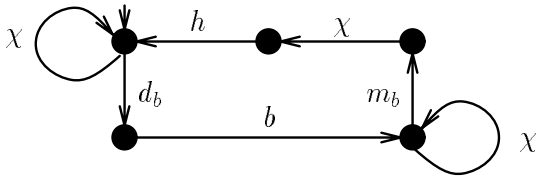


Figure 3.11 : modèle de la barrière

Le contrôleur se comporte de la manière suivante : lorsque le train approche (action a_c), il attend deux unités de temps avant d'ordonner à la barrière de descendre (action d_c) ; lorsque le train signale la sortie de la zone (action s_c), il envoie à la barrière le signal de montée (action m_c) en moins d'une unité de temps. Ce comportement est décrit en ATP par le processus

$$C \stackrel{\text{def}}{=} \widetilde{a_c}\,(2)d_c\,\widetilde{s_c}\,((\widetilde{m_c}\,C)\Downarrow^1_{\{m_c\}})$$

dont le modèle est présenté sur la figure 3.12.

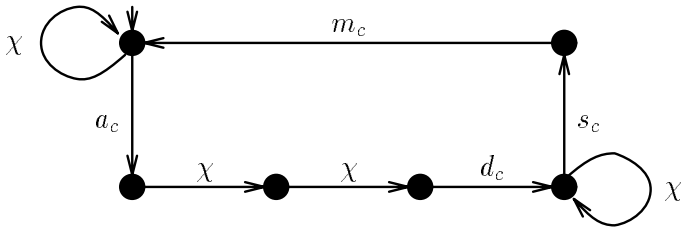


Figure 3.12 : modèle du contrôleur de passage à niveau

Le système complet S est constitué de la mise en parallèle des trois composantes T, B et C, où les seules communication possibles entre actions sont :

$$a_t|a_c = a \qquad s_t|s_c = s \qquad d_c|d_b = d \qquad m_c|m_b = m$$

On force ces communications à a avoir lieu en encapsulant la mise en parallèle avec l'ensemble

$$H \stackrel{\text{def}}{=} \{a_t, a_c, s_t, s_c, d_c, d_b, m_c, m_b\}$$

On a donc $S \stackrel{\text{def}}{=} \partial_H(T \| C \| B)$ Nous ne donnons pas le modèle de S. Il est décrit sous la forme d'un graphe temporisé au chapitre 5. Nous nous servons également de cet exemple pour illustrer le model checking symbolique dans le chapitre 6.

Chapitre 4

Les algèbres paramétrées par un domaine temporel

Les opérateurs définis dans le chapitre précédent permettent de décrire des contraintes temporelles imposées aux processus. Ces contraintes sont paramétrées par un délai, représentant un nombre d'*unités de temps* (nous rappelons que l'unité de temps est notés $\mathtt{U_t}$). La sémantique des opérateurs a été présentée dans le cadre de l'algèbre ATP du chapitre 2, et donc sous l'hypothèse que la base de temps $\mathtt{B_t}$ (dont la durée correspond à une transition χ) est identique à $\mathtt{U_t}$. En d'autres termes, cette sémantique considère un délai d comme représentant une durée $d\mathtt{B_t}$.

Lorsqu'on décrit un système temps réel, on fixe une fois pour toutes l'unité de temps. Par contre, la base de temps — qui définit la précision des mesures temporelles — dépend en général de contraintes imposées par le matériel sur lequel le système est implémenté. On exige seulement que $\mathtt{U_t}$ soit un multiple de $\mathtt{B_t}$.

Si on souhaite obtenir le modèle d'un système pour une base de temps particulière, il est nécessaire de transformer la spécification en multipliant toutes les valeurs des délais par $(\mathtt{U_t}/\mathtt{B_t})$. Ainsi, ceux-ci sont exprimés en fonction de la base de temps, et on peut appliquer les règles de sémantique des opérateurs.

Cette transformation, si elle peut être effectuée automatiquement, présente l'inconvénient de rendre les descriptions dépendantes de la base de temps. En d'autres termes, porter un système temps réel d'une machine vers une autre impose de modifier le code source avant de le recompiler.

Pour remédier à ce problème, il est souhaitable de disposer d'un formalisme dans lequel l'expression des processus est indépendante de la base de temps, celle-ci devenant un paramètre de la sémantique. La première section de ce chapitre décrit les algèbres ATP_g, paramétrées par la *granularité* $g \stackrel{\text{def}}{=} \mathtt{B_t}/\mathtt{U_t}$. Leur syntaxe est indépendante de la granularité.

Afin d'obtenir une description indépendante de la base de temps, il faut bien sûr proscrire l'utilisation du délai unitaire, et n'employer que les opérateurs du chapitre 3.

Par hypothèse, g est toujours de la forme $\frac{1}{n}$, où n est un entier strictement positif. On définit la relation d'ordre partiel "plus grossière que" entre granularités, notée $\preceq$, par

$$g \preceq g' \iff \exists k \in \mathbb{N}, g = kg'$$

Si g est plus grossière que g', on dit aussi que g' est plus fine que g.

Un autre problème important se pose lorsqu'on modifie la base de temps d'un système : les propriétés vérifiées par P avec la base de temps $\mathtt{B_{t_1}}$ sont-elles vraies avec la base de temps $\mathtt{B_{t_2}}$? La réponse est : "en général non". Cependant, il existe une certaine classe de propriétés, les *propriétés de sûreté* [MP89, BFG+91], qui restent vraies lorsqu'on multiplie la base de temps (donc lorsqu'on passe à une granularité plus grossière). Par exemple, si un système satisfait une propriété de sûreté pour $g = 10^{-6}$, alors il la satisfait également pour $g = 10^{-4}$. En conséquence, si on vérifie une telle propriété relativement à la base de temps la plus petite possible, on est certain qu'elle est vraie pour toute granularité plus grossière.

Il n'existe cependant pas de plus fine granularité. Nous reconsidérons le problème en définissant le *domaine temporel* relatif à la granularité g par

$$D_g \stackrel{\text{def}}{=} \{kg, k \in \mathbb{N}\}$$

La relation $\preceq$ induit un ordre partiel entre les D_g : $D_g \preceq D_{g'} \iff g \preceq g'$. On a trivialement $D_g \preceq D_{g'} \iff D_g \subseteq D_{g'}$. Donc, si $D_g \subseteq D_{g'}$, toute propriété de sûreté satisfaite pour la granularité g' l'est également pour la granularité g. Dans le treillis des parties de $\mathbb{R}$, la borne supérieure de l'ensemble des D_g est $\mathbb{Q}_+$.

Dans la section 4.2, nous définissons la sémantique des algèbres ATP_D, paramétrées par un domaine temporel D quelconque. Leur syntaxe est identique à celle des ATP_g. Nous montrons alors que si un processus satisfait une propriété de sûreté dans $\mathrm{ATP}_{\mathbb{Q}_+}$, alors il la satisfait dans toutes les ATP_{D_g}. La définition d'ATP_D apparaît dans [NSY91].

La dernière partie de ce chapitre est consacrée à une étude comparative de quelques algèbres proposées par d'autres auteurs, qu'on peut également trouver dans [NS91]. Nous nous intéressons principalement à l'influence du choix des opérateurs et de leur sémantique sur la classe des modèles (et donc sur l'expressivité de l'algèbre).

4.1 Les algèbres ATP_g

4.1.1 Syntaxe

Pour simplifier l'étude des algèbres, nous ne faisons pas apparaître l'opérateur de continuation, dont le traitement ne pose pas de problème particulier.

Un processus d'ATP_g ne doit pas comporter de délai unitaire, et les opérateurs temporels doivent être paramétrés par des délais entiers (représentant des multiples de $\mathtt{U_t}$).

Nous incluons dans les algèbres les opérateurs de chien de garde, d'urgence et de restriction temporelle.

Comme pour AUP et ATP, la définition d'ATP_g passe par celle d'une algèbre générale $\mathcal{P}_g$. Cette dernière comporte l'opérateur de délai unitaire, et les valeurs des délais peuvent être quelconques parmi $D_g^* \stackrel{\text{def}}{=} D_g - \{0\}$. Les éléments de $\mathcal{P}_g$ sont donc définis par la syntaxe suivante :

$$\begin{array}{rcl} P ::= & X & \mid\ a\,P \ \mid\ \widetilde{a}\,P \ \mid\ P + P \ \mid\ \lfloor P \rfloor P \ \mid\ \lceil P \rceil_H^d P \ \mid \\ & \mathtt{U}_H(P) & \mid\ P \Downarrow_H^d \ \mid\ P \| P \ \mid\ \partial_H(P) \ \mid\ \mathtt{rec}X \cdot P \end{array}$$

où $X \in \mathcal{V}$, $a \in \mathcal{A}$, $d \in D_g^*$ et $H \subseteq \mathcal{A}$.

La définition des opérateurs dérivées de blocage, de terminaison, de retard, de délai de commencement, d'exécution et de terminaison n'est pas modifiée :

$\mathbf{0}$	$\stackrel{\text{def}}{=}$	$\partial_{\{a\}}(a\,P)$	δ	$\stackrel{\text{def}}{=}$	$\partial_{\{a\}}(\widetilde{a}\,P)$	$(d)P$	$\stackrel{\text{def}}{=}$	$\lceil\delta\rceil^d_H P$
$P \stackrel{d}{\triangleright} Q$	$\stackrel{\text{def}}{=}$	$\lceil P\rceil^d_{\mathcal{A}} Q$	$\lceil P\rceil^d Q$	$\stackrel{\text{def}}{=}$	$\lceil P\rceil^d_\emptyset Q$	$\langle P\rangle^d Q$	$\stackrel{\text{def}}{=}$	$\lceil P\rceil^d_{\xi^1} Q$

Les définitions des variables libres et gardées et des termes fermés, réguliers et bien gardées sont inchangées.

L'algèbre ATP_g est alors la sous-algèbre des termes fermés, réguliers, bien gardés de $\mathcal{P}_g$, sans opérateur de délai unitaire, et où les valeurs des délais sont entières. L'ensemble des termes des ATP_g est donc identique pour toutes les granularités.

4.1.2 Sémantique opérationnelle

Afin de ne pas surcharger la présentation, nous ne définissons que la sémantique opérationnelle des termes fermés. Nous ne nous préoccupons donc pas des transitions par une variable. Celles-ci sont définies de la même manière que pour ATP.

Le domaine des relations de transition $\stackrel{}{\not\to}_g$ est $\mathcal{P}_g \times \mathcal{A} \cup \{\chi\} \times \mathcal{P}_g$. Une transition par χ correspond au passage du temps d'une durée $\mathtt{B_t} = g\mathtt{U_t}$.

Les règles d'action sont identiques à celles d'ATP pour tous les opérateurs. Les règles temporelles ne sont modifiées que pour le chien de garde, l'urgence et la restriction temporelle.

Les relations $\stackrel{}{\not\to}_g$ sont définies par les règles suivantes :

Règles d'action

$$\frac{}{a\,P \stackrel{a}{\not\to}_g P} \qquad \frac{}{\widetilde{a}\,P \stackrel{a}{\not\to}_g P} \qquad \frac{P \stackrel{a}{\not\to}_g P'}{P+Q \stackrel{a}{\not\to}_g P'} \qquad \frac{Q \stackrel{a}{\not\to}_g Q'}{P+Q \stackrel{a}{\not\to}_g Q'}$$

$$\frac{P \stackrel{a}{\not\to}_g P'}{\lfloor P\rfloor Q \stackrel{a}{\not\to}_g P'} \qquad \frac{P \stackrel{a}{\not\to}_g P'}{\lceil P\rceil^a_H Q \stackrel{a}{\not\to}_g \lceil P'\rceil^a_H Q}\, a \notin H \qquad \frac{P \stackrel{a}{\not\to}_g P'}{\lceil P\rceil^a_H Q \stackrel{a}{\not\to}_g P'}\, a \in H \qquad \frac{P \stackrel{a}{\not\to}_g P'}{\mathtt{U}_H(P) \stackrel{a}{\not\to}_g \mathtt{U}_H(P')}$$

$$\frac{P \stackrel{a}{\not\to}_g P'}{P\|Q \stackrel{a}{\not\to}_g P'\|Q} \qquad \frac{Q \stackrel{a}{\not\to}_g Q'}{P\|Q \stackrel{a}{\not\to}_g P\|Q'} \qquad \frac{P \stackrel{a}{\not\to}_g P',\, Q \stackrel{b}{\not\to}_g Q'}{P\|Q \stackrel{a|b}{\not\to}_g P'\|Q'}\, a|b \neq \bot$$

$$\frac{P \stackrel{a}{\not\to}_g P'}{P \Downarrow^d_H \stackrel{a}{\not\to}_g P' \Downarrow^d_H}\, a \notin H \qquad \frac{P \stackrel{a}{\not\to}_g P'}{P \Downarrow^d_H \stackrel{a}{\not\to}_g P'}\, a \in H \qquad \frac{P \stackrel{a}{\not\to}_g P'}{\partial_H(P) \stackrel{a}{\not\to}_g \partial_H(P')}\, a \notin H \qquad \frac{P[\mathtt{rec}X \cdot P/X] \stackrel{a}{\not\to}_g P'}{\mathtt{rec}X \cdot P \stackrel{a}{\not\to}_g P'}$$

Règles temporelles

$$\frac{}{\widetilde{a}P \stackrel{\chi}{\not\to}_g \widetilde{a}P} \qquad \frac{P \stackrel{\chi}{\not\to}_g P', Q \stackrel{\chi}{\not\to}_g Q'}{P+Q \stackrel{\chi}{\not\to}_g P'+Q'} \qquad \frac{}{\lfloor P\rfloor Q \stackrel{\chi}{\not\to}_g Q} \qquad \frac{P \stackrel{\chi}{\not\to}_g P'}{\partial_H(P) \stackrel{\chi}{\not\to}_g \partial_H(P')}$$

$$\frac{P \stackrel{\chi}{\not\to}_g P', Q \stackrel{\chi}{\not\to}_g Q'}{P\|Q \stackrel{\chi}{\not\to}_g P'\|Q'} \qquad \frac{P \stackrel{\chi}{\not\to}_g P'}{\lceil P\rceil^d_H Q \stackrel{\chi}{\not\to}_g \lceil P'\rceil^{d-g}_H Q}\, d > g \qquad \frac{P \stackrel{\chi}{\not\to}_g P'}{\lceil P\rceil^g_H Q \stackrel{\chi}{\not\to}_g Q}$$

$$\frac{P \stackrel{\chi}{\not\to}_g P',\, \forall a \in H\; P \stackrel{a}{\not\to}_g}{\mathtt{U}_H(P) \stackrel{\chi}{\not\to}_g \mathtt{U}_H(P')} \qquad \frac{P \stackrel{\chi}{\not\to}_g P'}{P \Downarrow^d_H \stackrel{\chi}{\not\to}_g P' \Downarrow^{d-g}_H}\, d > g \qquad \frac{P[\mathtt{rec}X \cdot P/X] \stackrel{\chi}{\not\to}_g P'}{\mathtt{rec}X \cdot P \stackrel{\chi}{\not\to}_g P'}$$

On constate aisément que la sémantique d'ATP$_1$ est identique à celle d'ATP, en comparant les règles temporelles du chien de garde, de l'urgence et de la restriction temporelles définies ci-dessus avec celles données dans le chapitre précédent pour ces opérateurs.

L'équivalence forte $\overset{g}{\sim}$ sur $\mathcal{P}_g$ et les modèles des processus sont définis de la même manière que pour ATP. Il est facile de vérifier que l'équivalence forte est toujours une congruence.

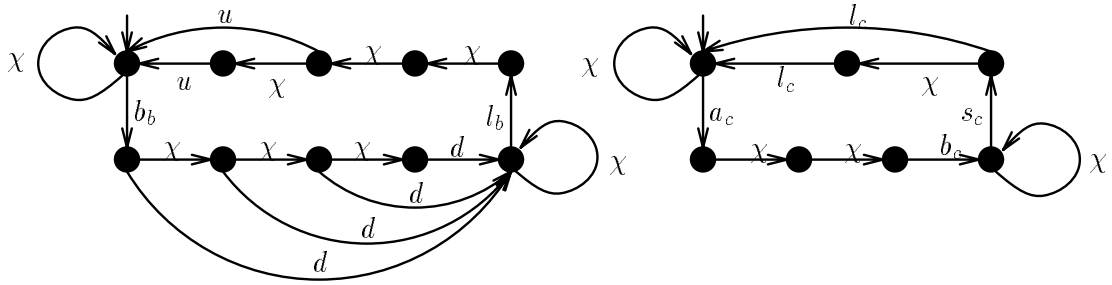
On peut également prouver que pour tout g, la classe des modèles d'ATP$_g$ est identique à celle d'ATP.

À titre d'exemple, nous présentons sur la figure 4.1 les modèles dans ATP$_{\frac{1}{2}}$ de la barrière et du contrôleur de passage à niveau (cf. section 3.3.4, page 68). Nous rappelons que la barrière est définie par

$$B \overset{\text{def}}{=} \widetilde{b_b}\,(B_1 \Downarrow^2_{\{d\}}) \qquad B_1 \overset{\text{def}}{=} \widetilde{d}\,\widetilde{l_b}\,(B_2 \Downarrow^2_{\{u\}}) \qquad B_2 \overset{\text{def}}{=} (1)\widetilde{u}\,B$$

et le contrôleur par

$$C \overset{\text{def}}{=} \widetilde{a_c}\,(1)b_c\,\widetilde{s_c}\;(\widetilde{l_c}\,C)\Downarrow^1_{\{l_c\}}$$



barrière contrôleur

Figure 4.1 : modèles dans ATP$_{\frac{1}{2}}$ de la barrière et du contrôleur

4.1.3 Axiomatisation

Tous les axiomes d'ATP donnés au chapitre 2 restent valides. Ceux de l'urgence (page 61) également. Pour le chien de garde, il faut généraliser les axiomes [⌈⌉5] et [⌈⌉6] (voir page 56), en remplaçant 1 par g :

$$[\lceil\rceil 5] \qquad \lceil\lfloor P\rfloor Q\rceil^g_H R \;\equiv\; \lfloor\lceil P\rceil^g_H R\rfloor R$$
$$[\lceil\rceil 6] \qquad \lceil\lfloor P\rfloor Q\rceil^{d+g}_H R \;\equiv\; \lfloor\lceil P\rceil^{d+g}_H R\rfloor\lceil Q\rceil^d_H R$$

Il faut procéder de même pour les axiomes [⇓5] et [⇓6] de la restriction temporelle (voir page 62) :

$$[\Downarrow 5] \qquad (\lfloor P\rfloor Q)\Downarrow^g_H \;\equiv\; P\Downarrow^g_H$$
$$[\Downarrow 6] \qquad (\lfloor P\rfloor Q)\Downarrow^{d+g}_H \;\equiv\; \lfloor P\Downarrow^{d+g}_H\rfloor(Q\Downarrow^d_H)$$

La proposition 2.5 reste vraie dans ATP$_g$: pour tout processus P d'ATP$_g$, il existe des termes $P_1, P_2, \ldots, P_n$ d'ATP$_g$, tels que $P \equiv P_1$ et chaque P_i est solution d'une équation dans une et une

seule des deux formes suivantes :

$$P_i \equiv \sum_{j \in J_i} a_j \, P_{j_i} \qquad \text{ou} \qquad P_i \equiv \left\lfloor \sum_{j \in J_i} a_j \, P_{j_i} \right\rfloor P_{k_i}$$

4.2 Les algèbres ATP_D

4.2.1 Non préservation des propriétés

Nous montrons à partir d'un exemple qu'en général, les propriétés des processus ne sont pas préservées lorsqu'on change de granularité.

Soit deux processus P et Q, qui peuvent communiquer via une action a de P et une action b de Q ($a|b = c$). Leur description informelle est la suivante :

- P essaie de communiquer avec Q une unité de temps après la réception d'un message de l'extérieur (action p). S'il n'a pas réussi avant une unité de temps, il s'arrête. Si la communication a eu lieu, il s'arrête également
- Q attend un message de l'extérieur (action q). S'il ne l'a pas reçu avant une unité de temps, il s'arrête. Dans le cas contraire, il essaie de communiquer avec P pendant une unité de temps, et s'arrête si ce n'est pas possible. Il s'arrête également après la communication.

En ATP_g, la définition des processus est

$$P \stackrel{\text{def}}{=} \widetilde{p}\,(1)\,(\widetilde{a}\,\delta \overset{1}{\triangleright} \delta) \qquad Q \stackrel{\text{def}}{=} \widetilde{q}\,\left(\widetilde{b}\,\delta \overset{1}{\triangleright} \delta\right) \overset{1}{\triangleright} \delta$$

Le système complet est $S \stackrel{\text{def}}{=} \partial_{\{a,b\}}(P \| Q)$ La figure 4.2 présente les modèles de P, Q et S dans le cas où $g = 1$. On constate que la communication est impossible.

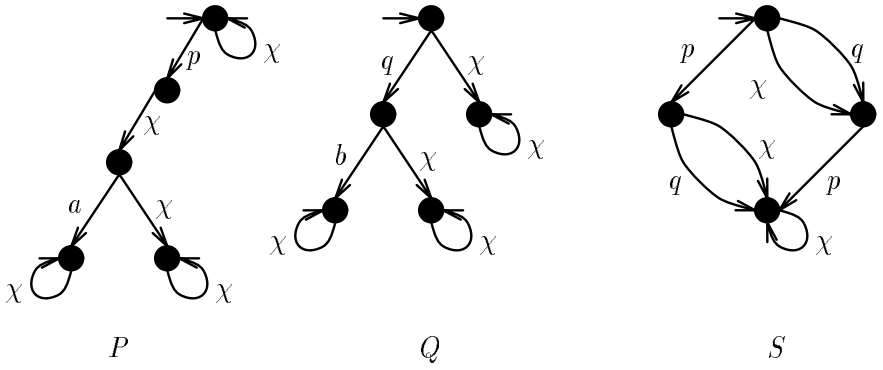


Figure 4.2 : communication impossible quand $g = 1$

Sur la figure 4.3 sont présentés les modèles des mêmes processus, mais avec $g = \frac{1}{2}$. La communication est cette fois possible.

La satisfaisabilité de la propriété "S peut exécuter l'action c" dépend donc de la granularité choisie.

Nous définissons ci-dessous des algèbres de processus temporisés paramétrées par un *domaine temporel* quelconque. Lorsque le domaine est *dense*, la sémantique opérationnelle ne peut plus

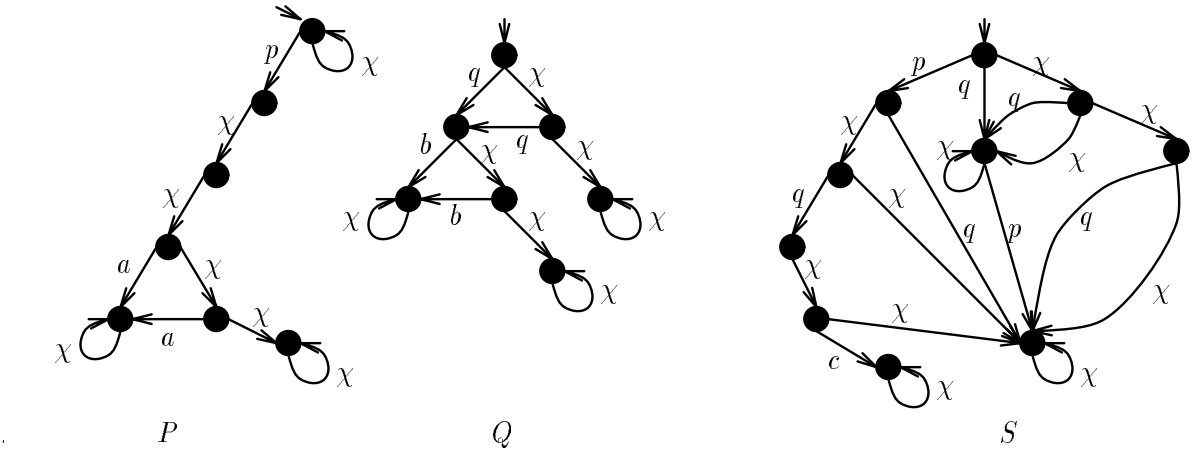


Figure 4.3 : communication possible quand $g = \frac{1}{2}$

être basée sur la notion de transition χ, puisque celle-ci représente la plus petite fraction de temps mesurable.

Nous présentons en premier lieu une définition formelle des domaines temporels.

4.2.2 Domaine temporel

Un domaine temporel est un monoïde commutatif $(D, 0, +)$ satisfaisant les deux propriétés suivantes :

1. pour tous d et d' dans D, $d + d' = d$ si et seulement si $d' = 0$;
2. le préordre $\leq$ défini sur D par

$$d \leq d' \iff \exists d'' : d + d'' = d'$$

est un *ordre total.*

Noter que les D_g ainsi que $\mathbb{Q}_+$ satisfont ces propriétés lorsque $+$ et $\leq$ sont respectivement l'addition et l'ordre naturel sur les rationnels.

La proposition suivante se démontre facilement.

Proposition 4.1

a. 0 est le plus petit élément de D.

b. Pour tous d et d' dans D, si $d \leq d'$ alors l'élément d'' tel que $d + d'' = d'$ est unique. On le note $d' - d$.

c. Le seul domaine temporel fini est $\{0\}$.

□

Preuve.

a. 0 étant l'élément neutre de $+$, on a donc $0 + d = d$ pour tout d. Donc, par définition de $\leq$, $0 \leq d$. Puisque $\leq$ est un ordre, 0 est donc l'unique plus petit élément de D.

b. Soient d_1 et d_2 tels que $d + d_1 = d'$ et $d + d_2 = d'$. L'ordre étant total, on peut supposer de plus $d_1 \leq d_2$. Il existe donc d_3 tel que $d_1 + d_3 = d_2$. On en déduit

$$(d + d_1) + d_3 = d + (d_1 + d_3) = d + d_2 = d' = (d + d_1)$$

La propriété 1 exigée ci-dessus entraîne alors $d_3 = 0$. On en conclut que $d_1 = d_2$.

c. Si D est fini, alors il existe un plus grand élément d_0 (l'ordre est total). Soit d quelconque dans D. On a alors $d_0 + d \leq d_0$. D'après la définition de $\leq$, il existe alors d' tel que $d_0 + d + d' = d_0$, et donc $d + d' = 0$. D'autre part, on a $d \leq d + d'$; comme 0 est le plus petit élément, on obtient $d = 0$. D est donc uniquement constitué de l'élément 0. ■

L'ensemble $D - \{0\}$ est noté D^*. Nous utilisons la notation $d < d'$ pour $d \leq d' \wedge d \neq d'$.

D est dit *dense* si

$$(\exists d, d' : d < d') \ \wedge \ [\forall d, d' : (d < d' \Rightarrow \exists d'' : d < d'' < d')]$$

D est dit *discret* si

$$\forall d, d' : [d < d' \Rightarrow \exists d'' : (d < d'' \ \wedge \ \forall d''' : d < d''' \Rightarrow d'' \leq d''')]$$

L'élément d'' est alors appelé le *successeur* de d ; il est unique car l'ordre est total. On le note $\mathtt{succ}(d)$.

Remarquer que selon ces définitions, les D_g sont discrets et $\mathbb{Q}_+$ et $\mathbb{R}_+$ sont denses.

Pour tout élément d de D et tout entier positif n, nous notons nd l'élément $\underbrace{d + \ldots + d}_{n \text{ fois}}$.

Pour $D \neq \{0\}$, nous supposons fixé un élément particulier u de D, appelé *unité*. L'ensemble $\{nu, n \in \mathbb{N}^*\}$ est noté U_D. Lorsque $D = \{0\}$, on pose $U_D \stackrel{\text{def}}{=} \emptyset$. Pour les D_g $\mathbb{Q}_+$ et $\mathbb{R}_+$, u est égal à 1, et donc U_D à $\mathbb{N}^*$.

4.2.3 Syntaxe et sémantique

Comme pour les ATP_g, les algèbres ATP_D sont définies à partir d'algèbres plus générales $\mathcal{P}_D$. Nous devons proscrire complètement l'emploi du délai unitaire qui n'a aucun sens lorsque le domaine temporel est dense.

Les éléments de $\mathcal{P}_D$ sont définis par la syntaxe suivante :

$$\begin{array}{rcl} P ::= & X & \mid \ a\,P \ \mid \ \widetilde{a}\,P \ \mid \ P + P \ \mid \ \lceil P \rceil_H^d P \ \mid \\ & \mathtt{U}_H(P) & \mid \ P \Downarrow_H^d \ \mid \ P \| P \ \mid \ \partial_H(P) \ \mid \ \mathtt{rec}X \cdot P \end{array}$$

où $X \in \mathcal{V}$, $a \in \mathcal{A}$, $d \in D^*$ et $H \subseteq \mathcal{A}$.

La définition des opérateurs dérivées de blocage, de terminaison, de retard, de délai de commencement, d'exécution et de terminaison est toujours inchangée.

Remarquer que pour toute granularité g, $\mathcal{P}_{D_g} \subset \mathcal{P}_g$.

De plus, si $D \subseteq D'$, alors $\mathcal{P}_D \subseteq \mathcal{P}_{D'}$.

La sous-algèbre ATP_D de $\mathcal{P}_D$ est l'ensemble des termes fermés, réguliers, bien gardés de $\mathcal{P}_D$, où les valeurs des délais sont dans U_D. Noter qu'ainsi, si $U_D = U_{D'}$, alors ATP_D et $\mathrm{ATP}_{D'}$ sont constituées des mêmes termes. Seuls les modèles des processus varient en fonction du domaine temporel. En particulier, $\mathrm{ATP}_{\mathbb{Q}_+}$, $\mathrm{ATP}_{\mathbb{R}_+}$ et les ATP_{D_g} sont syntaxiquement identiques.

La définition de la sémantique opérationnelle est inspirée des algèbres Timed CCS [Wan90, MT90]. Comme dans la section précédente, nous ne nous intéressons qu'aux termes fermés. Le cas des termes ouverts est traité exactement comme pour $\mathcal{P}_t$, au moyen d'étiquettes X^+ et X^-.

Les systèmes de transitions associés aux termes sont étiquetés par $\mathcal{L}_D \stackrel{\text{def}}{=} \mathcal{A} \cup D^*$. La relation de transition $\xrightarrow[D]{}$ est donc définie sur $\mathcal{P}_D \times \mathcal{L}_D \times \mathcal{P}_D$. Une transition $P \xrightarrow[D]{d} Q$ signifie que le processus P peut laisser passer le temps sur une durée d, en se comportant ensuite comme Q.

Un processus est dit *immédiat* s'il n'a de transition par aucun d ; dans le cas contraire, il est dit *retardable*.

Nous exigeons toujours le *déterminisme* du passage du temps. Cette propriété s'exprime ici par

$$\forall P, P', P'', \forall d : P \xrightarrow[D]{d} P' \ \wedge \ P \xrightarrow[D]{d} P'' \Rightarrow P' = P''$$

De plus, pour que la définition des transitions temporelles ait un sens, la sémantique doit être définie de telle sorte que

$$\forall P, P', \forall d, d' > 0 : \left(\exists P'' : P \xrightarrow[D]{d} P'' \ \wedge \ P'' \xrightarrow[D]{d'} P'\right) \iff P \xrightarrow[D]{d+d'} P'$$

Cette propriété est appelée *continuité temporelle* dans [Wan90]. Nous préférons utiliser le terme d'*additivité*.

Les règles d'action de la sémantique sont les mêmes que pour $\mathcal{P}_g$ (en supprimant celle du délai unitaire). Les règles temporelles sont les suivantes :

$\dfrac{}{\widetilde{a}\,P \xrightarrow[D]{d} \widetilde{a}\,P}$	$\dfrac{P \xrightarrow[D]{d} P', Q \xrightarrow[D]{d} Q'}{P+Q \xrightarrow[D]{d} P'+Q'}$	$\dfrac{P \xrightarrow[D]{d} P'.P \xrightarrow[D]{d} Q'}{P\|Q \xrightarrow[D]{d} P'\|Q'}$	$\dfrac{P \xrightarrow[D]{d} P'}{\partial_H(P) \xrightarrow[D]{d} \partial_H(P')}$
$\dfrac{P \xrightarrow[D]{d} P'}{\lceil P \rceil_H^{d+d'} Q \xrightarrow[D]{d} \lceil P' \rceil_H^{d'} Q}$	$\dfrac{P \xrightarrow[D]{d} P'}{\lceil P \rceil_H^{d} Q \xrightarrow[D]{d} Q}$	$\dfrac{P \xrightarrow[D]{d} P', Q \xrightarrow[D]{d'} Q'}{\lceil P \rceil_H^{d} Q \xrightarrow[D]{d+d'} Q'}$	$\dfrac{P[\mathtt{rec}X \cdot P/X] \xrightarrow[D]{d} P'}{\mathtt{rec}X \cdot P \xrightarrow[D]{d} P'}$
$\dfrac{P \xrightarrow[D]{d} P'}{P \Downarrow_H^{d+d'} \xrightarrow[D]{d} P' \Downarrow_H^{d'}}$	$\dfrac{P \xrightarrow[D]{d} P', \forall a \in H, [P \xrightarrow[D]{a} \ \wedge \ \forall d' < d, (P \xrightarrow[D]{d'} P'' \Rightarrow P'' \xrightarrow[D]{a})]}{\mathtt{U}_H(P) \xrightarrow[D]{d} \mathtt{U}_H(P')}$		

La règle de l'opérateur d'urgence peut paraître compliquée. Elle s'explique de la manière suivante : P n'a pas le droit de laisser passer le temps s'il peut exécuter une des actions de H. En conséquence, il ne peut être inactif pendant une durée d que s'il ne peut pas exécuter d'action de H avant l'instant d.

L'équivalence forte $\stackrel{D}{\sim}$ est à présent définie en considérant les éléments de $\mathcal{L}_D$ pour les étiquettes. Les règles de sémantique étant dans le format GSOS, $\stackrel{D}{\sim}$ est encore une congruence sur $\mathcal{P}_D$.

On peut montrer facilement que pour toute granularité g, les équivalences $\stackrel{g}{\sim}$ et $\stackrel{Dg}{\sim}$ coïncident sur les termes de $\mathcal{P}_{D_g}$:

$$\forall P, P' \in \mathcal{P}_{D_g} : P \stackrel{g}{\sim} P' \iff P \stackrel{Dg}{\sim} P'$$

4.2.4 Propriétés de la relation de transition

4.2.4.1 Déterminisme et additivité

On peut aisément vérifier que la relation de transition définie par les règles ci-dessus satisfait les deux propriétés de déterminisme et d'additivité.

Ces deux propriétés permettent de relier les systèmes de transitions associés par les sémantiques de $\mathcal{P}_g$ et $\mathcal{P}_{D_g}$ à un terme de $\mathcal{P}_{D_g}$ de la manière suivante :

$$\begin{array}{l}\forall P,Q \in \mathcal{P}_{D_g} : \\ \quad \forall a : P \xrightarrow[D_g]{a} Q \iff P \xrightarrow[g]{a} Q \\ \quad \forall k \in \mathbb{N}^* : P \xrightarrow[D_g]{kg} Q \iff \exists Q_1,\ldots,Q_k \in \mathcal{P}_{D_g} : Q_k = Q \ \wedge\ P \xrightarrow[g]{\chi} Q_1 \xrightarrow[g]{\chi} \ldots \xrightarrow[g]{\chi} Q_k\end{array}$$

4.2.4.2 Persistance partielle

Lorsque le domaine temporel est dense, une autre propriété des systèmes de transitions est que la possibilité d'exécuter une action ne peut pas apparaître et disparaître "instantanément" ; toute action reste possible durant un certain temps (à moins qu'une autre action soit exécutée immédiatement). Donc, si un processus est retardable, toute action est exécutable pendant un durée non nulle. Cette propriété est appelée *persistance partielle* ; formellement, elle s'écrit :

$$\forall P,P',\forall d : P \xrightarrow[D]{d} P' \Rightarrow\ \exists d' > 0,\forall a,\forall P'',\forall d'' \in]0,d'[,\ \exists P_1,P_2 : P \xrightarrow[D]{a} P'' \Rightarrow P \xrightarrow[D]{d''} P_1 \xrightarrow[D]{a} P_2$$

Remarquer que lorsque D est dense, si d' est strictement plus grand que 0, l'intervalle $]0,d'[$ est non vide. Remarquer également que cette propriété est toujours satisfaite dans le cas d'un domaine temporel discret : il suffit de choisir $d' = \mathtt{succ}(0)$.

Dans le cas général, la relation entre P'' et P_2 dans la formule ci-dessus est compliquée. Elle devient triviale dans le cas des *processus séquentiels* définis de la manière suivante :

Définition 4.1 (processus séquentiels)

Soit la sous-algèbre de $\mathcal{P}_D$ notée $\mathcal{P}^s_D$, définie par la syntaxe

$$P ::= \quad X \quad | \quad a\,P \quad | \quad \widetilde{a}\,P \quad | \quad P + P \quad | \quad P \overset{d}{\triangleright} P \quad | \quad \mathtt{U}_H(P) \quad | \quad \partial_H(P) \quad | \quad \mathtt{rec}X \cdot P$$

où $X \in \mathcal{V}$, $a \in \mathcal{A}$, $d \in D$ et $H \subseteq \mathcal{A}$.

L'algèbre ASTP_D est la sous-algèbre de $\mathcal{P}^s_D$, constituée des termes fermés, bien gardés, sans récursion à travers l'encapsulation, et dont les valeurs des délais appartiennent à U_D.

□

La propriété de persistance partielle pour $\mathcal{P}^s_D$ s'exprime alors de la manière suivante :

$$\forall P,P',\forall d : P \xrightarrow[D]{d} P' \Rightarrow\ \left(\exists d' > 0,\forall a,\forall P'',\forall d'' \in]0,d'[,\ \exists P_1 : P \xrightarrow[D]{a} P'' \Rightarrow P \xrightarrow[D]{d''} P_1 \xrightarrow[D]{a} P''\right)$$

En d'autres termes, si P est retardable et s'il peut exécuter l'action a, alors il existe une durée non nulle pendant laquelle a est possible, le comportement ultérieur étant le même quel que soit l'instant d'exécution de a.

Dans le cas d'ASTP_D, on peut montrer que tout processus retardable l'est pendant une durée au moins égale à l'unité u, le comportement ultérieur étant dans ASTP_D :

$$\forall P \in \mathrm{ASTP}_D,\forall d \in D,\forall P' \in \mathcal{P}^s_D : P \xrightarrow[D]{d} P' \Rightarrow \exists P_1 \in \mathrm{ASTP}_D : P \xrightarrow[D]{u} P_1$$

De plus, tout processus retardable d'ASTP_D peut exécuter ses actions initiales jusqu'à tout instant strictement inférieur à u. En utilisant la propriété précédente, ceci s'exprime formellement par :

$$\begin{array}{l}\forall P,P' \in \mathrm{ASTP}_D : \\ \quad P \xrightarrow[D]{u} P' \Rightarrow \left(\forall a,\forall P'' \in \mathrm{ASTP}_D,\ P \xrightarrow[D]{a} P'' \Rightarrow \forall d' \in]0,u[,\ \exists P_1 \in \mathcal{P}^s_D : P \xrightarrow[D]{d'} P_1 \xrightarrow[D]{a} P''\right)\end{array}$$

4.2.4.3 Préservation des propriétés de sûreté

Informellement, une propriété de sûreté affirme que quelque chose de "mauvais" ne survient jamais au cours de l'exécution d'un programme.

Dans [MP89, AL88], ces propriétés sont définies relativement à une sémantique de programmes définissant les modèles comme des ensembles de traces (sémantique linéaire). Une propriété de sûreté est alors un ensemble *fermé* (dans la topologie de Cantor) de séquences infinies d'états.

Cette définition est étendue dans [BFG+91], en particulier au cas où la sémantique des programmes est associée à une équivalence de bisimulation, comme l'équivalence forte (sémantique arborescente). Les modèles sont alors des systèmes de transitions étiquetées. La définition topologique des propriétés est relative à un préordre $\ll$ sur les arbres d'exécution.

Dans le cas qui nous intéresse, $\ll$ est le préordre de *simulation forte* (ou relation d'implantation sûre [Rod88]), noté $\sqsubseteq$, défini entre les modèles des termes de $\mathcal{P}_D$ et $\mathcal{P}_{D'}$ (avec $\mathcal{L}_D \subseteq \mathcal{L}_{D'}$) par :

$$P \sqsubseteq P' \iff \forall \ell \in \mathcal{L}_D, \forall P_1 \in \mathcal{P}_D : P \xrightarrow[D]{\ell} P_1 \Rightarrow \exists P'_1 \in \mathcal{P}_{D'} : P' \xrightarrow[D']{\ell} P'_1 \ \wedge \ P_1 \sqsubseteq P'_1$$

La définition des propriétés de sûreté est telle que si $P \sqsubseteq P'$, alors toute propriété de sûreté satisfaite par P' l'est également par P.

La proposition suivante se démontre facilement à partir des règles de sémantique des $\mathcal{P}_D$.

Proposition 4.2

Si D et D' sont deux domaines temporels tels que $D \subseteq D'$, on sait que $\mathcal{P}_D \subseteq \mathcal{P}_{D'}$ et $\mathcal{L}_D \subseteq \mathcal{L}_{D'}$. On a alors, pour tous termes P et P' de $\mathcal{P}_D$, pour toute étiquette ℓ de $\mathcal{L}_D$,

$$P \xrightarrow[D]{\ell} P' \Rightarrow P \xrightarrow[D']{\ell} P'$$

□

Les modèles de P dans la sémantique de $\mathcal{P}_D$ et dans celle de $\mathcal{P}_{D'}$ sont donc en relation par le préordre de simulation forte. En conséquence, toute propriété de sûreté satisfaite par P dans $\mathcal{P}_{D'}$ l'est également dans $\mathcal{P}_D$.

En particulier, on peut conclure que si un processus P d'ATP$_{\mathbb{Q}_+}$ satisfait une propriété de sûreté, alors celle-ci reste vraie pour le modèle de P dans tous les ATP$_{D_g}$. Il suffit donc de démontrer une propriété de sûreté dans le cadre du domaine temporel $\mathbb{Q}_+$ pour qu'elle soit prouvée pour toute granularité.

4.2.5 Axiomatisation des processus séquentiels

Lorsque le domaine temporel est discret, l'axiomatisation d'ATP$_D$ est similaire à celle des ATP$_g$, car on peut réintroduire le délai unitaire, qui expire à l'instant $\mathtt{succ}(0)$.

Dans le cas général, certains des axiomes des ATP$_g$ concernant les opérateurs de composition parallèle, de chien de garde et de restriction temporelle restent également valides (en les adaptant à ATP$_D$). Par exemple, on a toujours $(a\,P)\|Q \equiv a\,(P\|Q)$ et $\lceil a\,P \rceil^d_H Q \equiv a\,\lceil P \rceil^d_H Q$ si $a \notin H$. Par contre, tous les axiomes faisant intervenir le délai unitaire n'ont plus de raison d'être ; de plus, on ne peut pas leur substituer des axiomes mettant en jeu le délai de commencement.

En particulier, si le domaine temporel est dense, il est impossible d'obtenir un théorème d'expansion pour la composition parallèle, le chien de garde général ou la restriction temporelle, c'est-à-dire de trouver un terme séquentiel équivalent à un processus où apparaissent ces

opérateurs. Pour ce faire, il est nécessaire d'étendre l'algèbre en autorisant des délais paramétrés par des variables temporelles libres. Celles-ci sont liées par l'opérateur de préfixage : la valeur du délai dépend de l'instant d'exécution de l'action du préfixage. Cette idée a été développée par Wang [Wan91] (voir § 4.3.2.2). On obtient alors des théorèmes d'expansion relativement complexes pour la composition parallèle et le chien de garde, mais ce n'est toujours pas possible pour la restriction temporelle.

Nous n'envisageons donc pas d'axiomatiser les algèbres ATP_D. Nous nous restreignons aux processus d'ASTP_D, pour lesquels l'ensemble (non nécessairement minimal) d'axiomes suivant est complet.

$$
\begin{array}{rl}
[+_d1] & P + Q \equiv Q + P \\
[+_d2] & P + (Q + R) \equiv (P + Q) + R \\
[+_d3] & P + P \equiv P \\
[+_d4] & P + \delta \equiv P \\
[+_d5] & \mathbf{0} + a\,P \equiv a\,P \\
[+_d6] & \mathbf{0} + \widetilde{a}\,P \equiv a\,P \\
[+_d7] & \mathbf{0} + P \overset{d}{\triangleright} Q \equiv \mathbf{0} + P \\
[+_d8] & P_1 \overset{d}{\triangleright} Q_1 + P_2 \overset{d}{\triangleright} Q_2 \equiv (P_1 + P_2) \overset{d}{\triangleright} (Q_1 + Q_2)
\end{array}
$$

$$
\begin{array}{rl}
[\,\triangleright_d1] & \mathbf{0} \overset{d}{\triangleright} Q \equiv \mathbf{0} \\
[\,\triangleright_d2] & \delta \overset{d}{\triangleright} \delta \equiv \delta \\
[\,\triangleright_d3] & \widetilde{a}\,P \overset{d}{\triangleright} \widetilde{a}\,P \equiv \widetilde{a}\,P \\
[\,\triangleright_d4] & (P \overset{d}{\triangleright} Q) \overset{d'}{\triangleright} R \equiv P \overset{d'}{\triangleright} R \qquad \text{si } d \geq d' \\
[\,\triangleright_d5] & (P \overset{d}{\triangleright} Q) \overset{d'}{\triangleright} R \equiv P \overset{d}{\triangleright} Q \overset{d'-d}{\triangleright} R \qquad \text{si } d < d'
\end{array}
$$

$$
\begin{array}{rl}
[\mathtt{U}_d1] & \mathtt{U}_H(P + Q) \equiv \mathtt{U}_H(P) + \mathtt{U}_H(Q) \\
[\mathtt{U}_d2] & \mathtt{U}_H(\mathbf{0}) \equiv \mathbf{0} \\
[\mathtt{U}_d3] & \mathtt{U}_H(a\,P) \equiv a\,\mathtt{U}_H(P) \\
[\mathtt{U}_d4] & \mathtt{U}_H(\widetilde{a}\,P) \equiv \widetilde{a}\,\mathtt{U}_H(P) \qquad \text{si } a \notin H \\
[\mathtt{U}_d5] & \mathtt{U}_H(\widetilde{a}\,P) \equiv a\,\mathtt{U}_H(P) \qquad \text{si } a \in H \\
[\mathtt{U}_d6] & \mathtt{U}_H(P \overset{d}{\triangleright} Q) \equiv \mathtt{U}_H(P) \overset{d}{\triangleright} \mathtt{U}_H(Q)
\end{array}
$$

$$
\begin{array}{rl}
[\partial_d1] & \partial_H(P + Q) \equiv \partial_H(P) + \partial_H(Q) \\
[\partial_d2] & \partial_H(\mathbf{0}) \equiv \mathbf{0} \\
[\partial_d3] & \partial_H(a\,P) \equiv a\,\partial_H(P) \qquad \text{si } a \notin H \\
[\partial_d4] & \partial_H(a\,P) \equiv \mathbf{0} \qquad \text{si } a \in H \\
[\partial_d5] & \partial_H(\widetilde{a}\,P) \equiv \widetilde{a}\,\partial_H(P) \qquad \text{si } a \notin H \\
[\partial_d6] & \partial_H(\widetilde{a}\,P) \equiv \delta \qquad \text{si } a \in H \\
[\partial_d7] & \partial_H(P \overset{d}{\triangleright} Q) \equiv \partial_H(P) \overset{d}{\triangleright} \partial_H(Q)
\end{array}
$$

$$
\begin{array}{rl}
[\mathtt{rec}_d1] & \mathtt{rec}X \cdot P \equiv P[\mathtt{rec}X \cdot P/X] \\
[\mathtt{rec}_d2] & \text{si } P[Q/X] \equiv Q \text{ alors } \mathtt{rec}X \cdot P \equiv Q
\end{array}
$$

À l'aide de ces axiomes, on peut définir une *forme canonique* des processus d'ASTP_D :

Proposition 4.3 (forme canonique pour ASTP_D)
Pour tout P, il existe $P_1, \ldots, P_n$ dans ASTP_D, tels que $P \equiv P_1$, et chaque P_1 est solution d'une équation dans une des trois formes suivantes :

$$[1] \quad P_i \equiv \sum_{j \in J_i} a_i P_{j_i}$$

$$[2] \quad P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i}$$

$$[3] \quad P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \quad \overset{d_i}{\triangleright} P_{k_i} \quad \text{où } d_i = n_i u$$

Comme pour ATP, si J_i est vide, on obtient $\mathbf{0}$ dans le cas [1], δ dans le cas [2], et $\delta \overset{d_i}{\triangleright} P_{k_i}$ dans le cas [3].
□

La forme de l'équation de P_i n'est pas nécessairement unique. En particulier, à toute équation de la forme [2] correspond une équation de la forme [3] :

$$\text{si } P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \quad \text{alors} \quad P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \overset{d_i}{\triangleright} P_i \quad \text{quel que soit } d_i$$

De même, si

$$P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \overset{d_i}{\triangleright} P_k \quad \text{et} \quad P_k \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \overset{d_k}{\triangleright} P_l \quad \text{où } d_i = n_i u \text{ et } d_k = n_k u$$

alors on peut réécrire l'équation de P_i en

$$P_i \equiv \sum_{j \in J_i} \widetilde{a}_i P_{j_i} \overset{d'_i}{\triangleright} P_l \quad \text{où } d'_i = (n_i + n_k) u$$

On peut définir une forme canonique "minimale" en imposant que [2] soit préférée à [3], et que dans [3], la valeur du délai soit la plus grande possible.

4.3 Étude d'autres algèbres

Nous terminons ce chapitre par une comparaison d'ATP_D avec plusieurs algèbres de processus temporisés, en nous concentrant principalement sur l'influence des choix de la sémantique sur la structure des modèles.

Nous ne prétendons pas mener une étude exhaustive de l'ensemble des recherches en cours dans ce domaine. Nous avons sélectionné un ensemble d'algèbres dont les caractéristiques reflètent les divers choix syntaxiques et sémantiques possibles. Ces algèbres sont, par ordre alphabétique :

- ACP_ρ (Real Time ACP) de J. C. M. Baeten et J. A. Bergstra [BB90], dont une forme restreinte est étudiée en détail dans [Klu91].
- TCSP (Timed CSP) de G. M. Reed et A. W. Roscoe [RR88, DS89], dont une sémantique opérationnelle est décrite dans [Sch91].
- TeCCS (Timed CCS) de F. Moller et C. Tofts [MT90] ; les auteurs utilisent l'abréviation TCCS.
- TiCCS (Temporal CCS) de Wang Yi [Wan90], étendue dans [Wan91] pour obtenir une théorème d'expansion de la composition parallèle. L'auteur utilise également l'abréviation TCCS.

- TPCCS (Timed Probabilistic CCS) de H. Hansson et B. Jonsson [HJ90, Han91]. Cette algèbre comporte également des éléments permettant de traiter les probabilités d'exécution des actions. Nous ne nous intéressons ici qu'aux aspects temporels, qui sont orthogonaux aux aspects probabilistes.
- TPL (Temporal Process Language) de M. Hennessy et T. Regan [HR90]. L'algèbre est enrichie dans [HR91] au moyen du délai unitaire d'ATP.
- U-LOTOS (Urgent LOTOS) de T. Bolognesi et F. Lucidi [BL91].

Le formalisme utilisé pour la définition de la sémantique varie suivant l'algèbre. Il est cependant possible de se ramener au cadre unifié de la sémantique des ATP_D, ce que nous faisons dans cette étude.

4.3.1 Le domaine temporel

La syntaxe de TPCCS et de TPL impose, comme celle d'ATP, l'utilisation d'un domaine discret (les naturels).

Les algèbres TeCCS, TiCCS et U-LOTOS sont définies relativement à un domaine temporel quelconque. Le formalisme de description de la sémantique est similaire à celui d'ATP_D (l'idée originale apparaît simultanément dans [MT90] et [Wan90] ; la propriété d'additivité est mise en évidence par Wang). Moller et Tofts proposent une axiomatisation dans le cas discret, et Wang dans le cas dense, en faisant varier les délais en fonction de l'instant d'exécution des actions.

Finalement, les algèbres TCSP et ACP_ρ sont définies explicitement sur un domaine temporel dense (les réels ou les rationnels), mais elles peuvent sans problème être adaptées à un domaine discret. Seuls quelques axiomes sont présentés pour TCSP. En revanche, on trouve une axiomati –sation complète d'ACP_ρ, y compris des axiomes d'élimination de la composition parallèle, dans [Klu91]

Toutes les algèbres sauf TCSP adoptent implicitement les principes d'extension d'une algèbre non temporisée présentés au chapitre 2 (voir section 2.1, page 36). Dans TCSP un délai minimal δ est imposé après l'exécution de chaque action, et pour "dérouler" une récursion. Avec la syntaxe d'ATP, cela revient à écrire :

$$\widetilde{a}\,P \xrightarrow{a} (\delta)P \qquad \text{et} \qquad \mathtt{rec}X \cdot P \equiv (\delta)[\mathtt{rec}X \cdot P/X]$$

Les axiomes de CSP ne sont donc plus valides dans TCSP. Dans [Sch91], S. Schneider explique que ce délai minimal est nécessaire pour définir correctement la sémantique dénotationnelle présentée dans [DS89], mais qu'on peut le négliger si les modèles sont des systèmes de transitions étiquetées.

En fait, l'introduction de δ revient à considérer que les actions prennent un temps minimal δ. Deux occurrences d'actions d'un processus séquentiel sont donc distantes temporellement d'au moins δ, alors que deux actions de deux processus en parallèle peuvent être arbitrairement rapprochées dans le temps. Par conséquent, il n'y a pas de théorème d'expansion dans TCSP (même si le domaine temporel est discret). En d'autres termes, il n'est pas possible d'utiliser l'opérateur de composition parallèle pour décrire un processus constitué d'entités plus ou moins indépendantes, mais destiné à être exécuté séquentiellement. Les adeptes de cette philosophie parlent de *parallélisme réel* : deux termes mis en parallèle dans la spécification *doivent* être exécutés sur deux processeurs distincts.

4.3.2 Choix des opérateurs

L'algèbre ACP_ρ est une extension d'ACP. TCSP est une extension de CSP [Hoa78]. TeCCS, TiCCS, TPCCS et TPL étendent l'algèbre CCS. Enfin, U-LOTOS est basée sur un sous-ensemble du langage LOTOS [ISO88b], et est définie de manière à transposer dans un cadre algébrique les réseaux de Petri à transitions temporisées [MF76].

4.3.2.1 Caractéristiques de l'algèbre d'origine

Les opérateurs non temporels dépendent bien entendu de l'algèbre choisie initialement :

- TCSP et ACP_ρ possèdent un opérateur de composition séquentielle, tandis que le préfixage est adopté dans les autres algèbres.
- La terminaison et le blocage sont deux opérateurs différents de CSP, qui le restent dans TCSP.

  CCS et LOTOS utilisent un seul opérateur (respectivement Nil et STOP) pour les deux fonctionnalités. L'introduction du temps permet d'opérer la distinction (voir page 41), ce qui n'est pas fait dans toutes les algèbres (voir ci-dessous).

  ACP ne possède initialement que le blocage δ. Certaines sémantiques emploient la terminaison ϵ (cf. remarque 1.2, page 32), mais il est possible de s'en passer ; il n'est pas pris en compte dans ACP_ρ.
- Dans TCSP sont présents le choix interne et le choix externe de CSP. Le choix interne est résolu immédiatement par l'exécution d'une action interne τ ; le choix externe n'est résolu que lors de l'exécution d'une action visible.

  Les autres algèbres ne comportent qu'un équivalent du choix externe, qui est également résolu si une action interne est exécutée (comme dans AUP). Nous ne considérons pas ici le choix interne, sur lequel l'introduction du temps n'a pas d'influence.
- L'opérateur de composition parallèle est celui de CCS pour TeCCS, TiCCS, TPCCS et TPL : une communication n'a lieu qu'entre une action a et son "complémentaire" $\overline{a}$, résultant en une action interne τ.

  Dans ACP_ρ, la composition parallèle est celle d'ACP, comme dans ATP ; il est possible de simuler la composition parallèle de CCS : on pose $a|\overline{a} = \tau$ pour tout a.

  Finalement, TCSP et U-LOTOS emploient la composition parallèle de CSP : l'opérateur est paramétré par un ensemble d'actions sur lesquelles les deux processus P et Q doivent se synchroniser ; si a est dans cet ensemble, P et Q ne peuvent exécuter a que simultanément, l'action résultante étant a. Ce mode de synchronisation n'est pas simulable directement par la composition parallèle d'ACP, car il faudrait supprimer la possibilité d'évolution indépendante des deux processus par l'action a, sans toutefois interdire son exécution par communication. On peut contourner le problème en renommant les actions a de P et Q en a_P et a_Q, et en posant $a_P|a_Q = a$.

4.3.2.2 Opérateurs additionnels

Du point de vue de la méthode d'introduction du passage du temps, on peut classer les algèbres dans trois catégories :

- L'opérateur de retard est primitif dans TCSP, TiCCS, TPL, TeCCS et U-LOTOS, avec des syntaxes variées.
- TPCCS et TPL comportent le délai unitaire d'ATP comme opérateur de base. Remarquer que le retard n'est donc pas nécessaire dans TPL en tant qu'opérateur primitif.
- L'approche la plus différente des autres est celle choisie dans ACP_ρ. À chaque occurrence d'action d'un processus est attachée une *étiquette temporelle* absolue ou relative : $a(3)$ signifie que l'action a peut être exécutée exactement trois unités de temps après le début de l'exécution du processus (instant 0) ; $a[3]$ signifie que a peut être exécutée exactement trois unités de temps après l'instant où a devient "possible". Ainsi, le processus $a(3) \cdot b(5)$ exécute a à l'instant 3 et b à l'instant 5, tandis que $a[3] \cdot b[5]$ exécute a à l'instant 3 et b à l'instant 8. Il est possible, mais déconseillé de mélanger les deux types d'étiquettes dans un terme. La théorie est essentiellement développée en considérant des instants absolus.

  Une étiquette temporelle est également associée au blocage : $\delta(4)$ représente un blocage à l'instant 4.

  Pour modéliser un intervalle possible d'exécution d'une action, un opérateur d'*intégration* est introduit : $\int_{v \in]3,7]} a(v)$ peut exécuter a à tout moment entre les instants 3 (exclu) et 7 (inclus). Dans [BB90], l'intégration est très générale : on peut écrire $\int_{v \in V} P$ où V est un sous-ensemble quelconque de $\mathbb{R}_+$ et P un processus quelconque. En revanche, pour obtenir une axiomatisation complète, et donc pour rendre la théorie décidable, Klusener [Klu91] restreint son utilisation à des expressions $\int_{v \in I} a(v)$, où I est un intervalle. v est appelée une variable temporelle ; elle est *liée* par l'intégration.

  Les bornes des intervalles peuvent faire intervenir d'autres variables temporelles précédemment liées : dans le terme $\int_{v \in [3,5]} a(v) \cdot \int_{w \in [v+1, v+2]} b(w)$, b est exécutée entre 1 et 2 unités de temps après a qui, elle est exécutée entre les instants 3 et 5.

  Dans TiCCS, Wang utilise une approche qu'on peut comparer à l'intégration, mais moins générale, et uniquement destinée à l'obtention d'un théorème d'expansion. Le préfixage est noté $a@v\,P$, où v est une variable temporelle. Un tel terme correspond dans ACP_ρ à $\int_{v \in [0,\infty]} a[v] \cdot P$. Les paramètres des retards peuvent comporter des variables temporelles liées par un préfixage : dans le terme $a@v\ \epsilon(2+v)\,P$, si a est exécutée à l'instant 4, alors un retard de durée 6 est introduit avant l'exécution de P (le retard (d) de TeCCS et d'ATP est noté $\epsilon(d)$ dans TiCCS).

Concernant l'urgence des actions, on peut distinguer deux catégories :

- Dans TiCCS, TPCCS et TPL et TCSP, les actions visibles sont toujours retardables ; le préfixage correspond dans ATP à $\widetilde{a}\,P$. En revanche, les action internes τ sont immédiates, correspondant dans ATP à un préfixage $\tau\,P$.
- Dans TeCCS, ACP_ρ et U-LOTOS, toutes les actions peuvent être immédiates ou retardables, la syntaxe différant selon l'algèbre :
  - Moller et Tofts adoptent le préfixage immédiat comme opérateur de base de TeCCS. Il est possible de rendre retardable l'exécution des actions en utilisant l'opérateur de *délai arbitraire* δ : Le terme $\widetilde{a}\,P$ d'ATP s'écrit $\delta\,a\,P$.
  - Comme nous l'avons déjà vu, dans ACP_ρ l'action a est exécutée immédiatement dans $a[0]$, et n'importe quand dans $\int_{v \in [0,\infty]} a[v]$.

– À l'inverse de Moller et Tofts, Bolognesi et Lucidi considèrent le préfixage retardable comme primitif dans U-LOTOS. L'urgence des actions est exprimée au moyen de l'opérateur `asap` (as soon as possible) qui a fourni l'idée de l'opérateur d'urgence d'ATP.

Les algèbres permettant d'exprimer l'urgence des actions (TeCCS, U-LOTOS, ACP_ρ) comportent toutes un opérateur primitif ou dérivé de blocage, correspondant au **0** d'ATP. TCSP n'autorise pas l'urgence des actions visibles, mais le blocage étant présent dans CSP, il le reste dans TCSP. Dans TPL, TiCCS et TPCCS, un blocage ne peut jamais survenir ; en effet, les actions visibles sont toujours retardables, donc une encapsulation (qui ne s'applique pas aux actions internes) ne peut pas bloquer un processus.

Pour le choix non déterministe, il est possible d'adopter un choix fort ou un choix faible. Le premier correspond à celui d'ATP. Pour le second, noté $\oplus$, il faut ajouter des règles de sémantique qui s'expriment dans ATP par :

$$\frac{P \xrightarrow{\chi} P', Q \not\xrightarrow{\chi}}{P \oplus Q \xrightarrow{\chi} P'} \qquad \text{et} \qquad \frac{Q \xrightarrow{\chi} Q', P \not\xrightarrow{\chi}}{P \oplus Q \xrightarrow{\chi} Q'}$$

- Le choix fort est retenu dans U-LOTOS, TPL, TCSP, TPCCS et TiCCS.
- Les deux opérateurs sont présents dans TeCCS. Le choix faible est nécessaire pour supprimer la possibilité d'exécution d'une action en laissant le temps passer. En revanche, il est possible de se débarrasser du choix fort, mais pas de l'exprimer comme opérateur dérivé.
- Dans ACP_ρ, il est naturel d'utiliser le choix faible. En effet, pour décrire un processus qui peut exécuter a à l'instant 2, ou b à l'instant 7, il faut écrire $a(2) \oplus b(7)$. Si le choix était fort, b ne pourrait jamais être exécuté.

4.3.3 Propriétés des modèles

Nous étudions à présent les conséquences des choix sémantiques des différentes algèbres sur la classe des systèmes de transitions. Plus cette classe est grande, plus l'algèbre est expressive.

4.3.3.1 Déterminisme et additivité

Toutes les sémantiques proposées assurent le déterminisme et l'additivité des transitions temporelles. L'additivité est implicite dans TPCCS et TPL, où le formalisme utilisé pour la relation de transition est similaire à celui d'ATP (transitions χ).

4.3.3.2 Blocage

Les algèbres sans opérateur de blocage (primitif ou dérivé) produisent bien entendu des modèles sans état puits. C'est le cas de TiCCS, TPL et TPCCS. Il n'est alors pas possible de détecter une impossibilité de communication, puisque celle-ci se traduit par une terminaison normale.

Inversement, la possibilité de blocage entraîne l'existence d'états où le temps ne peut pas passer dans les modèles de TCSP, ACP_ρ, U-LOTOS et TeCCS, comme dans ceux d'ATP.

4.3.3.3 Urgence

Une propriété importante des modèles de TPCCS, TCSP, TiCCS et TPL est l'impossibilité de spécifier l'urgence des actions autre qu'internes. Tout état comportant une transition par une action visible possède forcément une transition temporelle. Il est donc impossible de décrire un système qui *doit* exécuter des actions ; on ne peut que spécifier que ces actions *peuvent* avoir lieu. En d'autres termes, un processus a toujours la possibilité de ne rien faire du tout (du moins rien de visible). Ce mode de fonctionnement surprenant semble en contradiction avec le principe fondamental des systèmes temps réel, qui doivent en général réagir le plus rapidement possible aux stimuli externes.

De même que pour ATP, l'urgence des comportements est spécifiable dans les algèbres ACP_ρ, TeCCS et U-LOTOS (à l'aide de l'opérateur `asap` dans le dernier cas).

4.3.3.4 Persistance

La propriété de *persistance* stipule que le passage du temps ne peut pas supprimer la possibilité d'exécuter une action. Formellement, elle s'exprime par

$$\forall P, P', P_1, \forall a\, \forall d : P \xrightarrow{a} P' \wedge P \xrightarrow{d} P_1 \Rightarrow \exists P_1' : P_1 \xrightarrow{a} P_1'$$

Si une algèbre ne comporte ni choix faible, ni opérateur de chien de garde de quelque nature qu'il soit, alors tous les modèles possèdent la propriété de persistance. C'est le cas pour TCSP, TiCCS et U-LOTOS.

Cette propriété semble être une exigence trop forte dans le cadre des systèmes temps réel, où le temps est justement utilisé comme un paramètre servant à contrôler la possibilité d'exécution des actions. La persistance n'est pas présente dans les modèles de TPCCS, TPL, TeCCS ou ACP_ρ.

La propriété de persistance partielle (voir § 4.2.4.2) est assurée dans TPL et TPCCS, puisque le domaine temporel est discret. Elle le serait également en modifiant ces algèbres pour autoriser un domaine dense, comme cela est fait entre ATP et ATP_D.

En revanche, la présence conjointe du choix faible et des actions immédiates dans ACP_ρ et TeCCS ne garantit pas la persistance partielle :

- Dans TeCCS, le processus $(2)a\,P \oplus (3)b\,Q$ ne peut rien faire avant l'instant 2, ni entre les instants 2 et 3 ; il ne peut qu'exécuter a ou b, respectivement aux instants 2 et 3 exactement.
- Le terme de ACP_ρ ayant le même comportement s'écrit $a(2) \cdot P + b(3) \cdot Q$.

La présence ou l'absence de persistance partielle ne sont pas très importantes si on considère l'expressivité ou le réalisme des descriptions. En effet, les systèmes sont destinés à être exécutés sur des machines pour lesquelles la mesure du temps est nécessairement discrète. L'autorisation ponctuelle d'exécuter une action se traduit alors par une possibilité d'exécution pendant une durée égale à la base de temps.

4.3.3.5 Systèmes bien temporisés

Comme pour ATP_D, les processus de toutes les algèbres sauf TCSP ne sont pas nécessairement fortement bien temporisés (voir section 2.5, page 51). Ceux de TCSP le sont, si on ne considère que les termes réguliers (sans récursion à travers la composition parallèle). Cela est garanti par

l'introduction du délai minimal δ entre deux occurrences d'actions dans un terme séquentiel. En fait, dans un système comportant n processus en parallèle, le nombre maximal d'actions exécutées dans un temps d est $\frac{nd}{\delta}$.

Dans TPL, TiCCS et TPCCS, puisque les actions visibles sont toujours retardables, tout processus sans boucle d'actions internes est nécessairement bien temporisé (à variabilité finie) : il est toujours possible d'atteindre un état à partir duquel le temps peut passer. Cette quasi-certitude d'être bien temporisé est acquise aux dépens de la possibilité d'exprimer l'urgence, ce qui nous paraît plus important dans le cadre des systèmes temps réel, d'autant plus qu'il est toujours possible de vérifier la variabilité finie d'un processus (voir sections 2.5 et 6.4.4).

4.3.4 Conclusion

Du point de vue du pouvoir d'expression, l'algèbre TeCCS est certainement celle qui se rapproche le plus d'ATP_D.

Les algèbres TPL et TPCCS sont très comparables en ce qui concerne les aspects temporels : elles n'offrent pas la possibilité d'exprimer l'urgence, mais la persistance des actions n'est pas exigée.

U-LOTOS permet l'expression de l'urgence, mais pas la persistance. Son intérêt réside principalement dans l'introduction de l'opérateur `asap`, qui est d'une grande utilité (cf. § 3.2.1).

TiCCS et TCSP imposent la persistance des actions et n'offrent pas l'urgence. TiCCS présente l'intérêt d'offrir un théorème d'expansion de la composition parallèle, grâce aux variables temporelles liées aux actions. Il n'est cependant pas évident que la méthode employée soit adaptable à d'autres opérateurs et dans une algèbre où la persistance n'est pas requise. Nous considérons que l'inconvénient essentiel de TCSP est l'exigence du délai minimal entre les actions, qui peut être supprimée sans problème si on en donne une sémantique opérationnelle.

Enfin, ACP_ρ possède un grand pouvoir d'expression, en particulier lorsque l'intégration est utilisée dans toute sa généralité. Cependant, même dans les cas particuliers, la sémantique est d'une grande complexité et le nombre d'axiomes et d'opérateurs intermédiaires nécessaires est impressionnant (voir [Klu91]). De plus, la théorie est essentiellement présentée en utilisant un temps absolu pour étiqueter les actions ; les développements semblent encore plus complexes si on considère le temps relatif. Celui-ci est pourtant plus agréable à manipuler, surtout pour des processus dont le comportement est infini.

Partie II

Vérification

Chapitre 5

Les graphes temporisés

Nous avons défini dans le chapitre précédent les algèbres ATP_D, dont la syntaxe est identique quel que soit le domaine temporel, celui-ci paramétrant la sémantique. Le comportement d'un processus est décrit par son système de transitions, qui n'est autre qu'un *modèle d'exécution* du processus. Les transitions représentent, soit les actions exécutables et leurs conséquences sur le comportement, soit l'effet du passage du temps.

Lorsqu'on implante un système, le choix de la granularité g permet d'obtenir un modèle fini, ayant la structure des systèmes de transitions définis par la sémantique des ATP_g. On peut considérer ce modèle comme une forme de code exécutable par la machine (un automate).

Le problème essentiel lié à cet automate est sa taille, qui peut être très grande pour un système réel, et ce pour deux raisons :

- Le nombre d'états du modèle de la composition parallèle de deux processus est de l'ordre du produit des nombres d'états des modèles de chacune des composantes. Cette explosion combinatoire n'est pas liée à l'introduction du temps, elle existe aussi pour les systèmes non temporisés. En fait, les transitions temporelles ont même tendance à réduire l'explosion, puisque elles imposent une synchronisation globale de l'ensemble des composantes, réduisant ainsi le degré d'asynchronisme.
- La seconde raison est par contre intrinsèque aux systèmes temporisés : chaque chien de garde introduit dans une description provoque une explosion locale du nombre d'états (limitée à la portée de l'opérateur), fonction de la valeur du délai. Il est en effet nécessaire de connaître la durée écoulée depuis l'activation de l'opérateur pour déterminer à quel instant le délai expire. Cette mémorisation multiplie le nombre d'états "contrôlés" par $\frac{d}{g}$, si d est la valeur du délai. En conséquence, la taille du modèle croît avec les valeurs des délais et la finesse de la granularité.

Afin de remédier à l'explosion combinatoire due aux opérateurs temporels, nous proposons un modèle d'exécution plus abstrait que les systèmes de transitions de la sémantique d'ATP_D : les *graphes temporisés*.

Un graphe temporisé est un graphe (composé de nœuds et d'arcs orientés) auquel est associé un ensemble de compteurs, appelés *horloges*. Ce sont des variables prenant leurs valeurs dans un domaine temporel, et qui croissent uniformément avec le passage du temps. Nous ne considérons ici que les domaines temporels D_g et $\mathbb{Q}_+$. Les arcs sont étiquetés par une action, une condition sur les valeurs des horloges et une fonction leur assignant de nouvelles valeurs.

Intuitivement, l'"exécution" d'un graphe temporisé consiste à séjourner dans un nœud pendant un certain temps, et à franchir un arc dont la condition est vérifiée par les valeurs des horloges. Le franchissement est instantané ; les valeurs des horloges peuvent être modifiées (par exemple remises à zéro) par la fonction associée à l'arc.

À chaque nœud est de plus associée une *condition d'activité* portant sur les valeurs des horloges, servant essentiellement à définir la durée maximale de séjour dans un nœud.

À l'aide des graphes temporisés, on peut modéliser des systèmes temps réel comportant des constructions temporelles très variées. Leur principal attrait est que la taille du graphe est indépendante de la granularité et des valeurs des délais. La seconde cause d'explosion combinatoire est ainsi éliminée. Cela permet de plus de les définir indépendamment du domaine temporel. Ils fournissent un modèle d'exécution implantable sur une machine. On peut d'ailleurs remarquer que la traduction en code OC [CPPS90] de l'instruction `watching` n `S` d'ESTEREL fait appel à un principe similaire afin d'éviter l'explosion du nombre d'états : un compteur initialisé à n est décrémenté à chaque occurrence du signal `S` ; son passage à zéro déclenche une transition d'expiration du chien de garde [Cou90].

Le formalisme présenté ici est fortement inspiré des *timed graphs* [ACD90] et des *timed automata* [AD90, AFH91]. Un autre type de modèles est présenté dans [HMP91] : les *timed transition systems*. Contrairement aux précédents, la notion d'horloge n'est dans ce dernier cas pas explicite. Un intervalle de temps $[l, u]$ est associé à chaque transition, qui ne peut être franchie que si elle est possible pendant une durée l, et qui doit alors être franchie dans l'intervalle $[l, u]$.

Outre la réduction du nombre des états et l'insensibilité au changement du domaine temporel, les graphes temporisés présentent l'avantage de se prêter à la vérification de propriétés de logique temporelle par model checking. Un algorithme de vérification est présenté dans [ACD90] ; nous en présentons un autre au chapitre 6, qui fait appel à des manipulations symboliques de prédicats.

Nous présentons ici la définition générale des graphes temporisés, et nous en donnons la sémantique dans le même cadre que celle d'ATP$_D$. Nous expliquons ensuite comment traduire un terme d'ATP$_D$ en un graphe temporisé, de manière à en préserver la sémantique. Ces résultats sont présentés dans [NSY91]. Une méthode effective de traduction a été mise au point et implantée par S. Yovine [NSY92].

Au lieu d'utiliser ATP$_D$, il peut parfois être agréable de définir les composantes d'un système temps réel directement sous forme de graphes temporisés. La méthode de traduction présentée est utilisable pour composer des graphes temporisés quelconques au moyen des opérateurs d'ATP$_D$, en particulier la composition parallèle et le chien de garde. Il suffit que les graphes aient des ensembles d'horloges disjoints.

5.1 Définitions, sémantique et exemples

5.1.1 Horloges, conditions, graphes temporisés

Les horloges sont notées x, y, z, x_1, x', ...

Soit $\mathcal{H}$ un p-uplet $(x_1, x_2, \ldots, x_p)$ d'horloges.

Une *valuation* $\vec{v}$ de $\mathcal{H}$ dans le domaine temporel D (soit un D_g, soit $\mathbb{Q}_+$) est un p-uplet $(v_1, v_2, \ldots, v_p)$ dont les composantes sont dans D. La valuation $(d, d, \ldots, d)$ est notée $\vec{d}$.

L'ensemble $\mathcal{C}(\mathcal{H})$ des *conditions* sur les horloges de $\mathcal{H}$ est défini par la syntaxe suivante :

$$c ::= x_i \leq k \quad | \quad x_i < k \quad | \quad x_i + n \leq x_j + m \quad | \quad \neg c \quad | \quad c \wedge c$$

où k, n et m appartiennent à $\mathbb{N}$, et n ou m sont nuls.

Une condition est interprétée comme un prédicat sur l'ensemble des valuations rationnelles (une application de $\mathbb{Q}_+^p$ dans $\{\mathtt{tt}, \mathtt{ff}\}$) :

$$\begin{aligned}
(x_i \leq k)(\vec{v}) = \mathtt{tt} \quad &\text{ssi} \quad v_i \leq k \\
(x_i < k)(\vec{v}) = \mathtt{tt} \quad &\text{ssi} \quad v_i < k \\
(x_i + n \leq x_j + m)(\vec{v}) = \mathtt{tt} \quad &\text{ssi} \quad v_i + n \leq v_j + m \\
(\neg c)(\vec{v}) = \mathtt{tt} \quad &\text{ssi} \quad c(\vec{v}) = \mathtt{ff} \\
(c \wedge c')(\vec{v}) = \mathtt{tt} \quad &\text{ssi} \quad c(\vec{v}) = \mathtt{tt} \text{ et } c'(\vec{v}) = \mathtt{tt}
\end{aligned}$$

Remarquer que des conditions telles que $\mathtt{true}$, $c \Rightarrow c'$, $(x_i > 5) \vee (x_j < 3)$ ou $2 < x_i < 5$ peuvent être définies comme des abréviations.

$\mathcal{F}(\mathcal{H})$ est l'ensemble des applications de $\mathbb{Q}_+^p$ dans $\mathbb{Q}_+^p$ telles que pour toute granularité g, pour tout f dans $\mathcal{F}(\mathcal{H})$, $f(D_g^p) \subseteq D_g^p$.

Soit ε un élément n'appartenant pas à $\mathcal{A}$. $\mathcal{A}_\varepsilon$ désigne l'ensemble $\mathcal{A} \cup \{\varepsilon\}$, dont les éléments sont notés α, β, ... L'élément ε n'est pas une action. Il est utilisé pour dénoter l'expiration d'un délai.

Un *graphe temporisé* est un quintuplet

$$(\mathcal{N}, \mathcal{H}, N_0, \mathtt{act}, \rightarrow)$$

où

- $\mathcal{N}$ est un ensemble fini de *nœuds* ;
- $\mathcal{H}$ est un ensemble fini d'*horloges* ;
- $N_0 \in \mathcal{N}$ est le *nœud initial* ;
- $\mathtt{act}$ est une application de $\mathcal{N}$ dans $\mathcal{C}(\mathcal{H})$, appelée *fonction d'activité*. $\mathtt{act}(N)$ est la *condition d'activité* du nœud N ;
- $\rightarrow \subseteq \mathcal{N} \times \mathcal{A}_\varepsilon \times \mathcal{C}(\mathcal{H}) \times \mathcal{F}(\mathcal{H}) \times \mathcal{N}$ est un ensemble fini d'*arcs*. On écrit $N \xrightarrow{\alpha,c,f} N'$ au lieu de $(N, \alpha, c, f, N') \in \rightarrow$; N est la *source*, N' le *but* et (α, c, f) l'*étiquette* de cet arc ; α en est l'action, c la condition de franchissement, et f la fonction d'évolution des horloges.

5.1.2 Sémantique opérationnelle

Une fois choisi un domaine temporel, un graphe temporisé définit un modèle d'exécution fonctionnant de la manière suivante.

Initialement, la valeur initiale des horloges est 0. Un état du contrôle est représenté par un nœud N et une valuation $\vec{v}$ dans D des horloges.

À partir du nœud N, si les valeurs des horloges satisfont à la fois $\mathtt{act}(N)$ et la condition c d'un arc $N \xrightarrow{a,c,f} N'$, alors la transition correspondante *peut* être effectuée : l'action a est exécutée, et l'état du contrôle devient $(N', f(\vec{v}))$.

Lorsque $\texttt{act}(N)$ et la condition de l'arc $N \xrightarrow{\varepsilon,c,f} N'$ sont satisfaites, il se produit un changement instantanée de l'état du contrôle, qui devient $(N', f(\vec{v}))$

Il est possible de "stationner" dans le nœud N tant que la condition $\texttt{act}(N)$ est vérifiée par les valeurs des horloges. Si elle ne peut plus être satisfaite plus longtemps, il est impératif d'exécuter une transition ; si aucune n'est possible, on se trouve dans une situation de blocage. C'est également le cas si une transition mène dans un état $(N, \vec{v})$ tel que $\texttt{act}(N)(\vec{v})$ est fausse ; on a alors un blocage dès l'entrée dans le nœud N.

Formellement, la sémantique opérationnelle des graphes temporisés leur associe un système de transitions étiquetées ayant la même structure que ceux des processus d'ATP_D. Elle est paramétrée par le domaine temporel D, choisi parmi les D_G et $\mathbb{Q}_+$.

Pour un graphe temporisé $(\mathcal{N}, \mathcal{H}, N_0, \texttt{act}, \rightarrow)$, où le cardinal de $\mathcal{H}$ est p, la relation de transition $\xrightarrow[D]{}$ est définie sur

$$(\mathcal{N} \times D^p) \times (\mathcal{A} \cup D^*) \times (\mathcal{N} \times D^p)$$

L'état initial du système de transitions est $(N_0, \vec{0})$. La relation $\xrightarrow[D]{}$ est définie en fonction des arcs et par induction structurelle par les règles suivantes :

$$\frac{\forall d' \leq d : \texttt{act}(N)(\vec{v} + \vec{d'}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{d} (N, \vec{v} + \vec{d})} \qquad \frac{N \xrightarrow{a,c,f} N' \ , \ (\texttt{act}(N) \wedge c)(\vec{v}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{a} (N', f(\vec{v}))}$$

$$\frac{N \xrightarrow{\varepsilon,c,f} N' \ , \ (N', f(\vec{v})) \xrightarrow[D]{a} (N'', \vec{v'}) \ , \ (\texttt{act}(N) \wedge c)(\vec{v}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{a} (N'', \vec{v'})}$$

$$\frac{N \xrightarrow{\varepsilon,c,f} N' \ , \ (N', f(\vec{v})) \xrightarrow[D]{d} (N'', \vec{v'}) \ , \ (\texttt{act}(N) \wedge c)(\vec{v}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{d} (N'', \vec{v'})}$$

$$\frac{(N, \vec{v}) \xrightarrow[D]{d} (N', \vec{v'}) \ , \ N' \xrightarrow{\varepsilon,c,f} N'' \ , \ (N'', f(\vec{v'})) \xrightarrow[D]{d'} (N''', \vec{v''}) \ , \ (\texttt{act}(N') \wedge c)(\vec{v'}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{d+d'} (N''', \vec{v''})}$$

La première règle définit le passage du temps dans un nœud : l'activité du nœud doit être continuellement vraie pour laisser passer le temps en y séjournant. Les quatre autres règles affirment qu'un arc ne peut être franchi que si sa condition de franchissement et celle d'activité du nœud sont toutes les deux satisfaites. De plus, les trois dernières règles illustrent le fait que le franchissement d'un arc étiqueté par ε est totalement invisible.

Nous prouvons dans l'annexe C que cette sémantique garantit l'additivité des transitions temporelles (cf. prop. 4.2.3, p. 78), et que la dernière règle peut être remplacée par

$$\frac{(N, \vec{v}) \xrightarrow[D]{d} (N', \vec{v'}) \ , \ (N', \vec{v'}) \xrightarrow[D]{d'} (N''', \vec{v''})}{(N, \vec{v}) \xrightarrow[D]{d+d'} (N''', \vec{v''})}$$

En revanche, le déterminisme temporel n'est pas assuré. Par exemple, si les deux nœuds N et N' ont comme condition d'activité $\texttt{true}$, et s'il existe un arc $N \xrightarrow{\varepsilon,x=0,id} N'$, alors depuis l'état $(N, \vec{0})$, on obtient des transitions $(N, \vec{0}) \xrightarrow[D]{d} (N, \vec{d})$ et $(N, \vec{0}) \xrightarrow[D]{d} (N', \vec{d})$ pour toute valeur d. Si on souhaite préserver le déterminisme, alors il est nécessaire de faire apparaître des transitions étiquetées par ε : on remplace les trois dernières règles par

$$\frac{N \xrightarrow{\varepsilon,c,f} N' \ , \ (\texttt{act}(N) \wedge c)(\vec{v}) = \texttt{tt}}{(N, \vec{v}) \xrightarrow[D]{\varepsilon} (N', f(\vec{v}))}$$

qui est plus facile à utiliser pour comprendre le "comportement" d'un graphe temporisé. Il est cependant nécessaire de considérer les cinq règles ci-dessus pour les modèles des graphes obtenus à partir de termes d'ATP_D, afin de prouver la correction de la traduction.

5.1.3 Exemples de description

Nous donnons ici la description directe en termes de graphes temporisés de quelques exemples présentés dans le chapitre 3.

Pour dessiner les graphes temporisés, nous adoptons les conventions suivantes :

- Le nœud initial est entouré d'un trait épais.
- La condition d'activité d'un nœud est inscrite à l'intérieur de celui-ci.
- Lorsque la condition de franchissement d'un arc est `true`, elle est omise ; il en va de même lorsque la fonction d'évolution est l'identité.
- La fonction d'évolution est représentée par une liste d'assignations simultanées de valeurs aux horloges. La valeur attribuée à une horloge peut dépendre de celle d'une autre horloge.

5.1.3.1 L'émetteur du protocole du bit alterné

Nous rappelons la spécification de l'émetteur : lorsqu'il reçoit une demande d'émission in, il transmet un message à la ligne L (action el), puis attend l'accusé de réception le qui doit arriver dans un délai de. Si c'est le cas, il passe au message suivant, sinon il retransmet le message. Un bit de contrôle 0 ou 1 est attaché aux messages et aux accusés de réception. Si le bit de l'accusé de réception ne correspond pas à celui du message transmis, l'émetteur renvoie le message.

On peut traduire directement cette spécification en un graphe temporisé comportant une seule horloge x (voir figure 5.1).

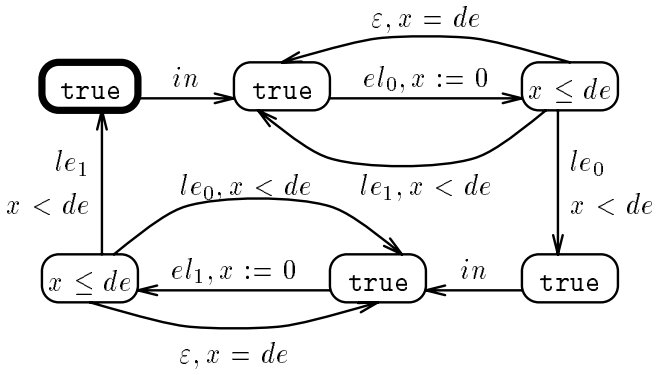


Figure 5.1 : graphe temporisé de l'émetteur

L'extension proposée pour l'émetteur consiste à imposer qu'une phase d'émission (après in et jusqu'à réception de l'accusé de réception correct) doit être complétée dans un délai da ; dans le cas contraire, le contrôle passe à la phase de déconnexion D.

Pour modéliser ce comportement, on utilise une seconde horloge y, mise à zéro lors de l'exécution de l'action in ; le chien de garde expire lorsque $y = da$. Le graphe temporisé correspondant est présenté sur la figure 5.2.

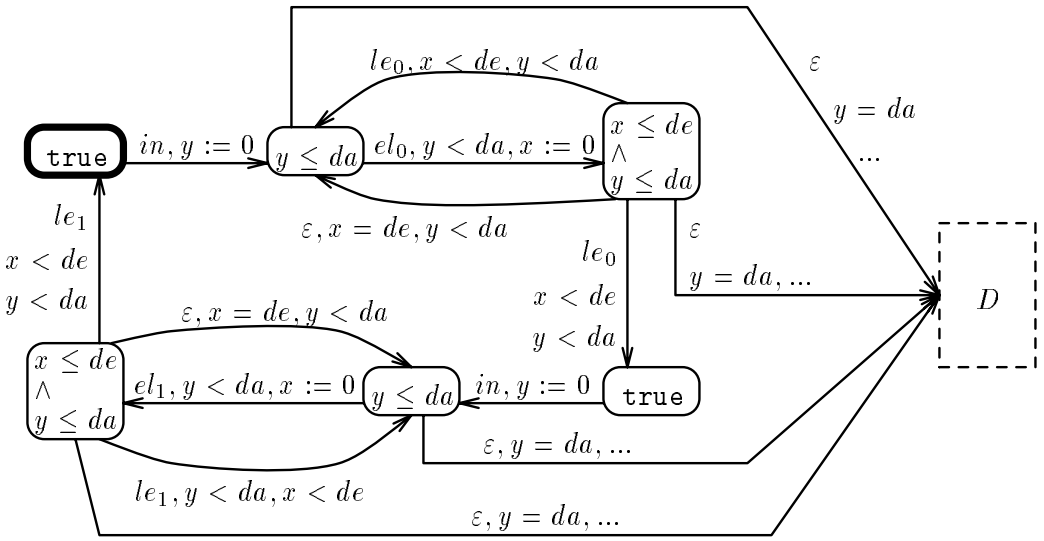


Figure 5.2 : l'émetteur étendu

5.1.3.2 La procédure de connexion

La spécification est la suivante : le processus L affiche immédiatement un message m, et attend une réponse valide v ou invalide i pendant une durée dr. Si la réponse est invalide ou s'il n'y en a pas avant dr, L recommence son exécution. Si la réponse est valide, le contrôle est transmis à la phase de session S. Le graphe temporisé correspondant à L est présenté sur la figure 5.3 ; une seule horloge x est nécessaire.

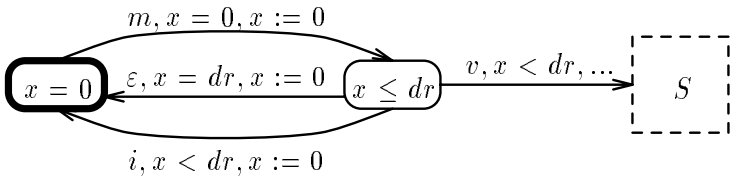


Figure 5.3 : procédure de connexion, graphe temporisé de L

La procédure complète consiste à contrôler L de telle sorte que si une réponse valide n'est pas donnée dans une durée dp, L est annulé et une exception E est lancée. Pour ce contrôle, on utilise une seconde horloge y ; on obtient le graphe temporisé de la figure 5.4.

5.1.3.3 Le contrôleur de passage à niveau

Le système est composé du train, de la barrière et du contrôleur.

Après l'envoi par le train du signal d'approche a_t, il se passe au moins trois unités de temps avant l'entrée dans la zone du passage à niveau (i). Le signal de sortie s_t est envoyé après la sortie effective de la zone (o). De plus, il se passe moins de 6 unités de temps entre a_t et s_t. Pour le graphe temporisé du train, on utilise une horloge x pour spécifier la durée maximale (< 6) de séjour dans les nœuds entre l'approche et la sortie, et pour assurer que 3 unités de temps au

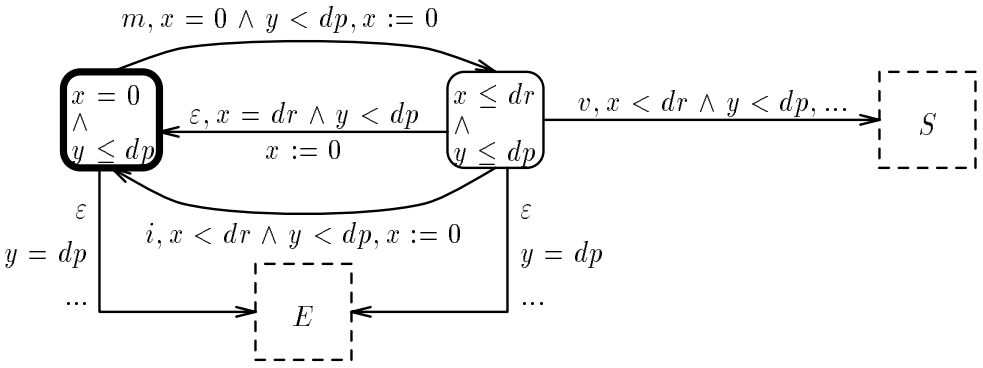
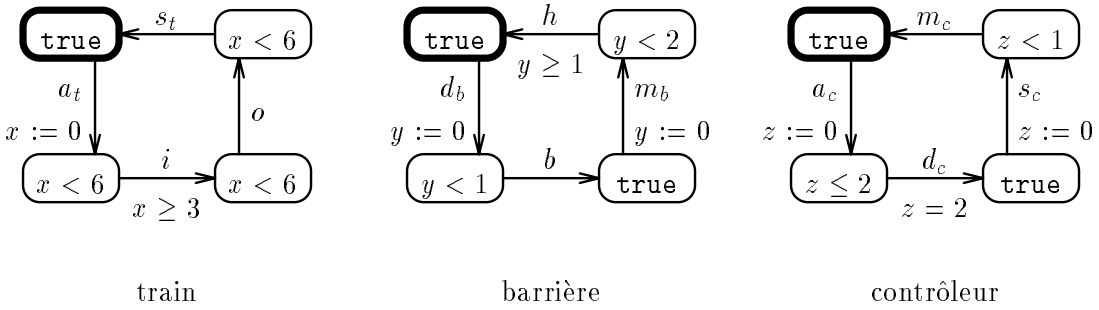


Figure 5.4 : procédure de connexion, graphe temporisé complet

moins s'écoulent entre a_t et i (voir figure 5.5).



train barrière contrôleur

Figure 5.5 : système de contrôle de passage à niveau

Lorsque la barrière reçoit le signal de descente d_b, il se passe moins d'une unité de temps avant qu'elle arrive en bas (action b). Ensuite, une fois reçu le signal de montée m_b, il faut au moins une et moins de deux unités de temps avant que la barrière arrive en haut (h). Le graphe temporisé comporte une horloge y pour contrôler les temps de descente et de montée (voir figure 5.5).

Le contrôleur envoie l'ordre d'abaissement d_c exactement deux unités de temps après réception du signal d'approche a_c ; l'ordre de relèvement m_c est émis moins d'une unité de temps après réception du signal de sortie s_c. À nouveau, une horloge suffit pour temporiser les ordres envoyés à la barrière, comme on le constate sur la figure 5.5.

5.2 Traduction d'ATP$_D$

Nous présentons dans cette section une méthode de traduction d'un processus d'ATP$_D$ en un graphe temporisé. Cette méthode garantit l'équivalence des modèles du terme et du graphe : les deux modèles sont fortement équivalents.

5.2.1 Restrictions et définitions préliminaires

Afin d'obtenir un graphe temporisé fini, nous imposons des restrictions sur les termes d'ATP$_D$ que nous considérons : une récursion est interdite

- à travers la composition parallèle, l'urgence et l'encapsulation ;
- à travers l'opérateur $\Downarrow_H^d$ (resp. le premier argument de l'opérateur) $\lceil\,\rceil_H^d$, si elle n'est pas gardée par un élément de H dans l'argument (resp. le premier argument) de l'opérateur ;
- à travers l'opérateur $+$, si elle n'est pas gardée par une action dans les arguments du choix non déterministe. Nous interdisons par exemple le terme $\mathtt{rec}X \cdot a\,(X + P)$.

Une variable ne peut donc apparaître libre dans un choix non déterministe que si elle est argument d'un préfixage ($a\,X$), ou second argument d'un chien de garde ($\lceil P\rceil_H^d X$). Dans ce dernier cas, le chien de garde doit lui-même être argument d'un préfixage ou second argument d'un autre chien de garde qui satisfait les mêmes conditions.

Soit un processus P_0 d'ATP$_D$ obéissant à ces contraintes. Le principe de la traduction de P_0 en un graphe temporisé est le suivant.

Les nœuds du graphe sont des termes d'ATP$_D$. En particulier tous les délais sont des entiers strictement positifs. Nous les notons n au lieu de d.

À des termes différentes correspondent des nœuds différents.

Les horloges du graphe servent à définir les contraintes temporelles. Nous en utilisons une pour chaque occurrence d'un chien de garde ou d'une restriction temporelle dans P_0. Nous associons à chacune de ces occurrences un indice, que nous faisons apparaître dans les termes : nous écrivons ${}_j\lceil P\rceil_H^n Q$ au lieu de $\lceil P\rceil_H^n Q$, et $P_j\Downarrow_H^n$ au lieu de $P\Downarrow_H^n$. L'horloge correspondant à l'occurrence d'indice j est notée x_j.

Nous utilisons une horloge supplémentaire, notée u, pour forcer l'urgence des actions le cas échéant.

Le nœud initial est P_0.

La fonction d'évolution des arcs consiste toujours à mettre à zéro certaines horloges. Nous la notons par l'ensemble de ces horloges.

Dans le but de simplifier la traduction pour l'opérateur d'urgence, nous définissons l'ensemble $I(P) \subseteq \mathcal{A}$ des actions initiales (immédiatement exécutables) d'un terme P :

$$
\begin{array}{ccc}
I(\delta) \stackrel{\text{def}}{=} \emptyset & I(\mathbf{0}) \stackrel{\text{def}}{=} \emptyset & I(\mathtt{rec}X \cdot P) \stackrel{\text{def}}{=} I(P[\mathtt{rec}X \cdot P/X]) \\
I(a\,P) \stackrel{\text{def}}{=} \{a\} & I(\widetilde{a}\,P) \stackrel{\text{def}}{=} \{a\} & I(P+Q) \stackrel{\text{def}}{=} I(P) \cup I(Q) \\
I({}_j\lceil P\rceil_H^n Q) \stackrel{\text{def}}{=} I(P) & I(\mathtt{U}_H(P)) \stackrel{\text{def}}{=} I(P) & I(P_j\Downarrow_H^n) \stackrel{\text{def}}{=} I(P) \\
I(P\|Q) \stackrel{\text{def}}{=} I(P) \cup I(Q) \cup (I(P)|I(Q)) & & I(\partial_H(P)) \stackrel{\text{def}}{=} I(P) - H
\end{array}
$$

où $I(P)|I(Q) \stackrel{\text{def}}{=} \{b \in \mathcal{A} \mid \exists a \in I(P), \exists a' \in I(Q) : a|a' = b \neq \perp\}$. Cette définition par induction est valide, car les termes d'ATP$_D$ sont bien gardés.

On constate aisément que $I(P) = I(Q)$ si $P \sim Q$.

Nous définissons finalement l'ensemble $h(P)$ des *horloges initiales* d'un nœud dénoté par terme P, en fonction de la structure de P :

$$h(\delta) \stackrel{\text{def}}{=} \emptyset \qquad h(\mathbf{0}) \stackrel{\text{def}}{=} \{u\} \qquad h(P+Q) \stackrel{\text{def}}{=} h(P) \cup h(Q)$$
$$h(a\,P) \stackrel{\text{def}}{=} \{u\} \qquad h(\widetilde{a}\,P) \stackrel{\text{def}}{=} \emptyset \qquad h(P\|Q) \stackrel{\text{def}}{=} h(P) \cup h(Q)$$
$$h(\partial_H(P)) \stackrel{\text{def}}{=} h(P) \qquad h({}_j\lceil P\rceil^n_H Q) \stackrel{\text{def}}{=} h(P) \cup \{x_j\} \qquad h(\mathtt{rec}X \cdot P) \stackrel{\text{def}}{=} h(P[\mathtt{rec}X \cdot P/X])$$
$$h(\mathtt{U}_H(P)) \stackrel{\text{def}}{=} \text{si } I(P) \cap H = \emptyset \text{ alors } h(P) \text{ sinon } h(P) \cup \{u\} \qquad h(P_j\Downarrow^n_H) \stackrel{\text{def}}{=} h(P) \cup \{x_j\}$$

Comme pour les ensembles d'actions initiales, cette définition par induction est valide.

5.2.2 Traduction

5.2.2.1 Fonction d'activité

La condition $\mathtt{act}(P)$ spécifie les valeurs des horloges pour lesquelles on peut stationner dans le nœud P. Elle est également définie par induction :

$$\mathtt{act}(\delta) \stackrel{\text{def}}{=} \mathtt{true} \qquad \mathtt{act}(P+Q) \stackrel{\text{def}}{=} \mathtt{act}(P) \wedge \mathtt{act}(Q)$$
$$\mathtt{act}(\mathbf{0}) \stackrel{\text{def}}{=} (u=0) \qquad \mathtt{act}(P\|Q) \stackrel{\text{def}}{=} \mathtt{act}(P) \wedge \mathtt{act}(Q)$$
$$\mathtt{act}(a\,P) \stackrel{\text{def}}{=} (u=0) \qquad \mathtt{act}({}_j\lceil P\rceil^n_H Q) \stackrel{\text{def}}{=} \mathtt{act}(P) \wedge (x_j \le n)$$
$$\mathtt{act}(\widetilde{a}\,P) \stackrel{\text{def}}{=} \mathtt{true} \qquad \mathtt{act}(P_j\Downarrow^n_H) \stackrel{\text{def}}{=} \mathtt{act}(P) \wedge (x_j < n)$$
$$\mathtt{act}(\partial_H(P)) \stackrel{\text{def}}{=} \mathtt{act}(P) \qquad \mathtt{act}(\mathtt{rec}X \cdot P) \stackrel{\text{def}}{=} \mathtt{act}(P[\mathtt{rec}X \cdot P/X])$$
$$\mathtt{act}(\mathtt{U}_H(P)) \stackrel{\text{def}}{=} \text{si } I(P) \cap H = \emptyset \text{ alors } \mathtt{act}(P) \text{ sinon } (u=0)$$

5.2.2.2 Arcs

Nous définissons à présent les arcs reliant les nœuds. La fonction `act` permet d'associer aux arcs des conditions de franchissement relativement simples. Les fonctions d'évolution sont des ensembles d'horloges remises à zéro ; nous les notons h, h'.

Les arcs issus d'un nœud P sont définis par induction sur la structure de P par les règles suivantes.

$$a\,P \xrightarrow{a,\mathtt{true},h(P)} P \qquad \widetilde{a}\,P \xrightarrow{a,\mathtt{true},h(P)} P \qquad \frac{P[\mathtt{rec}X \cdot P/X] \xrightarrow{\alpha,c,h} P'}{\mathtt{rec}X \cdot P \xrightarrow{\alpha,c,h} P'}$$

$$\frac{P \xrightarrow{a,c,h} P'}{P+Q \xrightarrow{a,c,h} P'} \qquad \frac{Q \xrightarrow{a,c,h} Q'}{P+Q \xrightarrow{a,c,h} Q'} \qquad \frac{P \xrightarrow{\varepsilon,c,h} P'}{P+Q \xrightarrow{\varepsilon,c,h} P'+Q} \qquad \frac{Q \xrightarrow{\varepsilon,c,h} Q'}{P+Q \xrightarrow{\varepsilon,c,h} P+Q'}$$

$$\frac{P \xrightarrow{\alpha,c,h} P'}{P\|Q \xrightarrow{\alpha,c,h} P'\|Q} \qquad \frac{Q \xrightarrow{\alpha,c,h} Q'}{P\|Q \xrightarrow{\alpha,c,h} P\|Q'} \qquad \frac{P \xrightarrow{a,c,h} P', Q \xrightarrow{a',c',h'} Q', a|a' =\neq \bot}{P\|Q \xrightarrow{a|a',c\wedge c',h\cup h'} P'\|Q'}$$

$$\frac{P \xrightarrow{\alpha,c,h} P', \alpha \notin H}{\partial_H(P) \xrightarrow{\alpha,c,h} \partial_H(P')} \qquad \frac{P \xrightarrow{\alpha,c,h} P'}{\mathtt{U}_H(P) \xrightarrow{\alpha,c,h} \mathtt{U}_H(P')} \qquad \frac{P \xrightarrow{\alpha,c,h} P', \alpha \notin H}{P_j\Downarrow^n_H \xrightarrow{\alpha,c,h} P'_j\Downarrow^n_H} \qquad \frac{P \xrightarrow{a,c,h} P', a \in H}{P_j\Downarrow^n_H \xrightarrow{a,c,h} P'}$$

$$\frac{P \xrightarrow{\alpha,c,h} P', \alpha \notin H}{{}_j\lceil P\rceil^n_H Q \xrightarrow{\alpha,c\wedge(x_j<n),h} {}_j\lceil P'\rceil^n_H Q} \qquad \frac{P \xrightarrow{a,c,h} P', a \in H}{{}_j\lceil P\rceil^n_H Q \xrightarrow{a,c\wedge(x_j<n),h} P'} \qquad {}_j\lceil P\rceil^n_H Q \xrightarrow{\varepsilon,(x_j=n),h(Q)} Q$$

Ces règles appellent quelques commentaires :

- Aucun arc n'est issu des nœuds δ et $\mathbf{0}$. Ils ne diffèrent que par leur condition d'activité, qui autorise à stationner indéfiniment dans le nœud δ, et qui bloque le temps dans le nœud $\mathbf{0}$.
- La condition de l'arc du préfixage immédiat peut également être $u = 0$, ce qui est redondant avec $\mathtt{act}(a\,P)$.
- Aucune synchronisation temporelle n'est calculée pour le choix non déterministe et la composition parallèle, parce qu'il n'est pas possible de les calculer.

  Considérons par exemple le nœud ${}_j\lceil P\rceil^n_H Q + {}_{j'}\lceil P'\rceil^n_{H'} Q'$. Les deux chiens de garde expirent lorsque $x_j = n$ et $x_{j'} = n$. Cela n'implique en aucune façon qu'ils expirent simultanément, puisque les instants où x_j et $x_{j'}$ ont été remises à zéro ne coïncident pas nécessairement ; de plus ces instants sont inconnus dans le nœud concerné.

  Une conséquence de cette remarque est que si les graphes temporisés suppriment l'explosion du nombre d'états due aux valeurs des délais, en revanche, ils font disparaître l'avantage apporté par la synchronisation forte des processus sur les transitions temporelles. En fait, la taille du graphe temporisée est de l'ordre de celle qu'on obtient pour les modèles de processus décrits dans une algèbre non temporisée, où les expirations de délais sont modélisés par des transitions internes.

  La synchronisation temporelle est uniquement spécifiée par la contrainte de progression uniforme des valeurs des horloges, qui intervient au moment de la construction du modèle du graphe temporisé. Elle est également prise en compte pour la vérification des propriétés (voir le chapitre suivant).
- Dans le cas de l'opérateur d'urgence, la règle est simplifiée par le calcul de la condition d'activité, qui détermine l'urgence des transitions en fonction des actions initiales de l'argument.
- Pour la restriction temporelle, la contrainte $x_j < n$ n'est pas nécessaire sur les arcs, puisqu'elle est déjà présente dans la condition d'activité.
- Un chien de garde expire lorsque son horloge associée est égale à la valeur du délai. Si le corps *doit* évoluer avant l'expiration, celle-ci ne peut avoir lieu dans ce nœud, dont l'activité est restreinte par la condition d'activité du corps.

On peut montrer que les modèles d'un terme d'ATP$_D$ et de sa traduction en graphe temporisé sont fortement équivalents. En particulier, les modèles des graphes temporisé ainsi obtenus sont déterministes pour les transitions temporelles, ce qui n'est pas nécessairement le cas pour un graphe quelconque, comme on l'a vu dans la section 5.1.2. Nous ne présentons pas la preuve de l'équivalence, qui est longue et fastidieuse mais sans réelle difficulté technique.

La nécessité d'interdire une récursion à travers le choix non déterministe si elle n'est pas gardée par une action dans les arguments de + est illustrée par l'exemple suivant.

Exemple 5.1

Soit le processus d'ATP$_D$ défini par

$$P \stackrel{\text{def}}{=} \mathtt{rec}X \cdot ({}_1\lceil\delta\rceil^1_{\mathcal{A}} X \;+\; {}_2\lceil\delta\rceil^2_{\mathcal{A}} a\,\delta)$$

Les horloges initiales de P sont x_1 et x_2 et sa condition d'activité est $x_1 \leq 1 \wedge x_2 \leq 2$. Le problème se pose lors de l'expiration du premier chien de garde. D'après la sémantique d'ATP$_D$ on a une transition

$$P \xrightarrow{1}_D P \;+\; \lceil\delta\rceil^1_{\mathcal{A}} a\,\delta \;=\; \lceil\delta\rceil^1_{\mathcal{A}} P \;+\; \lceil\delta\rceil^2_{\mathcal{A}} a\,\delta \;+\; \lceil\delta\rceil^1_{\mathcal{A}} a\,\delta$$

On constate qu'il se produit une "duplication" du second chien de garde, *avec deux valeurs différentes du délai*, et que *ces deux instances sont "actives" simultanément*, c'est-à-dire qu'elles ne sont gardées ni l'une ni l'autre.

En suivant les règles des arcs pour les graphes temporisés, on obtient :

$$P \xrightarrow{\varepsilon, x_1=1, h(P)} P \;+\; {}_2\lceil\delta\rceil^2_{\mathcal{A}}\, a\,\delta \;=\; {}_1\lceil\delta\rceil^1_{\mathcal{A}}\, P \;+\; {}_2\lceil\delta\rceil^2_{\mathcal{A}}\, a\,\delta \;+\; {}_2\lceil\delta\rceil^2_{\mathcal{A}}\, a\,\delta$$

Il survient également une duplication du second chien de garde. Les horloges de P, dont x_2 fait partie, sont remises à zéro par la transition ; par conséquent, l'instance initiale du second chien de garde ne peut pas expirer une unité de temps après cette transition, ce qui ne correspond pas au comportement attendu.

Les deux instances actives du chien de garde d'indice 2 doivent en fait être différenciées : il faut associer l'indice 3 à la nouvelle instance, et remettre x_3 à zéro lors de la transition. Cette solution est inapplicable, puisque on obtient alors une infinité d'horloges et de nœuds : le prochain arc correspondant de nouveau à la condition $x_1 = 1$ crée une troisième instance du même chien de garde, à laquelle il faut associer une nouvelle horloge x_4...

Ce processus ne pose pas de problème dans le cadre d'ATP$_D$, car le modèle sous forme de système de transitions est calculé en tenant compte de la synchronisation forte des transitions temporelles, dans les composantes d'un choix non déterministe. Comme nous l'avons expliqué plus haut, ce calcul ne peut pas être effectué dans le cas général lors de la construction du graphe temporisé.

□

Interdire les récursions non gardées par une action à travers le choix non déterministe garantit que les duplications non gardées de processus ne peuvent se produire, ce qui permet de déterminer statiquement le nombre (fini) d'horloges et d'obtenir un nombre fini de nœuds.

5.2.3 Exemples

À titre d'exemples de traduction, nous considérons une fois encore l'émetteur du protocole du bit alterné, la procédure de connexion et le contrôleur de passage à niveau. Leur graphe temporisé est construit en suivant les règles ci-dessus à partir de leur spécification en ATP$_D$.

5.2.3.1 L'émetteur du protocole du bit alterné

Nous décrivons l'émetteur sous la forme du système d'équations suivant, en indexant les chiens de garde.

$$\begin{aligned}
P_0 &\stackrel{\text{def}}{=} \widetilde{in}\, P_1 \\
P_1 &\stackrel{\text{def}}{=} \widetilde{el_0}\, P_2 \\
P_2 &\stackrel{\text{def}}{=} {}_1\Big\lceil \widetilde{le_0}\, P_3 \;+\; \widetilde{le_1}\, P_1 \Big\rceil^{de}_{\mathcal{A}} P_1 \\
P_3 &\stackrel{\text{def}}{=} \widetilde{in}\, P_4 \\
P_4 &\stackrel{\text{def}}{=} \widetilde{el_1}\, P_5 \\
P_5 &\stackrel{\text{def}}{=} {}_2\Big\lceil \widetilde{le_0}\, P_4 \;+\; \widetilde{le_1}\, P_0 \Big\rceil^{de}_{\mathcal{A}} P_4
\end{aligned}$$

Il faut utiliser deux horloges x_1 et x_2 pour les chiens de garde. L'horloge u est inutile puisque aucune action n'est immédiate et il n'y a pas d'opérateur d'urgence.

Les équations sont écrites de telle sorte que le graphe temporisé comporte six nœuds, dénotés par chacun des P_i (ou des E_i correspondantes).

Les horloges initiales des nœuds sont les suivantes :

$$h(P_0) = h(P_1) = h(P_3) = h(P_4) = \emptyset \qquad h(P_2) = \{x_1\} \qquad h(P_5) = \{x_2\}$$

La figure 5.6 présente le graphe temporisé obtenu en utilisant les règles ci-dessus. On constate qu'il est presque identique à celui décrit au paragraphe 5.1.3.1 : la seule différence provient de l'utilisation de deux horloges, là où une seule suffit.

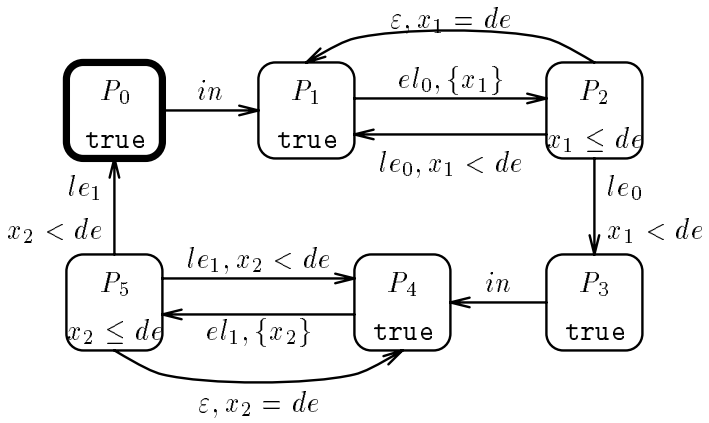


Figure 5.6 : traduction de l'émetteur

Si on souhaite minimiser le nombre des horloges, il est possible de parcourir le graphe temporisé afin de détecter les horloges "indépendantes". Si, dans le graphe, toute occurrence d'utilisation (dans une condition) de x_1 est séparée de toute occurrence d'utilisation de x_2 par une remisa à zéro de x_2, et vice-versa, alors on peut utiliser une unique horloge au lieu de deux. Il en va bien sûr de même si deux horloges sont toujours remises à zéro simultanément.

La traduction de l'extension de l'émetteur produit un graphe semblable à celui présenté au paragraphe 5.1.3.1, avec quatre horloges au lieu de deux, puisque le terme ATP_D comporte deux délais de commencement et deux chiens de garde.

5.2.3.2 La procédure de connexion

Contrairement au cas précédent, nous détaillons cette fois complètement la traduction de la procédure de connexion.

Nous écrivons sa définition en ATP_D de la manière suivante, en n'utilisant que des chiens de garde indexés :

$$\begin{aligned} P &\stackrel{\text{def}}{=} {}_1\lceil L \rceil^{dp}_{\{v\}} E \\ L &\stackrel{\text{def}}{=} m\, L_1 \\ L_1 &\stackrel{\text{def}}{=} {}_2\lceil \widetilde{v}\, S \,+\, \widetilde{\imath}\, L \rceil^{dr}_{A} L \end{aligned}$$

L_1 est introduit pour simplifier les noms des nœuds. Nous avons besoin de trois horloges : x_1 et x_2 pour les chiens de garde, et u puisque le préfixage par m est immédiat.

La partie du graphe qui nous intéresse comporte deux nœuds, appelés N_1 et N_2, dénotés par P et ${}_1\lceil L_1 \rceil^{dp}_{\{v\}} E$. Nous ne connaissons pas E et S, qui dénotent cependant deux nœuds du graphe.

Les seuls ensembles d'horloges initiales qui nous intéressent sont ceux de L et L_1, qui sont respectivement $\{u\}$ et $\{x_2\}$.

Les conditions d'activité en N_1 et N_2 sont définies par :

$$\begin{aligned} \mathtt{act}(N_1) &= \mathtt{act}({}_1\lceil L \rceil^{dp}_{\{v\}} E) \\ &= \mathtt{act}(L) \wedge (x_1 \leq dp) \\ &= \mathtt{act}(m\, L_1) \wedge (x_1 \leq dp) \\ &= (u = 0) \wedge (x_1 \leq dp) \end{aligned} \qquad \begin{aligned} \mathtt{act}(N_2) &= \mathtt{act}({}_1\lceil L_1 \rceil^{dp}_{\{v\}} E) \\ &= \mathtt{act}({}_2\lceil \widetilde{v}\, S \ + \ \widetilde{\imath}\, L \rceil^{dr}_{A} L) \wedge (x_1 \leq dp) \\ &= \mathtt{act}(\widetilde{v}\, S \ + \ \widetilde{\imath}\, L) \wedge (x_2 \leq dr) \wedge (x_1 \leq dp) \\ &= (x_1 \leq dp) \wedge (x_2 \leq dr) \end{aligned}$$

Nous obtenons les arcs issus de N_1 (P) par les déductions suivantes :

$$\frac{L \xrightarrow{m,\mathtt{true},\{x_2\}} L_1}{P \xrightarrow{m,x_1<dp,\{x_2\}} {}_1\lceil L_1 \rceil^{dp}_{\{v\}} E} \qquad P \xrightarrow{\varepsilon,x_1=dp,h(E)} E$$

Ceux issus de N_2 (${}_1\lceil L_1 \rceil^{dp}_{\{v\}} E$) se déduisent de la manière suivante :

$$\frac{\dfrac{\dfrac{\widetilde{v}\, S \xrightarrow{v,\mathtt{true},h(S)} S}{\widetilde{v}\, S + \widetilde{\imath}\, L \xrightarrow{v,\mathtt{true},h(S)} S}}{L_1 \xrightarrow{v,x_2<dr,h(S)} S}}{N_2 \xrightarrow{v,x_2<dr \wedge x_1<dp,h(S)} S} \qquad \frac{\dfrac{\dfrac{\widetilde{\imath}\, L \xrightarrow{i,\mathtt{true},\{u\}} L}{\widetilde{v}\, S + \widetilde{\imath}\, L \xrightarrow{i,\mathtt{true},\{u\}} L}}{L_1 \xrightarrow{i,x_2<dr,\{u\}} L}}{N_2 \xrightarrow{i,x_2<dr \wedge x_1<dp,\{u\}} P}$$

$$\frac{L_1 \xrightarrow{\varepsilon,x_2=dr,\{u\}} L}{N_2 \xrightarrow{\varepsilon,x_2=dr \wedge x_1<dp,\{u\}} P} \qquad N_2 \xrightarrow{\varepsilon,x_1=dp,h(E)} E$$

Nous obtenons donc le graphe temporisé de la figure 5.7. Il est semblable à celui de la figure 5.4 : les noms des horloges changent et la condition de l'arc étiqueté par m est `true` au lieu de $u = 0$, ce qui revient au même puisque cette contrainte est présente dans la condition d'activité du nœud N_1.

5.2.3.3 Le contrôleur de passage à niveau

La traduction des processus présentés dans le chapitre 3 (§ 3.3.4) fait apparaître de nouvelles horloges et un nœud supplémentaire pour chacune des composantes (figure 5.8).

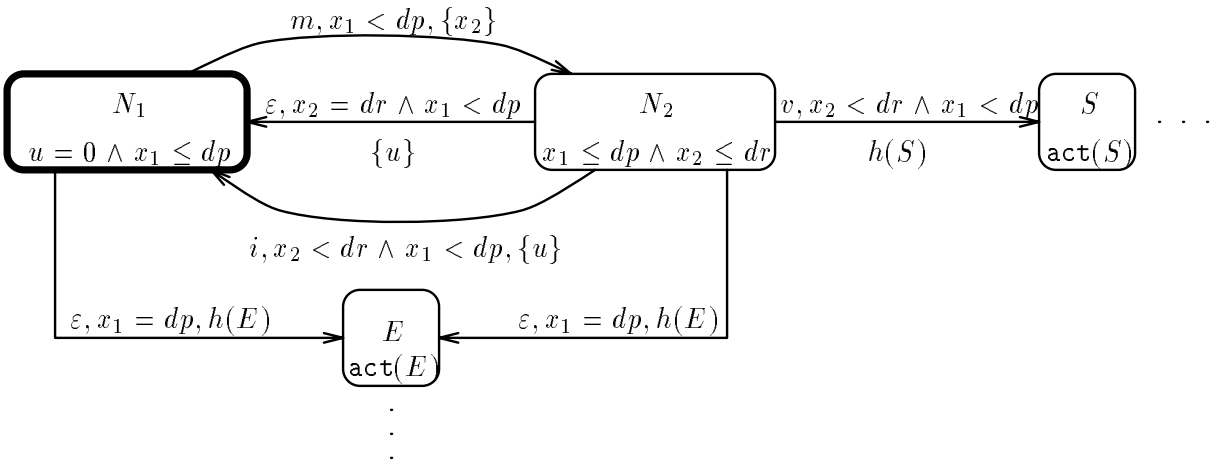


Figure 5.7 : traduction de la procédure de connexion

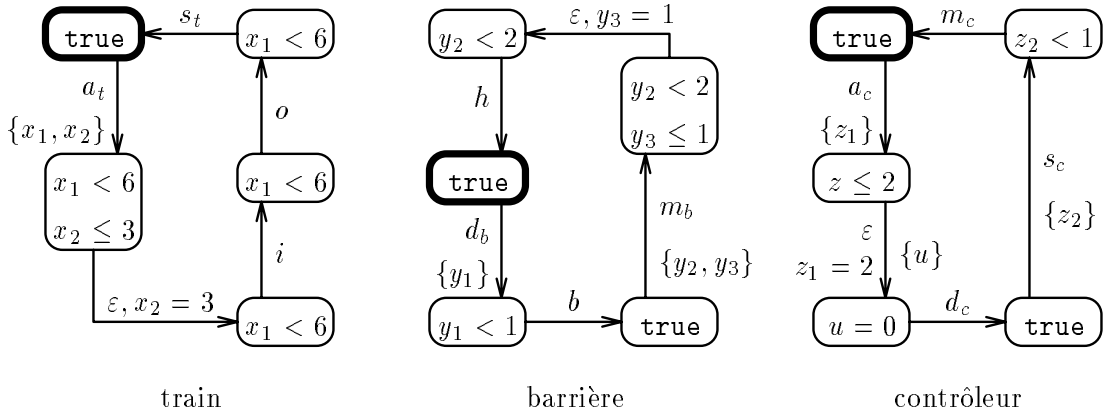


Figure 5.8 : traduction des trois composantes du contrôleur

Composer ces trois graphes donne un résultat avec un trop grand nombre d'états pour être dessiné facilement. Nous appliquons donc les règles de la composition parallèle aux graphes présentés au paragraphe 5.1.3.3. Nous obtenons alors le graphe de la figure 5.9.

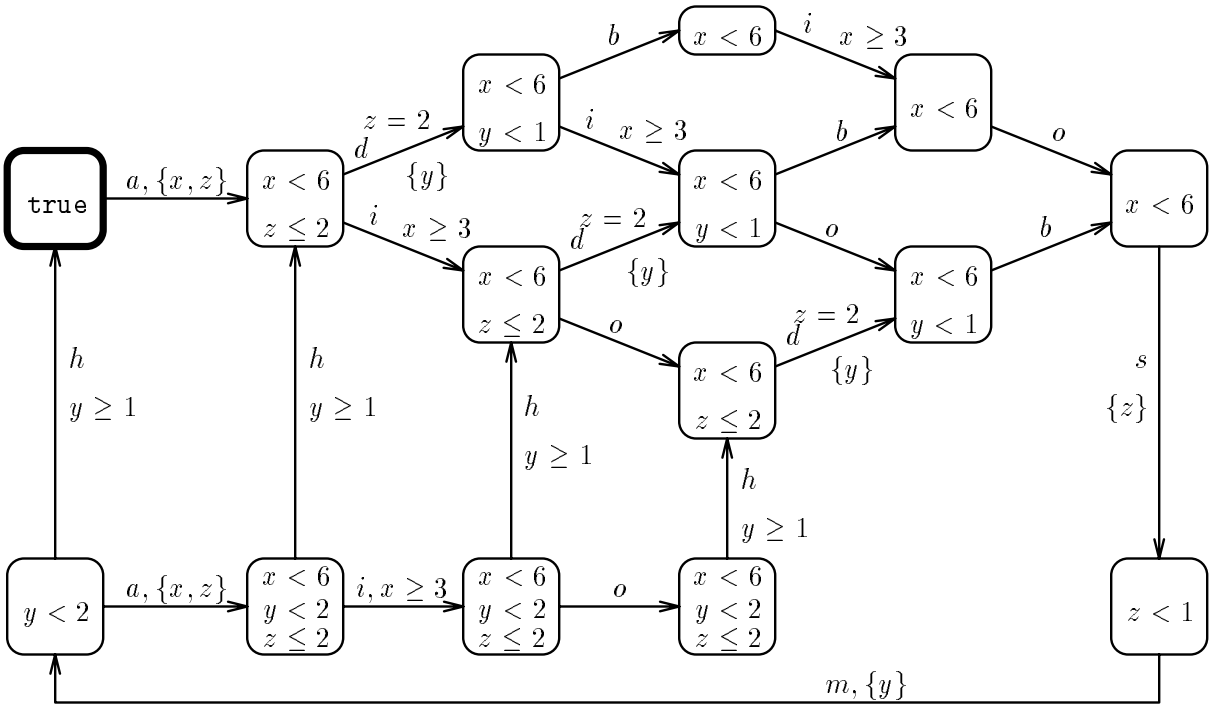


Figure 5.9 : système complet du contrôleur de passage à niveau

Chapitre 6

Model checking symbolique

Nous abordons dans ce dernier chapitre le problème de la vérification des systèmes temps réel. Nous nous intéressons ici à la technique dite de "model checking", qui consiste à déterminer quels états du modèle d'un système satisfont une formule de *logique temporelle*. Ce calcul est effectué par une analyse de l'espace d'états [CE81, QS81, CES86, EL86, CS91].

La limitation essentielle de cette technique est due à la taille du modèle. Dans les systèmes non temporisés, celle-ci croît exponentiellement avec le nombre de composantes parallèles. Une approche pour contourner ce problème consiste à représenter les ensembles d'états de manière *symbolique* (et non *énumérative*) sous la forme de prédicats. On calcule alors l'ensemble des états qui satisfont une formule comme un point fixe d'une fonctionnelle sur les prédicats.

Dans le cas non temporisé, la possibilité théorique du model checking symbolique est établie depuis de nombreuses années [EC80, Sif82]. La méthode n'est cependant mise en pratique que depuis peu de temps, grâce à la représentation des ensembles d'états au moyen de *graphes binaires de décision* (BDD) [Bry86]. Les BDD sont à présent couramment utilisés dans le domaine de la vérification [CBM89, BCD+90, Rat92].

L'histoire du model checking dans le cas des systèmes temporisés est considérablement plus réduite. Les premières résultats dans le domaine ne s'appliquent qu'aux domaines temporels discrets [EMSS89, AH91, Eme91]. Les auteurs des *timed graphs* ont présenté dans [ACD90] un algorithme de model checking énumératif pour des systèmes décrits au moyen de timed graphs, qui s'applique dans le cas d'un domaine temporel dense (les réels ou les rationnels). Comme on l'a vu dans les deux chapitres précédents, la composante temporelle engendre alors un espace d'états infini. L'algorithme de [ACD90] repose sur une construction d'un quotient fini de cet espace d'états, appelé *graphe de régions*. La vérification est alors effectuée à l'aide d'une technique énumérative sur le graphe de régions. La taille de ce graphe croît exponentiellement, non seulement en fonction du nombre de composantes parallèles du système, mais aussi en fonction des valeurs des délais ainsi que du nombre d'horloges utilisées. C'est également le cas pour des systèmes à temps discret.

Le besoin d'une approche symbolique est donc particulièrement pressant dans le domaine des systèmes temps réel.

Le problème principal pour concevoir une telle méthode consiste à définir une relation "état suivant" correcte, dont l'itération permet de calculer toutes les propriétés qu'on souhaite vérifier. Puisque nous envisageons le cas des domaines temporels denses, cette relation ne doit pas forcer

le temps à avancer de plus qu'une fraction infinitésimale. Son itération doit cependant permettre de progresser au-delà de tout instant. En effet, une contrainte de borne supérieure sur la valeur d'une horloge peut restreindre le temps qu'un système peut passer dans un ensemble d'états particulier ; par contre, elle n'impose aucune borne supérieure au nombre de transitions entre ces états pendant cette durée. Le système peut donc effectuer un nombre arbitraire de telles transitions mais ce nombre doit être fini. En d'autres termes, toute contrainte de valeur maximale d'une horloge stipule une condition d'*équité*.

Nous définissons dans la section 6.1 les modèles sur lesquels sont évaluées les formules de logique. Il diffèrent légèrement de ceux présentés dans le chapitre 4 ; la correspondance entre les deux formalismes est cependant immédiate.

Nous présentons ensuite deux *logiques temporelles temps réel*, appelées Tμ et TCTL. La première est un μ-calcul temporel, pour lequel les ensembles d'états qui satisfont une formule sont définis par des points fixes. La seconde est une extension de la logique CTL [CES86]. On montre que les deux logiques sont incomparables du point de vue de l'expressivité pour les modèles définis dans la section 6.1.

Nous décrivons un algorithme de model checking symbolique de formules de Tμ sur des *programmes temporisés à commandes gardées*. Ceux-ci permettent de représenter les graphes temporisés sous une forme plus agréable pour présenter les résultats de ce chapitre. L'algorithme consiste à transformer des prédicats qui représentent des ensembles de régions du graphe de régions (qu'on ne construit pas). Il fait également appel à des procédures de décision dans une théorie arithmétique décidable (en fait une sous-théorie extrêmement restrictive de l'arithmétique de Presburger).

Nous montrons alors que Tμ est strictement plus expressive que TCTL dans le cas des modèles des programmes temporisés à commandes gardées, si ceux-ci sont *bien temporisés*. La notion de programme bien temporisée correspond à celle présentée dans le chapitre 2 (§ 2.5.2). Nous décrivons également comment décider si un programme à commandes gardées est bien temporisé. Nous en déduisons finalement un algorithme de model checking symbolique pour TCTL sur de tels programmes. L'essentiel des résultats de ce chapitre sont présentés dans [HNSY92] ; ils ont été établis en partie à l'aide de S. Yovine et T. Henzinger, lors d'un séjour de ce dernier à Grenoble à l'automne 1991.

L'ensemble de l'étude est paramétrée par un domaine temporel D, qui est soit un D_g, soit $\mathbb{Q}_+$, soit $\mathbb{R}_+$. Par exemple, lorsqu'on écrit $\exists d$, on sous-entend $\exists d \in D$. De même, on écrit $\sigma \xrightarrow{d} \sigma'$ au lieu de $\sigma \xrightarrow[D]{d} \sigma'$.

Le résultat des procédures de décision dépend du domaine choisi : $d < 3 \Rightarrow d \leq 2$ est vrai sur $\mathbb{N}$, et faux sur $\mathbb{Q}_+$ ou $\mathbb{R}_+$. En fait, toutes les équations (ou inéquations) étant linéaires à coefficients 1, on obtient toujours le même résultat en choisissant $\mathbb{Q}_+$ ou $\mathbb{R}_+$.

Nous souhaiterions conclure cette introduction en insistant sur le fait qu'il n'y a rien de "magique" dans les méthodes symboliques. Il est prouvé [Alu91] que le model checking temps réel est PSPACE-difficile, et l'algorithme énumératif présenté dans [ACD90], qui construit le graphe de régions complet, est déjà quasi optimal dans le pire des cas. En pratique cependant, la "complexité intuitive" [BCD+90] de l'espace d'états est souvent considérablement plus réduite que celle du graphe de régions. Seule une méthode symbolique peut exploiter ce phénomène en représentant des ensembles de régions par des prédicats. L'algorithme présenté ici ne construit (symboliquement) le graphe complet que dans des cas extrêmes. Il engendre dans la plupart des

cas un quotient d'un sous-ensemble de ce graphe dont la taille dépend de la formule à vérifier.

6.1 Les modèles

Soit Π un ensemble fini de variables booléennes p (propositions).

Soit $\mathcal{H}$ un ensemble fini de variables x à valeurs dans D (horloges). Les sous-ensembles de $\mathcal{H}$ sont notés h.

L'ensemble $\Pi \cup \mathcal{H}$ est noté V, ses éléments v.

6.1.1 Système temporel

Définition 6.1 (états)
Un *état* σ est une valuation des éléments de Π et $\mathcal{H}$. L'ensemble des états est noté Σ. La valeur d'une variable v dans l'état σ est notée $\sigma(v)$.
□

Pour tout état σ et pour tout d, $\sigma + d$ dénote l'état σ' tel que $\sigma'(p) = \sigma(p)$ pour tout p, et $\sigma'(x) = \sigma(x) + d$ pour tout x. Remarquer que $\sigma + 0 = \sigma$.

Une *affectation* est un ensemble $L = \{v_1 := l_1, v_2 := l_2, \ldots, v_n := l_n\}$, où $l_i \in \{\mathtt{tt}, \mathtt{ff}\}$ si v_i est dans Π, et $l_i \in D$ si v_i est dans $\mathcal{H}$. On note $\sigma[L]$ l'état σ' tel que $\sigma'(v_i) = l_i$ pour tout $1 \leq i \leq n$, et $\sigma'(v) = \sigma(v)$ pour les autres variables. Si L consiste uniquement à remettre à zéro les horloges d'un sous-ensemble h de $\mathcal{H}$, $\sigma[L]$ peut également être noté $\sigma[h]$.

Définition 6.2 (système temporel)
Un système temporel T est un système de transitions sur Σ, sans état initial, défini par un ensemble d'horloges H_T et deux relations de transition $\mapsto_T$ et $\rightarrow_T$ telles que

1. $\mapsto_T \subseteq \Sigma \times \Sigma$ satisfait $\forall \sigma, \sigma' : \sigma \mapsto_T \sigma' \Rightarrow \exists L = \{p_i := l_i\} \cup \{x_j := 0\} : \sigma' = \sigma[L]$
2. $\rightarrow_T \subseteq \Sigma \times D \times \Sigma$ satisfait $\forall \sigma, \sigma', d : \sigma \xrightarrow{d}_T \sigma' \Rightarrow \sigma' = \sigma + d$
3. si $\sigma \mapsto_T \sigma'$ ou $\sigma \xrightarrow{d}_T \sigma'$ alors $\sigma \mapsto_T \sigma$, $\sigma' \mapsto_T \sigma'$, $\sigma \xrightarrow{0}_T \sigma$ et $\sigma' \xrightarrow{0}_T \sigma'$.
4. si $L = \{x_1 := l_1, \ldots, x_n := l_n\}$ où les x_i n'appartiennent pas à H_T, alors
$$\sigma \mapsto_T \sigma' \Rightarrow \sigma[L] \mapsto_T \sigma'[L]$$
$$\sigma \xrightarrow{d}_T \sigma + d \Rightarrow \sigma[L] \xrightarrow{d}_T \sigma[L] + d$$

□

Remarquer que $\rightarrow_T$ vérifie la propriété de déterminisme temporel. On impose de plus que $\rightarrow_T$ satisfasse la propriété d'additivité temporelle.

Une transition $\sigma \mapsto_T \sigma'$ est appelée *transition instantanée* ; une transition $\sigma \xrightarrow{d}_T \sigma'$ est appelée *transition temporelle de durée* d. Noter que pour tout état σ, l'ensemble des σ' tels que $\sigma \mapsto_T \sigma'$ est fini. En d'autres termes, le facteur de branchement de la relation $\mapsto_T$ est fini. Cela est dû au fait qu'une transition instantanée consiste à modifier la valeur d'un nombre fini de propositions et à remettre à zéro un nombre fini d'horloges.

L'ensemble H_T est appelé ensemble des *horloges* de T. La condition 4 stipule que les valeurs des horloges qui n'appartiennent pas à H_T sont sans influence sur les transitions.

Les systèmes temporels correspondent à une transformation des modèles présentés dans le chapitre 4 (§ 4.2.3) :

- Les transitions d'action deviennent des transitions instantanées non étiquetées $\mapsto_T$. Si besoin est, on peut spécifier la possibilité d'exécution d'une action ou le fait qu'on vient d'en exécuter une, au moyen de propositions $\texttt{enable}(a)$ ou $\texttt{after}(a)$.
- Les transitions $\sigma \mapsto_T \sigma$ et $\sigma \xrightarrow{0}_T \sigma$ sont ajoutées en tout état source ou but d'une autre transition (condition 3).

6.1.2 Séquences de pas et exécutions

Une exécution d'un système est modélisée par la succession des états visités. Contrairement au cas des systèmes de transitions non temporisés, une exécution d'un système temporel ne peut pas être représentée par une séquence discrète d'états visités si D est dense. En effet, une transition $\sigma \xrightarrow{d}_T \sigma'$ correspond au passage par chacun des états de l'ensemble (dense) $\{\sigma + d' \mid 0 \leq d' \leq d\}$. Il n'est pas non plus possible de modéliser une exécution par une application de D dans Σ, puisque plusieurs états peuvent être visités consécutivement mais au même instant, même lorsque le domaine temporel est $\mathbb{Q}_+$ ou $\mathbb{R}_+$.

Afin de remédier à ce problème, nous introduisons la notion de *séquence de pas* pour spécifier les exécutions d'un système temporel.

Définition 6.3 (pas)

Pour σ, σ' dans Σ et pour d dans D, on note $\sigma \overset{d}{\rightsquigarrow}_T \sigma'$, et on dit qu'il existe un *pas de longueur* d de σ à σ', si et seulement si $\sigma \xrightarrow{d}_T \sigma + d$ et $\sigma + d \mapsto_T \sigma'$. Il existe alors h dans H_T tel que $\sigma' = (\sigma + d)[h]$. On note $\sigma \rightsquigarrow_T \sigma'$, et on dit qu'il existe un pas de σ à σ', si et seulement si il existe d et un pas de longueur d entre σ et σ'.

□

Définition 6.4 (séquence de pas)

Une *séquence de pas* induite par T est une suite $\overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}}$ telle que

$$\forall i \in \mathbb{N} : \sigma_i \overset{d_i}{\rightsquigarrow}_T \sigma_{i+1}$$

□

L'ensemble des séquences de pas induites par T est noté $\mathcal{S}_T$. L'ensemble H_T est également appelé ensemble des horloges de $\mathcal{S}_T$.

Une séquence de pas $\overline{\sigma}$ *diverge* si et seulement si, pour tout d, il existe un indice i tel que $\sum_{j=0}^{i} d_j > d$. Dans le cas contraire, on dit que la séquence converge. L'ensemble des séquences divergentes de $\mathcal{S}_T$ est noté $\mathcal{S}_T^{\Delta}$.

La proposition suivante, qui permet de relier T et $\mathcal{S}_T$, se vérifie facilement.

Proposition 6.1

1. Pour tous σ et σ' dans Σ, il existe une transition instantanée $\sigma \mapsto_T \sigma'$ si et seulement si il existe une séquence $\overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}}$ dans $\mathcal{S}_T$, telle que $\sigma = \sigma_0$, $d_0 = 0$ et $\sigma' = \sigma_1$.

2. Pour tout σ dans Σ, pour tout d dans D, il existe une transition temporelle $\sigma \xrightarrow{d}_T \sigma + d$ si et seulement si il existe une séquence $\overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}}$ dans $\mathcal{S}_T$, telle que $\sigma = \sigma_0$ et $d = d_0$.

□

L'ensemble $\mathcal{S}_T$ caractérise donc entièrement le système T.

La notion de séquence divergente permet de définir les systèmes temporels bien temporisés :

Définition 6.5

Un système temporel T est *bien temporisé* si et seulement si tout préfixe fini d'un séquence de $\mathcal{S}_T$ est préfixe d'une séquence de $\mathcal{S}_T^{\triangle}$. On dit alors également que $\mathcal{S}_T$ est bien temporisé.

□

Cette définition correspond à celle présentée au paragraphe 2.5.2 du chapitre 2 dans le cas des domaines temporels discrets.

On déduit alors de la proposition 6.1 le corollaire suivant :

Corollaire 6.2

Lorsque T est bien temporisé, les préfixes finis de $\overline{\sigma}$ dans la proposition 6.1 sont des préfixes de séquences de $\mathcal{S}_T^{\triangle}$; on a donc :

$$\forall \sigma, \sigma' : \sigma \mapsto_T \sigma' \iff \exists \overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}} \in \mathcal{S}_T^{\triangle} : \sigma = \sigma_0 \wedge d_0 = 0 \wedge \sigma' = \sigma_1$$

$$\forall \sigma, \forall d : \sigma \xrightarrow{d}_T \sigma + d \iff \exists \overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}} \in \mathcal{S}_T^{\triangle} : \sigma = \sigma_0 \wedge d = d_0$$

□

Un système temporel bien temporisé T est donc entièrement déterminé par la donnée de $\mathcal{S}_T^{\triangle}$.

La notion de séquence de pas permet de spécifier celle d'exécution :

Définition 6.6 (exécution)

L'*exécution* d'une séquence de pas $\overline{\sigma} = (\sigma_i, d_i)_{i \in \mathbb{N}}$ de $\mathcal{S}_T$ est l'ensemble totalement ordonné $(, ^{\overline{\sigma}}, \leq)$, défini par

$$, ^{\overline{\sigma}} \stackrel{\text{def}}{=} \bigcup_{i \in \mathbb{N}} , _i^{\overline{\sigma}} \qquad \text{où} \qquad , _i^{\overline{\sigma}} \stackrel{\text{def}}{=} \{\langle \sigma_i + d, i \rangle \mid d \leq d_i\}$$

et

$$\langle \sigma_i + d, i \rangle \leq \langle \sigma_{i'} + d', i' \rangle \iff i < i' \ \vee \ (i = i' \wedge d \leq d')$$

□

$, ^{\overline{\sigma}}$ désigne donc l'ensemble des occurrences d'états visités au cours d'une exécution du système temporel. Les indices permettent de l'ordonner totalement. $, _i^{\overline{\sigma}}$ est l'ensemble des états visités entre σ_i et $\sigma_i + d_i$.

L'état d'une occurrence $\gamma = \langle \sigma, i \rangle$ est noté $\gamma^{\mathtt{e}}$ et son indice est noté $\gamma^{\mathtt{i}}$.

6.2 Logiques temporelles temps réel

Nous étudions dans cette section deux logiques temporelles arborescentes temps réel, qui constituent des formalismes de spécification pour les systèmes temps réel.

Nous introduisons un μ-calcul temporel Tμ avec horloges. Ses formules sont construites à partir des propositions de base et de conditions sur les horloges au moyen d'opérateurs logiques, d'un opérateur temporel binaire "next" noté $\propto$, d'un opérateur $x\downarrow$ d'initialisation d'horloge, et d'un opérateur de plus petit point fixe. Un opérateur "next" similaire a été proposé pour une extension de la logique de M. Hennessy et R. Milner [HLW91] ; l'opérateur d'initialisation est inspiré du

quantificateur "freeze" introduit dans la logique TPTL de R. Alur et T. Henzinger [AH89]. C'est leur combinaison dans une extension du μ-calcul propositionnel [Koz83] qui permet de calculer symboliquement les propriétés des systèmes temps réel.

Nous présentons également une variante de la logique TCTL [Alu91]. Celle-ci comporte également un quantificateur "freeze", que nous remplaçons par l'opérateur d'initialisation d'horloge. TCTL est basée sur deux opérateurs binaires temporels, exprimant respectivement la *possibilité* et l'*inévitabilité* de propriétés.

Les formules sont interprétées sur l'ensemble Σ des états, relativement à un ensemble de séquences de pas $\mathcal{S}$, qui est soit un $\mathcal{S}_T$, soit un $\mathcal{S}_T^\Delta$. Nous notons $\mathcal{S}^\Delta$ l'ensemble des séquences divergentes de $\mathcal{S}$. Noter que $\mathcal{S}^{\Delta\Delta} = \mathcal{S}^\Delta$.

6.2.1 Le μ-calcul temporel

Soit $\mathcal{X}$ un ensemble de variables de formules. Les formules φ du μ-calcul temporel Tμ sont définies par la syntaxe suivante :

$$\varphi \ ::= \ X \ \mid \ p \ \mid \ x + n \leq y + m \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \propto \varphi \ \mid \ x{\downarrow}\varphi \ \mid \ \mu X.\varphi$$

où $X \in \mathcal{X}$, $p \in \Pi$, $x, y \in \mathcal{H}$, $n, m \in \mathbb{N}$ et n ou m sont nuls.

Une variable de formule X est dite *liée* si elle apparaît dans la portée de l'opérateur de plus petit point fixe μX. Nous exigeons que les variables liées soient *positives*, c'est-à-dire que X apparaisse sous un nombre pair de négations dans la portée de μX.

Parmi les abréviations classiques, on peut citer $\varphi_1 \wedge \varphi_2$ pour $\neg(\neg\varphi_1 \vee \neg\varphi_2)$ et $\nu X.\varphi$ pour $\neg\mu X.\neg\varphi[\neg X/X]$ (la notation $\varphi[\neg X/X]$ désigne la formule φ où toutes les occurrences libres de X sont remplacées par $\neg X$).

L'opérateur d'initialisation $x{\downarrow}$ est utilisé pour désigner des instants ultérieurs, en mettant à zéro l'horloge x, appelée *horloge initialisée*. Dans la formule $x{\downarrow}\varphi$, les occurrences de x dans φ sont liées par l'opérateur $x{\downarrow}$. On peut procéder à un renommage de x en y, si celle-ci n'est pas présente dans φ. Lorsqu'on interprète une formule φ relativement à une séquence de pas $\mathcal{S}$, on exige que les horloges initialisées dans φ ne soient pas des horloges de $\mathcal{S}$. Si ce n'est pas le cas, on procède à un renommage des horloges initialisés de φ.

L'opérateur d'initialisation permet de définir des abréviations supplémentaires, par exemple $x \leq 5$ pour $y{\downarrow}(x \leq y + 5)$.

Un *environnement* $\mathcal{E}$ est une fonction qui associe à toute variable de formule X un sous-ensemble de Σ noté $\mathcal{E}(X)$, ensemble d'états dans lesquels X est vraie. On désigne par $\mathcal{E}[X := \Sigma']$ l'environnement $\mathcal{E}'$ tel que $\mathcal{E}'(X) = \Sigma'$ et $\mathcal{E}'(Y) = \mathcal{E}(Y)$ pour toutes les autres variables.

Un état σ de Σ *satisfait* une formule φ de Tμ relativement à un ensemble de séquences de pas $\mathcal{S}$, ce qu'on note $\sigma \models_{\mathcal{S}} \varphi$, si et seulement si, pour tout environnement $\mathcal{E}$, $\sigma \models_{\mathcal{S},\mathcal{E}} \varphi$, où :

$$\begin{aligned}
\sigma \models_{\mathcal{S},\mathcal{E}} X \quad &\text{ssi} \quad \sigma \in \mathcal{E}(X) \\
\sigma \models_{\mathcal{S},\mathcal{E}} p \quad &\text{ssi} \quad \sigma(p) = \texttt{tt} \\
\sigma \models_{\mathcal{S},\mathcal{E}} x + n \leq y + m \quad &\text{ssi} \quad \sigma(x) + n \leq \sigma(y) + m \\
\sigma \models_{\mathcal{S},\mathcal{E}} \neg\varphi \quad &\text{ssi} \quad \sigma \not\models_{\mathcal{S},\mathcal{E}} \varphi \\
\sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \vee \varphi_2 \quad &\text{ssi} \quad \sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \text{ ou } \sigma \models_{\mathcal{S},\mathcal{E}} \varphi_2
\end{aligned}$$

$$\begin{array}{rcl}
\sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \propto \varphi_2 & \text{ssi} & \text{il existe } (\sigma_i, d_i)_{i\in\mathbb{N}} \text{ dans } \mathcal{S}, \text{ telle que} \\
 & & \sigma = \sigma_0, \quad \sigma_1 \models_{\mathcal{S},\mathcal{E}} \varphi_2 \quad \text{et} \quad \forall d \leq d_0, \sigma_0 + d \models_{\mathcal{S},\mathcal{E}} \varphi_1 \vee \varphi_2 \\
\sigma \models_{\mathcal{S},\mathcal{E}} x\downarrow\varphi & \text{ssi} & \sigma[x := 0] \models_{\mathcal{S},\mathcal{E}} \varphi \\
\sigma \models_{\mathcal{S},\mathcal{E}} \mu X.\varphi & \text{ssi} & \sigma \in \bigcap \left\{ \Sigma' \subseteq \Sigma \;\middle|\; \{\sigma' \mid \sigma' \models_{\mathcal{S},\mathcal{E}[X:=\Sigma']} \varphi\} \subseteq \Sigma' \right\}
\end{array}$$

Dans le μ-calcul propositionnel classique, les formules sont interprétées relativement à un ensemble de séquences d'états discrètes. Un couple d'états consécutifs représente un "pas". L'opérateur "next" est noté $\bigcirc$. Un état satisfait $\bigcirc\varphi$ s'il existe une séquence dont il est état initial et dont l'état suivant satisfait φ. À l'aide de cet opérateur et de celui de point fixe, la propriété φ_0 :

> "il existe une séquence le long de laquelle φ_1 est continuellement satisfaite jusqu'à ce que φ_2 soit satisfaite"

est définie par la formule $\mu X.(\varphi_2 \vee (\varphi_1 \wedge \bigcirc X))$.

Il est nécessaire de disposer dans Tμ d'un opérateur "next" binaire pour exprimer une telle propriété. En effet, lorsque le domaine temporel est dense, il n'existe pas de notion d'"état suivant". La propriété φ_0 serait inexprimable au moyen d'un opérateur unaire dans Tμ. Avec l'opérateur $\propto$, elle s'écrit $\mu X.(\varphi_2 \vee (\varphi_1 \propto X))$.

Il faut définir soigneusement la sémantique de $\propto$. La formule $\varphi_1 \propto \varphi_2$ caractérise les états à partir desquels il est possible d'atteindre un état satisfaisant φ_2, *tous les états intermédiaires satisfaisant* $\varphi_1 \vee \varphi_2$. Exiger $\varphi_1 \vee \varphi_2$ pour ces états est imposé par les domaines temporels denses. Si on n'exige que φ_1, alors la formule $x \leq 2 \propto x > 2$ ne serait pas satisfaite par un état σ où la valeur de x est 1, possédant un successeur σ' (dans la relation $\rightsquigarrow_{\mathcal{S}}$) où la valeur de x est plus grande que 2. En effet, si $\sigma'(x) > 2$, alors il existe des états entre σ et σ' pour lesquels x est également plus grand que 2.

Exemple 6.1

Considérons la formule $\varphi \stackrel{\text{def}}{=} x\downarrow(p \propto (q \wedge y\downarrow x < y + 2))$.

En appliquant les définitions ci-dessus, on obtient :

Un état σ satisfait φ si et seulement si $\sigma' \stackrel{\text{def}}{=} \sigma[x := 0]$ satisfait $p \propto (q \wedge y\downarrow x < y + 2)$. Il doit donc exister une séquence de pas (σ_i, d_i) telle que $\sigma' = \sigma_0$, σ_1 satisfait $q \wedge y\downarrow x < y + 2$, et pour tout $d \leq d_0$, $\sigma' + d$ satisfait $p \vee (q \wedge y\downarrow x < y + 2)$.

L'état σ_1 doit donc satisfaire q et être tel que $\sigma_1[y := 0]$ satisfait $x < y + 2$. Cette dernière condition impose que $\sigma_1[y := 0]$ (et donc σ_1) satisfait $x < 2$. On constate au passage que la formule $y\downarrow x < y + 2$ peut bien être considérée comme une abréviation de $x < 2$.

De même, les états $\sigma' + d$ doivent satisfaire p ou $q \wedge x < 2$ pour $d \leq d_0$.

La formule φ caractérise donc les situations où q peut devenir vraie en un pas de longueur inférieure à 2, p étant continuellement vraie avant la fin du pas.

□

6.2.2 La logique TCTL

TCTL est une extension de la logique temporelle arborescente CTL [CES86]. Ses formules sont construites à partir des propositions et de contraintes sur les horloges au moyen de connectives

logiques, de l'opérateur d'initialisation d'horloge et des deux opérateurs temporels $\exists\mathcal{U}$ et $\forall\mathcal{U}$. Intuitivement, un état σ satisfait $\varphi_1 \exists\mathcal{U} \varphi_2$ s'il existe *une* exécution à partir de σ pour laquelle φ_1 est continuellement vraie jusqu'à un état où φ_2 est satisfaite. $\varphi_1 \forall\mathcal{U} \varphi_2$ signifie que cette propriété est vraie pour *toute* exécution commençant en σ. Le premier opérateur spécifie donc une *possibilité*, et le second une *inévitabilité*.

Les formules de TCTL sont définies par la syntaxe suivante :

$$\varphi \ ::= \ p \ \mid \ x + n \leq y + m \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ x{\downarrow}\varphi \ \mid \ \varphi \exists\mathcal{U} \varphi \ \mid \ \varphi \forall\mathcal{U} \varphi$$

où $p \in \Pi$, $x, y \in \mathcal{H}$, $n, m \in \mathbb{N}$ et n ou m sont nuls.

On définit classiquement plusieurs abréviations à partir des deux opérateurs temporels : $\exists\Diamond\varphi$ pour $\texttt{true}\,\exists\mathcal{U}\varphi$, $\forall\Diamond\varphi$ pour $\texttt{true}\,\forall\mathcal{U}\varphi$, $\forall\Box\varphi$ pour $\neg\exists\Diamond\neg\varphi$ et $\exists\Box\varphi$ pour $\neg\forall\Diamond\neg\varphi$.

On note $\sigma \models_{\mathcal{S}} \varphi$ lorsqu'un état σ satisfait une propriété φ de TCTL relativement à un ensemble de séquences de pas $\mathcal{S}$. La relation de satisfaction est définie par :

$\sigma \models_{\mathcal{S}} p$	ssi	$\sigma(p) = \texttt{tt}$
$\sigma \models_{\mathcal{S}} x + n \leq y + m$	ssi	$\sigma(x) + n \leq \sigma(y) + m$
$\sigma \models_{\mathcal{S}} \neg\varphi$	ssi	$\sigma \not\models_{\mathcal{S}} \varphi$
$\sigma \models_{\mathcal{S}} \varphi_1 \vee \varphi_2$	ssi	$\sigma \models_{\mathcal{S}} \varphi_1$ ou $\sigma \models_{\mathcal{S}} \varphi_2$
$\sigma \models_{\mathcal{S}} x{\downarrow}\varphi$	ssi	$\sigma[x := 0] \models_{\mathcal{S}} \varphi$
$\sigma \models_{\mathcal{S}} \varphi_1 \exists\mathcal{U} \varphi_2$	ssi	il existe $(\sigma_i, d_i)_{i\in\mathbb{N}} \in \mathcal{S}$ avec $\sigma = \sigma_0$, telle que
		$\exists\gamma \in \ , \overline{\sigma} : \gamma^{\mathsf{e}} \models_{\mathcal{S}} \varphi_2$ et $\forall\gamma' \in \ , \overline{\sigma} : \gamma' < \gamma \Rightarrow \gamma'^{\mathsf{e}} \models_{\mathcal{S}} \varphi_1 \vee \varphi_2$
$\sigma \models_{\mathcal{S}} \varphi_1 \forall\mathcal{U} \varphi_2$	ssi	pour toute $(\sigma_i, d_i)_{i\in\mathbb{N}} \in \mathcal{S}$ avec $\sigma = \sigma_0$, on a
		$\exists\gamma \in \ , \overline{\sigma} : \gamma^{\mathsf{e}} \models_{\mathcal{S}} \varphi_2$ et $\forall\gamma' \in \ , \overline{\sigma} : \gamma' < \gamma \Rightarrow \gamma'^{\mathsf{e}} \models_{\mathcal{S}} \varphi_1 \vee \varphi_2$

Les opérateurs temporels bornés présentés dans [ACD90] sont exprimables en TCTL. Par exemple, la formule $\varphi_1 \exists\mathcal{U}_{\leq 2} \varphi_2$ spécifie qu'un état satisfaisant φ_2 peut être atteint avant 2 unités de temps, φ_1 étant auparavant continuellement satisfaite. Cette propriété s'exprime en TCTL par la formule

$$x{\downarrow}\,(\varphi_1 \exists\mathcal{U} (\varphi_2 \wedge x \leq 2))$$

si x n'apparaît pas libre dans φ_1 ou φ_2.

Nous pouvons donc utiliser en tant qu'abréviations des opérateurs temporels bornés, comme $\forall\Diamond_{>5}\,\varphi$ pour $x{\downarrow}\,\forall\Diamond(\varphi \wedge x > 5)$.

Nous préférons utiliser l'opérateur d'initialisation d'horloge plutôt que le quantificateur "freeze" adopté par R. Alur [Alu91], car la notion de remise à zéro de variables dynamiques est tout à fait adaptée à la structure des modèles. Dans l'approche "freeze", la formule $x.\varphi$ affecte à x la "date" courante (x n'est pas une horloge, mais une variable classique). Les deux approches sont de toutes façons équivalentes [Alu91]. Par exemple, la formule $x{\downarrow}\,p \exists\mathcal{U}(q \wedge x \leq 2)$ (avec l'opérateur d'initialisation) s'écrit $x.p \exists\mathcal{U}(q \wedge y.\, y \leq x + 2)$ (avec le quantificateur "freeze").

6.2.3 Expressivité

Nous comparons à présent l'expressivité des deux logiques Tμ et TCTL.

Définition 6.7 (ensemble caractéristique)
Soit $\mathcal{S}$ un ensemble de séquences de pas, et φ une formule de Tμ ou TCTL. L'*ensemble caractéristique* de φ relativement à $\mathcal{S}$ est

$$[\![\varphi]\!]_{\mathcal{S}} \stackrel{\text{def}}{=} \{\sigma \in \Sigma \mid \sigma \models_{\mathcal{S}} \varphi\}$$

c'est-à-dire l'ensemble des états qui satisfont φ relativement à $\mathcal{S}$.
□

Dans cette section, on identifiera parfois une formule et son ensemble caractéristique ; celui-ci présente l'avantage d'être indépendant du langage de formules.

Définition 6.8 (expressivité)
On dit que la logique $\mathcal{A}$ est aussi expressive au sens fort (resp. faible) que la logique $\mathcal{B}$ si pour toute formule $\varphi_{\mathcal{B}}$ de $\mathcal{B}$ il existe une formule $\varphi_{\mathcal{A}}$ de $\mathcal{A}$, telle que pour tout ensemble de séquences de pas $\mathcal{S}$, $[\![\varphi_{\mathcal{A}}]\!]_{\mathcal{S}} = [\![\varphi_{\mathcal{B}}]\!]_{\mathcal{S}}$ (resp. $[\![\varphi_{\mathcal{A}}]\!]_{\mathcal{S}^\Delta} = [\![\varphi_{\mathcal{B}}]\!]_{\mathcal{S}^\Delta}$).
□

Proposition 6.3
Pour tout $\mathcal{S}$ bien temporisé, pour toute formule φ de Tμ, $[\![\varphi]\!]_{\mathcal{S}} = [\![\varphi]\!]_{\mathcal{S}^\Delta}$. En d'autres termes, une formule de Tμ ne permet pas de distinguer un ensemble de séquences de pas bien temporisé de son sous-ensemble de séquences divergentes.
□

Preuve. On montre par induction sur la structure de φ que pour tout état σ, pour tout environnement $\mathcal{E}$, $\sigma \models_{\mathcal{S},\mathcal{E}} \varphi \iff \sigma \models_{\mathcal{S}^\Delta,\mathcal{E}} \varphi$ Le résultat est évident pour les variables de formule, les propositions, les contraintes d'horloges, les opérations logiques, l'initialisation et le plus petit point fixe. Pour l'opérateur $\propto$, on a :

$$\begin{aligned}
\sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \propto \varphi_2 \quad \iff \quad & \text{il existe } (\sigma_i, d_i)_{i\in\mathbb{N}} \text{ dans } \mathcal{S}, \text{ telle que} \\
& \sigma = \sigma_0, \quad \sigma_1 \models_{\mathcal{S},\mathcal{E}} \varphi_2 \quad \text{et} \quad \forall d \leq d_0, \sigma_0 + d \models_{\mathcal{S},\mathcal{E}} \varphi_1 \vee \varphi_2
\end{aligned}$$

Or $(\sigma_0, d_0)(\sigma_1, d_1)$ est préfixe d'une séquence de $\mathcal{S}^\Delta$, donc

$$\begin{aligned}
\sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \propto \varphi_2 \quad \iff \quad & \text{il existe } (\sigma_i, d_i)_{i\in\mathbb{N}} \text{ dans } \mathcal{S}^\Delta, \text{ telle que} \\
& \sigma = \sigma_0, \quad \sigma_1 \models_{\mathcal{S},\mathcal{E}} \varphi_2 \quad \text{et} \quad \forall d \leq d_0, \sigma_0 + d \models_{\mathcal{S},\mathcal{E}} \varphi_1 \vee \varphi_2 \\
\iff \quad & \text{il existe } (\sigma_i, d_i)_{i\in\mathbb{N}} \text{ dans } \mathcal{S}^\Delta, \text{ telle que} \\
& \sigma = \sigma_0, \quad \sigma_1 \models_{\mathcal{S}^\Delta,\mathcal{E}} \varphi_2 \quad \text{et} \quad \forall d \leq d_0, \sigma_0 + d \models_{\mathcal{S}^\Delta,\mathcal{E}} \varphi_1 \vee \varphi_2 \\
& \text{(par hypothèse d'induction)}
\end{aligned}$$

On obtient donc bien $\sigma \models_{\mathcal{S},\mathcal{E}} \varphi_1 \propto \varphi_2 \iff \sigma \models_{\mathcal{S}^\Delta,\mathcal{E}} \varphi_1 \propto \varphi_2$ ■

Pour la suite, il est nécessaire de pouvoir identifier les états qui font partie d'une exécution quelconque de $\mathcal{S}$. Il est trivial de vérifier que les formules `true` $\propto$ `true` (dans Tμ) et `true`$\exists\mathcal{U}$`true` (dans TCTL) possèdent relativement à $\mathcal{S}$ les mêmes ensembles caractéristiques, dont les éléments sont exactement les *états des exécutions* de $\mathcal{S}$. Ces deux formules ainsi que leur ensemble caractéristique sont notés EdE. Pour tout état σ de EdE, il existe alors une séquence de pas $(\sigma_i, d_i)_{i\in\mathbb{N}}$ telle que $\sigma = \sigma_0$.

Proposition 6.4
Pour tout système temporel T, pour toutes formules φ_1 et φ_2 de TCTL,

$$\mathsf{EdE} \cap [\![\varphi_1 \forall\mathcal{U} \varphi_2]\!]_{\mathcal{S}_T} = \mathsf{EdE} \cap [\![\varphi_2]\!]_{\mathcal{S}_T}$$

□

Preuve. Nous prouvons l'inclusion dans chacun des deux sens.

- $\subseteq$ Soit $\sigma \in \mathsf{EdE} \cap [\![\varphi_1 \forall\mathcal{U} \varphi_2]\!]_{\mathcal{S}_T}$. Puisque σ est dans EdE, il existe une séquence de pas dont l'état initial est σ. D'après la définition de $\mathcal{S}_T$, la séquence $\overline{\sigma} = (\sigma,0)(\sigma,0)\ldots$ est également dans $\mathcal{S}_T$. Il existe alors une configuration γ dans , $^{\overline{\sigma}}$ telle que γ^{e} appartient à $[\![\varphi_2]\!]_{\mathcal{S}_T}$. Or γ est nécessairement de la forme $\langle\sigma, i\rangle$. On en conclut que σ satisfait φ_2 relativement à $\mathcal{S}_T$.
- $\supseteq$ Soit $\sigma \in \mathsf{EdE} \cap [\![\varphi_2]\!]_{\mathcal{S}_T}$. Toute séquence de pas de $\mathcal{S}_T$ dont l'état initial est σ possède la configuration $\langle\sigma, 0\rangle$, dont l'état satisfait φ_2. σ satisfait donc $\varphi_1 \forall\mathcal{U} \varphi_2$ relativement à $\mathcal{S}_T$. ■

Soit la formule de TCTL

$$\varphi \stackrel{\text{def}}{=} z{\downarrow}\,(\mathsf{EdE} \wedge \mathtt{true} \forall\mathcal{U} z = 1)$$

Soit T un système temporel bien temporisé tel que $z \notin H_T$ et $[\![\mathsf{EdE}]\!]_{\mathcal{S}_T^\Delta} \neq \emptyset$. Soit σ un état de Σ. D'après la sémantique de TCTL, on a :

$$\sigma \models_{\mathcal{S}_T} \varphi \iff \sigma[z := 0] \models_{\mathcal{S}_T} \mathsf{EdE} \wedge \mathtt{true} \forall\mathcal{U} z = 1$$

La proposition 6.4 entraîne alors :

$$\begin{aligned} \sigma \models_{\mathcal{S}_T} \varphi &\iff \sigma[z := 0] \models_{\mathcal{S}_T} \mathsf{EdE} \wedge z = 1 \\ &\iff \sigma[z := 0] \models_{\mathcal{S}_T} \mathsf{EdE} \text{ et } \sigma[z := 0] \models_{\mathcal{S}_T} z = 1 \\ &\iff \sigma[z := 0] \models_{\mathcal{S}_T} \mathsf{EdE} \text{ et } \sigma[z := 0](z) = 1 \end{aligned}$$

La dernière condition étant trivialement fausse, on déduit que $[\![\varphi]\!]_{\mathcal{S}_T} = \emptyset$.

En revanche, d'après la définition des séquences de pas divergentes, et puisque $z \notin H_T$, on obtient trivialement $[\![\varphi]\!]_{\mathcal{S}_T^\Delta} = [\![\mathsf{EdE}]\!]_{\mathcal{S}_T^\Delta}$. Donc $[\![\varphi]\!]_{\mathcal{S}_T}$ est différent de $[\![\varphi]\!]_{\mathcal{S}_T^\Delta}$.

Il n'existe pas de formule φ' de Tμ telle que, pour tout ensemble $\mathcal{S}$ de séquences de pas, $[\![\varphi]\!]_{\mathcal{S}} = [\![\varphi']\!]_{\mathcal{S}}$. En effet, la proposition 6.3 indique que pour le système temporel T ci-dessus, $[\![\varphi']\!]_{\mathcal{S}_T}$ est identique à $[\![\varphi']\!]_{\mathcal{S}_T^\Delta}$. On devrait alors avoir

$$[\![\varphi]\!]_{\mathcal{S}_T} = [\![\varphi']\!]_{\mathcal{S}_T} = [\![\varphi']\!]_{\mathcal{S}_T^\Delta} = [\![\varphi]\!]_{\mathcal{S}_T^\Delta}$$

ce qui est faux, comme on vient de le montrer. On a donc établi le théorème suivant :

Théorème 6.5

La logique Tμ n'est pas aussi expressive au sens fort que la logique TCTL.

□

On peut montrer que TCTL n'est pas aussi expressive (au sens fort comme au sens faible) que Tμ, par un raisonnement similaire à celui employé dans le cas non temporisé, pour CTL et le μ-calcul propositionnel [EC80, EH86]. Par exemple, il est possible dans Tμ d'exprimer la propriété

> "il existe une séquence d'exécution le long de laquelle la valeur d'une proposition p alterne indéfiniment entre `tt` et `ff`"

qui ne possède pas d'équivalent dans TCTL. Dans Tμ, elle s'écrit

$$\nu X.\, \mu Y.\, (p \wedge \mu Z.\, ((\neg p \wedge X) \vee \ \mathtt{true} \propto Z) \vee \ \mathtt{true} \propto Y)$$

Nous prouvons à présent que l'opérateur $\exists\mathcal{U}$ de TCTL possède un "équivalent" dans Tμ.

Proposition 6.6

Pour tout $\mathcal{S}$, pour tous ensembles caractéristiques s_1 et s_2,

$$[\![s_1 \exists\mathcal{U} s_2]\!]_{\mathcal{S}} = [\![\mathsf{EdE} \wedge \mu X.\, (s_2 \vee \ (s_1 \propto X))]\!]_{\mathcal{S}}$$

□

Preuve. Soit $A = [\![s_1 \exists\mathcal{U} s_2]\!]_{\mathcal{S}}$. Nous prouvons que A est le plus petit point fixe de la fonctionnelle

$$F(X) = \mathsf{EdE} \wedge (s_2 \vee (s_1 \propto X))$$

1. Montrons d'abord que c'est un point fixe, c'est-à-dire

$$A = \mathsf{EdE} \wedge (s_2 \vee (s_1 \propto A))$$

   $\subseteq$ Soit $\sigma \in A$. σ appartient à EdE, car il existe une séquence de pas $\overline{\sigma}$ dans $\mathcal{S}$ telle que $\sigma_0 = \sigma$. L'état σ appartient également à $s_1 \propto A$. En effet, pour tous ensembles d'états B et s, on a $B \cap \mathsf{EdE} \subseteq s \propto B$. On applique cette relation avec $B = A$ et $s = s_1$; puisque $A \subseteq \mathsf{EdE}$, on a bien $A \subseteq s_1 \propto A$.

   $\supseteq$ Soit $\sigma \in \mathsf{EdE} \wedge (s_2 \vee (s_1 \propto A))$. Puisque σ est dans EdE, il existe une séquence de pas dans $\mathcal{S}$ dont l'état initial est σ. Si σ appartient à s_2, on a terminé. Sinon, il existe une séquence de pas $\overline{\sigma}$ de $\mathcal{S}$ telle que $\sigma_0 = \sigma \in s_1 \vee s_2$, $\sigma_1 \in A$ et

$$\forall \langle \sigma, 0\rangle \leq \gamma' \leq \langle \sigma_1, 1\rangle \in \ , \overline{\sigma} : \gamma'^{\mathsf{e}} \in s_1 \vee s_2$$

   Puisque σ_1 est dans A, il existe une séquence de pas $\overline{\sigma'} \in \mathcal{S}$ telle que $\sigma'_0 = \sigma_1$ et

$$\exists \gamma \in \ , \overline{\sigma'} : \gamma^{\mathsf{e}} \in s_2 \text{ et } \forall \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \in s_1 \vee s_2$$

   En "prolongeant" le premier pas de $\overline{\sigma}$ par $\overline{\sigma'}$, on obtient une séquence de pas $\overline{\sigma''} \in \mathcal{S}$ telle que $\sigma''_0 = \sigma$ et

$$\exists \gamma \in \ , \overline{\sigma''} : \gamma^{\mathsf{e}} \in s_2 \text{ et } \forall \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \in s_1 \vee s_2$$

   Par conséquent, σ appartient à A. A est donc bien un point fixe de la fonctionnelle F.

2. Montrons que c'est le plus petit point fixe. Soit B ce plus petit point fixe. On a $B \subseteq A$. Il faut prouver que $A \subseteq B$. Soit donc σ appartenant à A. Il existe une séquence de pas $\overline{\sigma}$ dans $\mathcal{S}$ avec $\sigma_0 = \sigma$ et

$$\exists \gamma \in \ , \overline{\sigma} : \gamma^{\mathsf{e}} \in s_2 \text{ et } \forall \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \in s_1 \vee s_2$$

   Puisque $\gamma \in s_2$, γ^{e} appartient à B. Posons $n = \gamma^{\mathsf{i}}$. Nous montrons que pour tout $\gamma' \leq \gamma$, γ'^{e} est dans B. Soit i l'indice de γ'. La propriété est prouvée pour $i = n$. Si $i < n$, supposons la propriété prouvée pour $i + 1$. Il existe alors un γ'' dans $\ , \overline{\sigma}$ avec $\gamma''^{\mathsf{i}} = i + 1$ et $\gamma'^{\mathsf{e}} \rightsquigarrow \gamma''^{\mathsf{e}}$. L'état γ''^{e} est dans B, et tous les états entre γ'^{e} et γ''^{e} sont dans $s_1 \vee s_2$, donc dans $s_1 \vee B$. Par conséquent, γ'^{e} est dans $s_1 \propto B$, donc dans B. σ étant l'état d'une occurrence d'indice 0, on en conclut que σ est dans B. Par conséquent, on a bien $A \subseteq B$, ce qui termine la preuve. ■

Nous disposons à présent d'une caractérisation en termes de point fixe de l'opérateur $\exists\mathcal{U}$, quel que soit l'ensemble $\mathcal{S}$.

Dans le cas non temporisé, la caractérisation de $\varphi_1 \forall\mathcal{U} \varphi_2$ est donnée en employant l'opérateur dual de $\bigcirc$, au moyen de la formule

$$\mu X.\, \varphi_2 \vee (\varphi_1 \wedge \bigcirc X \wedge \neg\bigcirc\neg X)$$

La formule $\neg\bigcirc\varphi$ est satisfaite par les états dont *aucun* successeur immédiat ne satisfait φ.

L'opérateur dual de $\propto$ n'est d'aucune utilité dans notre cas. Cela est dû au fait qu'un état σ peut avoir un "successeur" (par la relation $\rightsquigarrow$) σ' qui satisfait φ, alors que tout les états intermédiaires

entre σ et σ' ne la satisfont pas. Donc σ satisfait $\texttt{true} \propto \varphi$, et pourtant à partir de σ il peut être inévitable de satisfaire $\neg\varphi$ avant φ.

Il est cependant possible de caractériser l'inévitabilité en termes de point fixe, mais seulement dans le cas de séquences divergentes particulières. Nous produisons cette caractérisation dans la section 6.4.3.3, qui s'appuie sur la proposition suivante :

Proposition 6.7

Soit φ_1 et φ_2 deux formules de TCTL, et z une horloge qui n'est ni une horloge de $\mathcal{S}^\Delta$, ni une horloge libre de φ_1 ou φ_2. Soit n un entier strictement positif. L'ensemble

$$E \stackrel{\text{def}}{=} [\![\varphi_1 \forall\mathcal{U} \varphi_2]\!]_{\mathcal{S}^\Delta}$$

est le plus petit point fixe de l'équation

$$[\![\varphi]\!]_{\mathcal{S}^\Delta} = [\![\varphi_2 \ \vee \ z{\downarrow}\, \varphi_1 \forall\mathcal{U}(\varphi \wedge z \leq n)]\!]_{\mathcal{S}^\Delta}$$

□

Preuve. Nous montrons d'abord que E est un point fixe, c'est-à-dire E est égal à l'ensemble

$$E_1 \stackrel{\text{def}}{=} [\![\varphi_2 \ \vee \ z{\downarrow}\, \varphi_1 \forall\mathcal{U}((\varphi_1 \forall\mathcal{U} \varphi_2) \wedge z \leq n)]\!]_{\mathcal{S}^\Delta}$$

$\subseteq$: Soit σ un élément de E. Supposons que $\sigma \notin E_1$. Il existe alors une séquence $\overline{\sigma}$ de $\mathcal{S}^\Delta$ avec $\sigma_0 = \sigma[z := 0]$, telle que

$$\forall \gamma \in , ^{\overline{\sigma}} : \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \neg(\varphi_1 \forall\mathcal{U} \varphi_2) \vee z > n \ \ \vee \ \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \neg\varphi_1 \wedge (\neg(\varphi_1 \forall\mathcal{U} \varphi_2) \vee z > n)$$

Si on applique cette formule à $\gamma = \langle \sigma_0, 0 \rangle$, dont l'état satisfait $z \leq n$, on obtient alors :

$$\sigma_0 \models_{\mathcal{S}^\Delta} \neg(\varphi_1 \forall\mathcal{U} \varphi_2) \ \ \vee \ \ \sigma_0 \models_{\mathcal{S}^\Delta} \neg\varphi_1 \wedge \neg(\varphi_1 \forall\mathcal{U} \varphi_2)$$

Donc $\sigma_0 \models \neg(\varphi_1 \forall\mathcal{U} \varphi_2)$. Puisque z n'est pas libre dans φ_1 ou φ_2, on obtient alors $\sigma \notin E$, qui est en contradiction avec l'hypothèse. Par conséquent $\sigma \in E_1$.

$\supseteq$: Soit σ dans E_1. Si $\sigma \in [\![\varphi_2]\!]_{\mathcal{S}^\Delta}$, alors σ est dans E. Sinon, pour toute séquence $\overline{\sigma}$ dont l'état initial est $\sigma_0 = \sigma[z := 0]$, il existe γ dans $, ^{\overline{\sigma}}$, tel que

$$\gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} (\varphi_1 \forall\mathcal{U} \varphi_2) \wedge z \leq n \ \wedge \ \forall \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_1 \ \vee \ ((\varphi_1 \forall\mathcal{U} \varphi_2) \wedge z \leq n)$$

Chacun des γ'^{e} satisfait donc $\varphi_1 \vee \varphi_2$.

De plus, puisque γ^{e} satisfait $\varphi_1 \forall\mathcal{U} \varphi_2$, il existe $\gamma_1 \geq \gamma$ tel que

$$\gamma_1{}^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_2 \ \wedge \ \forall \gamma \leq \gamma_1' \leq \gamma_1 : \gamma_1'{}^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_1 \ \vee \ \varphi_2$$

On obtient donc, pour toute séquence $\overline{\sigma}$ telle que $\sigma_0 = \sigma[z := 0]$:

$$\exists \gamma_1 \in , ^{\overline{\sigma}} : \gamma_1{}^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_2 \ \wedge \ \forall \gamma' \leq \gamma_1 : \gamma'^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_1 \ \vee \ \varphi_2$$

Donc $\sigma[z := 0]$ satisfait $\varphi_1 \forall\mathcal{U} \varphi_2$, et par conséquent σ est dans E.

Donc E est un point fixe. Nous montrons à présent que c'est le plus petit. Soit φ telle que

$$E_1 \stackrel{\text{def}}{=} [\![\varphi]\!]_{\mathcal{S}^\Delta} = [\![\varphi_2 \ \vee \ z{\downarrow}\, \varphi_1 \forall\mathcal{U}(\varphi \wedge z \leq n)]\!]_{\mathcal{S}^\Delta}$$

Nous montrons que $E \subseteq E_1$.

Soit σ_0 un élément de E. Posons $\delta_0 \stackrel{\text{def}}{=} \sigma_0(z)$. Supposons que σ_0 n'appartienne pas à E_1. On a alors $\sigma_0 \notin [\![\varphi_2]\!]_{\mathcal{S}^\Delta}$ et $\sigma_0 \notin [\![z{\downarrow}\, \varphi_1 \forall\mathcal{U}(\varphi \wedge z \leq n)]\!]_{\mathcal{S}^\Delta}$. Il existe donc une séquence $\overline{\sigma'}$ dans $\mathcal{S}^\Delta$, dont l'état initial est $\sigma_0' = \sigma_0[z := 0]$, telle que

$$\forall \gamma \in , ^{\overline{s'}} : \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi \wedge z \leq n \ \Rightarrow \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \neg\varphi_1 \wedge \neg\varphi \wedge z \leq n \qquad (6.1)$$

Les propriétés élémentaires de la logique nous indiquent qu'un et un seul des deux cas suivants est vrai :

1. $\exists \gamma \in , \overline{^{s'}} : \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} (\neg\varphi_1 \vee \varphi) \wedge z \leq n$
2. $\forall \gamma \in , \overline{^{s'}} : \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} z \leq n \Rightarrow \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_1 \wedge \neg\varphi$

Si le premier cas est vrai, la propriété (6.1) entraîne

$$\exists \gamma \in , \overline{^{s'}} : \gamma^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \neg\varphi_1 \wedge \neg\varphi \wedge z \leq n$$

Mais alors, comme $[\![\varphi_2]\!]_{\mathcal{S}^\Delta} \subseteq [\![\varphi]\!]_{\mathcal{S}^\Delta}$, γ^{e} satisfait $\neg\varphi_1 \wedge \neg\varphi_2$. Puisque σ_0 satisfait $\varphi_1 \forall\mathcal{U} \varphi_2$, il existe alors un $\gamma' \leq \gamma$ dont l'état est dans $[\![\varphi_2]\!]_{\mathcal{S}^\Delta}$, tel que les états de tous les $\gamma'' \leq \gamma$ sont dans $[\![\varphi_1 \vee \varphi_2]\!]$. Ceci n'est pas possible, car ça entraînerait $\sigma_0 \in E_1$, à nouveau du fait que $[\![\varphi_2]\!]_{\mathcal{S}^\Delta} \subseteq [\![\varphi]\!]_{\mathcal{S}^\Delta}$.

Le cas 2 est donc vrai. Soit alors γ'_1 un élément de $, \overline{^{\sigma'}}$ tel que ${\gamma'_1}^{\mathsf{e}}(z) = n$. Une telle occurrence existe car $\overline{\sigma'}$ est divergente.

Puisque z est une horloge sans influence sur $\mathcal{S}^\Delta$ et non libre dans φ_1 ou φ_2, à $\overline{\sigma'}$ correspond une séquence $\overline{\sigma}$ d'état initial σ_0 telle que, pour tout i dans $\mathbb{N}$,

$$\forall p : \sigma_i(p) = \sigma'_i(p) \qquad \forall x \neq z : \sigma_i(x) = \sigma'_i(x) \qquad \sigma_i(z) = \sigma'_i(z) + \delta_0$$

À γ'_1 correspond donc une occurrence γ_1 dans $, \overline{^{\sigma}}$, telle que

$${\gamma_1}^{\mathsf{e}}(z) = \delta_0 + n \text{ et } \forall \gamma' \leq \gamma_1 : {\gamma'}^{\mathsf{e}} \models_{\mathcal{S}^\Delta} \varphi_1 \wedge \neg\varphi$$

Tous ces états ${\gamma'}^{\mathsf{e}}$ satisfont donc $\neg\varphi_2$. Or σ_0 satisfait $\varphi_1 \forall\mathcal{U} \varphi_2$; les états ${\gamma'}^{\mathsf{e}}$ doivent donc également être dans E.

Soit $\sigma_1 \stackrel{\text{def}}{=} {\gamma_1}^{\mathsf{e}}$ et $\delta_1 \stackrel{\text{def}}{=} \delta_0 + n$. On a $\sigma_1 \in E$ et $\sigma_1 \notin E_1$. On peut donc reproduire sur σ_1 le même raisonnement que celui qu'on vient d'effectuer sur σ_0. On construit ainsi de proche en proche une séquence divergente infinie $\overline{\sigma''}$ de $\mathcal{S}^\Delta$ avec $\sigma''_0 = \sigma_0$, dont les états des occurrences satisfont tous $\neg\varphi_2$. On aboutit donc à une contradiction, puisque $\sigma_0 \models_{\mathcal{S}^\Delta} \varphi_1 \forall\mathcal{U} \varphi_2$. ■

On peut prouver que la fonctionnelle

$$X = \varphi_2 \vee z{\downarrow}\, \varphi_1 \forall\mathcal{U} (X \wedge z \leq n)$$

n'est pas continue ; son plus petit point fixe n'est donc pas calculable comme la limite d'une suite croissante d'ensembles caractéristiques.

Cette proposition est appliquée dans la section 6.4.3.3 pour trouver, dans un cas restreint de séquences de pas divergentes, une formule de Tμ équivalente à une inévitabilité quelconque de TCTL. Elle ne suffit malheureusement pas dans le cas général, car il faut encore caractériser l'inévitabilité bornée. Nous conjecturons que cela est impossible dans le cas général :

Conjecture 6.8

Il existe des formules φ_1 et φ_2 de TCTL et un entier positif n, tel que pour toute formule φ de Tμ, il existe un ensemble $\mathcal{S}^\Delta$ tel que

$$[\![z{\downarrow}\, \varphi_1 \forall\mathcal{U} (\varphi_2 \wedge z \leq n)]\!]_{\mathcal{S}^\Delta} \neq [\![\varphi]\!]_{\mathcal{S}^\Delta}$$

En d'autres termes, il existe des formules d'inévitabilité bornée qui n'ont pas d'équivalent dans Tμ.

□

Nous présentons dans la section 6.4.3.2 quelques éléments à l'appui de cette conjecture.

Si cette conjecture est vérifiée, on peut en déduire que Tμ n'est pas aussi expressive au sens faible que TCTL, c'est-à-dire que les deux logiques sont incomparables même pour les ensembles de séquences de pas divergentes.

6.3 Programmes temporisés à commandes gardées

Nous présentons dans cette section un autre formalisme de description des systèmes temps réels : les *programmes temporisés à commandes gardées*. Ils s'apparentent fortement aux graphes temporisés, mais ils sont plus agréables à manipuler pour décrire l'algorithme de model checking. Il n'est de toutes façons pas difficile de passer de l'un à l'autre des formalismes, leurs niveaux de description étant équivalents.

Nous prouvons alors dans la section 6.4 que lorsqu'on se restreint aux modèles de ces programmes, toute formule de TCTL possède un équivalent dans Tμ.

6.3.1 Définition et sémantique

Nous étendons la notion de *condition* présentée dans le chapitre précédent (§ 5.1.1), en y incluant les propositions de base.

Définition 6.9
L'ensemble Φ des *prédicats d'état* est défini par la syntaxe suivante :

$$\phi ::= p \quad | \quad x \leq k \quad | \quad x < k \quad | \quad x + n \leq y + m \quad | \quad \neg\phi \quad | \quad \phi \wedge \phi$$

où $p \in \Pi$, $x, y \in \mathcal{H}$, $k, n, m \in \mathbb{N}$ et n ou m sont nuls.

□

Un prédicat d'état est interprété de manière standard comme un prédicat sur Σ, et on note $\sigma \models \phi$ ("σ satisfait ϕ") pour $\phi(\sigma) = \texttt{tt}$.

Il est évident que pour tout prédicat d'état ϕ, le problème (paramétré par D) de la validité de ϕ est décidable.

Un *programme temporisé à commandes gardées* (PCG en abrégé) P est un quadruplet $(H_P, \phi_P^0, \phi_P^\Box, \mathcal{G}_P)$, tel que

- H_P est un sous-ensemble de $\mathcal{H}$, appelé *ensemble d'horloges de* P.
- ϕ_P^0 est un prédicat d'état, dont les horloges sont dans H_P, appelé *condition initiale* de P. Il définit l'ensemble des états initiaux possibles du programme.
- $\phi_P^\Box$ est également un prédicat d'état à horloges dans H_P, appelé *invariant* de P. Il restreint le comportement du programme aux états qui le satisfont.
- $\mathcal{G}_P$ est un ensemble fini de *commandes gardées* G, de la forme $\psi \xrightarrow{\alpha} L$, où
  - La *garde* ψ est un prédicat d'état ;
  - L'étiquette α est une action ou ε ;
  - L est un ensemble $\{p_i := \phi_i,\ i \in I\} \cup \{x_j := 0,\ j \in J\}$, où les p_i sont dans Π, les ϕ_i dans Φ, et les x_j dans H_P.

Pour un tel ensemble L et pour un prédicat d'état ϕ, on note $\phi[L]$ le prédicat d'état résultant de ϕ par substitution simultanée des ϕ_i (resp. de 0) aux p_i (resp. aux x_j). Si σ est un état, on note $\sigma\langle L\rangle$ l'état $\sigma[L']$, où $L' = \{p_i := \phi_i(\sigma),\ i \in I\} \cup \{x_j := 0,\ j \in J\}$, c'est-à-dire l'état σ' tel que

$$\forall i \in I : \sigma'(p_i) = \phi_i(\sigma) \qquad \forall j \in J : \sigma'(x_j) = 0$$

les autres variables restant inchangées.

Une commande gardée $G = \psi \xrightarrow{\alpha} L$ peut être exécutée dans un état σ si σ satisfait $\psi \wedge \phi_P^{\Box}$ et si l'état but $\sigma\langle L\rangle$ satisfait $\phi_P^{\Box}$.

La sémantique d'un PCG P est définie en termes d'un ensemble $\mathcal{S}_P$ de séquences de pas induites par un système temporel T_P dont l'ensemble d'horloges est H_P. T_P est décrit par les deux relations $\mapsto_P$ et $\rightarrow_P$ de la manière suivante : pour tous σ et σ' dans Σ,

$$\sigma \mapsto_P \sigma' \iff \sigma \models \phi_P^{\Box}, \sigma' \models \phi_P^{\Box} \text{ et } (\sigma = \sigma' \text{ ou } \exists \psi \xrightarrow{\alpha} L \in \mathcal{G}_P : \sigma \models \psi \text{ et } \sigma' = \sigma\langle L\rangle)$$
$$\forall d : \sigma \xrightarrow{d}_P \sigma' \iff \sigma' = \sigma + d \text{ et } \forall d' \leq d : \sigma + d' \models \phi_P^{\Box}$$

Les relations $\overset{d}{\rightsquigarrow}_P$ sont définies à partir de ces relations de la même manière que dans la section 6.1.2, ce qui permet d'obtenir $\mathcal{S}_P$ et $\mathcal{S}_P^{\triangle}$, respectivement ensemble des séquences de pas et ensemble des séquences de pas divergentes du programme P.

Un PCG est dit bien temporisé si son système temporel est bien temporisé (cf. définition 6.5).

6.3.2 Exemples

Soit le PCG $P_1 = (H_1, \phi_1^0, \phi_1^{\Box}, \mathcal{G}_1)$, défini par :

$$\begin{aligned} H_1 &\overset{\text{def}}{=} \{x\} \\ \phi_1^0 &\overset{\text{def}}{=} s = 0 \wedge x = 0 \\ \phi_1^{\Box} &\overset{\text{def}}{=} (s = 0 \wedge x \leq 4) \vee (s = 1 \wedge x \leq 4) \vee (s = 2) \\ \mathcal{G}_1 &\overset{\text{def}}{=} \left\{ s = 0 \xrightarrow{a} s := 1 \ , \ s = 1 \xrightarrow{b} s := 0 \ , \ (s = 0 \wedge x \geq 1) \xrightarrow{c} s := 2 \right\} \end{aligned}$$

s est une variable à valeurs dans $\{0, 1, 2\}$, représentant deux variables booléennes.

On peut représenter ce programme sous la forme du graphe temporisé de la figure 6.1. La variable s modélise les nœuds du graphe.

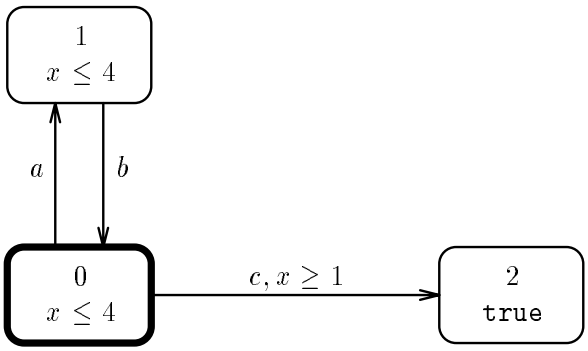


Figure 6.1 : graphe temporisé du programme P_1

Le comportement du programme est le suivant. Il commence avec s et x tous les deux nuls. La valeur de s est alternativement 0 et 1 avant l'instant 4 ; elle peut passer de 0 à 2 à tout instant entre 1 et 4. La valeur 4 est imposée par l'invariant, la valeur 1 par la transition c du nœud 0 au nœud 2. Une fois que s vaut 2, c'est-à-dire quand le nœud 2 est atteint, le programme ne fait plus rien.

L'ensemble $\mathcal{S}_{P_1}$ des séquences de pas $(\sigma_i, d_i)_{i\in\mathbb{N}}$ de ce programme est caractérisé par : pour tout i dans $\mathbb{N}$,

$$\sigma_i(s) \in \{0,1,2\} \qquad \sigma_{i+1}(x) = \sigma_i(x) + d_i$$
$$\sigma_i(s) = 0 \Rightarrow \sigma_i(x) \leq 4 \qquad \sigma_i(s) = 1 \Rightarrow \sigma_i(x) \leq 4 \wedge \sigma_{i+1}(s) \neq 2$$
$$\sigma_i(s) = 2 \Rightarrow \sigma_{i+1}(s) = 2 \qquad \sigma_i(s) = 0 \wedge \sigma_{i+1}(s) = 2 \Rightarrow 1 \leq \sigma_{i+1}(x) \leq 4$$

Les éléments de $\mathcal{S}^{\mathbb{A}}_{P_1}$ (séquences divergentes) satisfont la propriété suivante :

$$\exists i \in \mathbb{N}, \forall j \geq i : \sigma_j(s) = 2$$

En d'autres termes, toute séquence divergente finit par donner définitivement la valeur 2 à s.

P_1 est bien temporisé. En effet, tout préfixe fini d'une séquence de $\mathcal{S}_{P_1}$ est préfixe d'une séquence divergente. Sous l'hypothèse d'équité du passage du temps, le nœud 2 est donc inévitablement atteint.

Un exemple de séquence convergente est (σ_i, d_i), où, pour tout i dans $\mathbb{N}$,

$$\sigma_{2i}(s) = 0 \qquad \sigma_{2i+1}(s) = 1 \qquad \sigma_i(x) = 4\sum_{j=0}^{i} \frac{2^i - 1}{2^i} \qquad d_i = \frac{4}{2^{i+1}}$$

Cette séquence n'est pas équitable du point de vue du passage du temps, car elle empêche celui-ci d'aller au-delà de l'instant 4, alors qu'une telle évolution est toujours possible.

Considérons à présent le PCG $P_2 = (H_2, \phi_2^0, \phi_2^\Box, \mathcal{G}_2)$, défini par :

$$H_2 \stackrel{\text{def}}{=} \{x\}$$
$$\phi_2^0 \stackrel{\text{def}}{=} s = 0 \wedge x = 0$$
$$\phi_2^\Box \stackrel{\text{def}}{=} (s = 0 \wedge x \leq 5) \vee (s = 1 \wedge x \leq 5) \vee (s = 2)$$
$$\mathcal{G}_2 \stackrel{\text{def}}{=} \left\{ s = 0 \xrightarrow{a} s := 1 \ , \ s = 1 \xrightarrow{b} s := 0 \ , \ (s = 0 \wedge 1 \leq x \leq 4) \xrightarrow{c} s := 2 \right\}$$

Le graphe temporisé correspondant est présenté sur la figure 6.2.

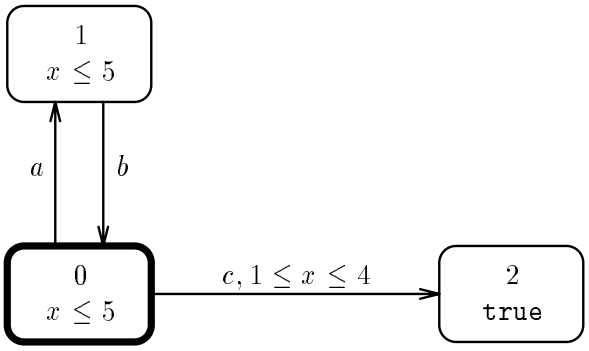


Figure 6.2 : graphe temporisé du programme P_2

Les séquences de pas définissant le comportement de P_2 sont cette fois caractérisées par :

$$\sigma_i(s) \in \{0,1,2\} \qquad \sigma_{i+1}(x) = \sigma_i(x) + d_i$$
$$\sigma_i(s) = 0 \Rightarrow \sigma_i(x) \leq 5 \qquad \sigma_i(s) = 1 \Rightarrow \sigma_i(x) \leq 5 \wedge \sigma_{i+1}(s) \neq 2$$
$$\sigma_i(s) = 2 \Rightarrow \sigma_{i+1}(s) = 2 \qquad \sigma_i(s) = 0 \wedge \sigma_{i+1}(s) = 2 \Rightarrow 1 \leq \sigma_{i+1}(x) \leq 4$$

Le programme P_2 n'est pas bien temporisé. En effet, il est possible d'"exécuter" P_2 en alternant entre $s = 0$ et $s = 1$ jusqu'à un instant où la transition c n'est plus possible. La séquence résultant est alors inévitablement convergente, l'instant 5 ne pouvant être dépassé. Un exemple de séquence irrémédiablement convergente après un certain nombre de pas est (σ'_i, d'_i), avec, pour tout i dans $\mathbb{N}$,

$$\S'_{2i}(s) = 0 \qquad \sigma'_{2i+1}(s) = 1 \qquad \sigma'_i(x) = 5\sum_{j=0}^{i} \frac{2^i - 1}{2^i} \qquad d_i = \frac{5}{2^{i+1}}$$

Par contre, les séquences divergentes de P_2 sont exactement les mêmes que celles de P_1 : ce sont celles qui mènent dans le nœud 2 entre les instants 1 et 4.

6.4 Model checking

Nous présentons un algorithme de vérification de propriétés exprimées dans Tμ. Il consiste, pour un PCG P donné, à calculer, sous la forme d'un prédicat d'état $|\varphi|$, l'ensemble $[\![\varphi]\!]_{\mathcal{S}_P}$ des états satisfaisant une formule φ relativement à $\mathcal{S}_P$.

Nous exposons également comment calculer l'ensemble caractéristique d'une inévitabilité de TCTL ($\forall\mathcal{U}$), ce qui est possible pour les modèles des PCG bien temporisés, mais pas dans le cas général des systèmes temporels, même bien temporisés.

La correction de l'algorithme et de la traduction de $\forall\mathcal{U}$ reposent entièrement sur la notion de *graphe de régions* présentée ci-dessous, utilisée pour le model checking énumératif [ACD90, Alu91].

6.4.1 Graphe de régions

Soit $P \stackrel{\text{def}}{=} (H_P, \phi^0_P, \phi^\Box_P, \mathcal{G}_P)$ un programme temporisé à commandes gardées, et soit φ une formule de Tμ. On note c la plus grande constante entière apparaissant dans les prédicats d'état de P (donc dans ϕ^0_P, $\phi^\Box_P$ et les gardes des commandes) et de φ.

Définition 6.10 (région)
Nous définissons une relation d'équivalence $\overset{c}{\sim}$ entre états de Σ par : $\sigma \overset{c}{\sim} \sigma'$ si et seulement si

1. Pour toute proposition p de Π, $\sigma(p) = \sigma'(p)$;
2. Pour toute horloge x de $\mathcal{H}$, $\texttt{int}(\sigma(x)) = \texttt{int}(\sigma'(x))$ ou $\sigma(x) > c$ et $\sigma'(x) > c$, (`int` dénote la partie entière) ;
3. Pour tout x dans $\mathcal{H}$ telle que $\sigma(x) \leq c$ ou $\sigma'(x) \leq c$, $\texttt{frac}(\sigma(x)) = 0$ si et seulement si $\texttt{frac}(\sigma'(x)) = 0$, où `frac` dénote la partie fractionnelle ;
4. Pour tout couple d'horloges (x, y) de $\mathcal{H}$, et pour tous n et m dans $\mathbb{N}$ tels que $n + m \leq c$, $\sigma(x) + n \leq \sigma(y) + m$ si et seulement si $\sigma'(x) + n \leq \sigma'(y) + m$.

Une *région* est une classe d'équivalence de $\overset{c}{\sim}$. La région contenant σ est notée $[\sigma]$. Une *région terminale* est une région dont les états satisfont $x > c$ pour toute horloge x.

□

On constate facilement que l'ensemble $\mathcal{R}$ des régions est fini. La figure 6.3 présente les différentes régions existantes pour une valeur particulière des propositions, dans le cas où $\mathcal{H} = \{x, y\}$ et $c = 3$. On distingue ici des régions à zéro dimension (les points), à une dimension (les segments ouverts et les demi-droites ouvertes) et à deux dimensions (les polygones ouverts bornés par des segments et les polygones ouverts non bornés). On dénombre également quinze régions terminales, qui sont les demi-droites ouvertes et les polygones ouverts non bornés situés dans le quadrant du plan défini par $x > 3$ et $y > 3$.

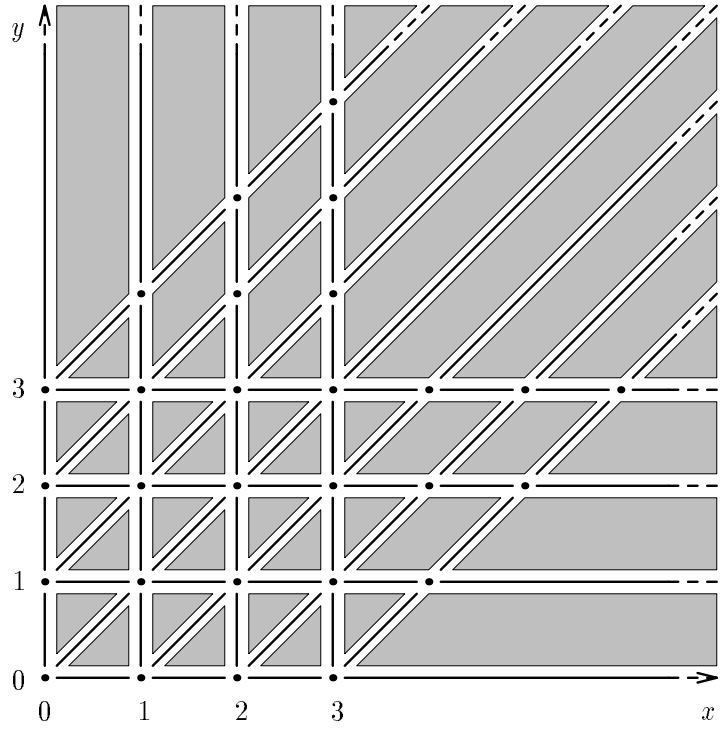


Figure 6.3 : régions pour deux horloges

Chaque région r est caractérisée par un prédicat d'état ϕ_r dont la plus grande constante entière est inférieure ou égale à c. C'est-à-dire, pour tout état σ, $\sigma \in r$ si et seulement si $\sigma \models \phi_r$ ($\phi(\sigma) = \texttt{tt}$). Le prédicat $\phi_{[\sigma]}$ est défini comme la conjonction des prédicats suivants :

1. Pour tout p dans Π, si $\sigma(p) = \texttt{tt}$ alors $\boxed{p}$ sinon $\boxed{\neg p}$
2. Pour tout x dans $\mathcal{H}$,
   - si $\sigma(x) = n \leq c$, où $n \in \mathbb{N}$, alors $\boxed{x = n}$
   - si $n < \sigma(x) < n + 1 \leq c$, où $n \in \mathbb{N}$, alors $\boxed{n < x < n + 1}$
   - si $\sigma(x) > c$ alors $\boxed{x > c}$
3. pour tous x et y dans $\mathcal{H}$,
   - si $\sigma(x) = \sigma(y) + n$, où $n \in \mathbb{N}$ et $n \leq c$, alors $\boxed{x = y + n}$
   - si $\sigma(y) + n < \sigma(x) < \sigma(y) + n + 1$, où $n \in \mathbb{N}$ et $n < c$, alors $\boxed{y + n < x < y + n + 1}$
   - si $\sigma(x) > \sigma(y) + c$, alors $\boxed{x > y + c}$

Exemple 6.2

Supposons $D = \mathbb{Q}_+$, $c = 3$ $\Pi = \{p, q\}$ et $\mathcal{H} = \{x, y\}$.

1. Soit l'état σ défini par

$$\sigma(p) = \texttt{tt} \qquad \sigma(q) = \texttt{ff} \qquad \sigma(x) = 0{,}1 \qquad \sigma(y) = 1{,}4$$

Alors un état est dans $[\sigma]$ si et seulement si il satisfait le prédicat d'état

$$p \wedge \neg q \wedge (0 < x < 1) \wedge (1 < y < 2) \wedge (x + 1 < y < x + 2)$$

2. Soit à présent l'état σ' défini par

$$\sigma(p) = \sigma(q) = \texttt{tt} \qquad \sigma(x) = 4{,}3 \qquad \sigma(y) = 1{,}4$$

Les états de $[\sigma']$ sont alors caractérisés par le prédicat d'état

$$p \wedge q \wedge (x > 3) \wedge (1 < y < 2) \wedge (y + 2 < x < y + 3)$$

3. Un prédicat d'état caractérisant $[\sigma] \cup [\sigma']$ est

$$p \wedge (1 < y < 2) \wedge$$
$$((\neg q \wedge (0 < x < 1) \wedge (x + 1 < y < x + 2)) \ \vee \ (q \wedge (x > 3) \wedge (y + 2 < x < y + 3)))$$

□

Remarque 6.1

Noter qu'une région dont les états satisfont $x \leq c$ pour toute horloge x peut également être caractérisée par la conjonction des prédicats adéquats p ou $\neg p$ et $x = n$ ou $n < x < n + 1$ et par l'ordre des parties fractionnelles des valeurs des horloges. Par exemple, la région $[\sigma]$ de l'exemple 6.2 ci-dessus est caractérisée par

$$p \wedge \neg q \wedge (0 < x < 1) \wedge (1 < y < 2)$$

et

$$\texttt{frac}(x) < \texttt{frac}(y)$$

□

On peut déduire de la caractérisation des régions par des prédicats d'état la propriété suivante :

Proposition 6.9

À tout ensemble R de régions correspond un prédicat d'état ϕ_R, dont les constantes sont inférieures à c, tel que

$$\forall \sigma : \sigma \in R \iff \sigma \models \phi_R$$

ϕ_R est bien entendu la disjonction des ϕ_r, $r \in R$.

□

Il est de plus facile de prouver que

Proposition 6.10

Pour tout prédicat d'état ϕ dont aucune constante n'excède c, pour tous états σ et σ', si $\sigma \overset{c}{\sim} \sigma'$, alors $\phi(\sigma) = \phi(\sigma')$.

□

Par conséquent, tout prédicat d'état dont les constantes sont bornées par c peut être considéré comme un ensemble de régions. Nous utilisons indifféremment la notation ϕ pour les prédicats d'état et les ensembles de régions.

Finalement, on peut démontrer que les relations $\mapsto_P$ et $\bigcup_{d\in D} \xrightarrow{d}_P$ sont *stables* par rapport à la partition de Σ en régions ; c'est-à-dire :

Proposition 6.11

Pour tous σ, σ' et d, $\quad \sigma \mapsto_P \sigma' \;\Rightarrow\; \forall \sigma_1 \in [\sigma], \exists \sigma'_1 \in [\sigma'] : \sigma_1 \mapsto_P \sigma'_1$

$$\sigma \xrightarrow{d}_P \sigma' \;\Rightarrow\; \forall \sigma_1 \in [\sigma], \exists d_1, \exists \sigma'_1 \in [\sigma'] : \sigma_1 \xrightarrow{d_1}_P \sigma'_1$$

□

En d'autres termes, si un état d'une région r possède une transition instantanée (resp. temporelle de durée quelconque) vers une région r', alors c'est également le cas pour tous les états de r.

On peut en conclure directement que la relation $\rightsquigarrow_P$ est elle-même stable. Donc, s'il existe un pas d'un état de r vers un état de r', alors il en existe un à partir de tout état de r. Le théorème suivant, sur lequel s'appuient l'algorithme énumératif de [ACD90, Alu91] ainsi que l'algorithme symbolique de la section 6.4 est donc trivialement vrai, d'après la sémantique de Tμ et TCTL.

Théorème 6.12

Pour toute formule φ de Tμ ou TCTL dont aucune constante n'excède c, pour tous états σ et σ' appartenant à la même région,

$$\sigma \models_{\mathcal{S}_P} \varphi \iff \sigma' \models_{\mathcal{S}_P} \varphi \qquad \text{et} \qquad \sigma \models_{\mathcal{S}_P^{\Delta}} \varphi \iff \sigma' \models_{\mathcal{S}_P^{\Delta}} \varphi$$

Par conséquent il existe un prédicat d'état dont aucune constante n'excède c, tel que $[\![\varphi]\!]_{\mathcal{S}_P} = [\![\phi]\!]_{\mathcal{S}_P}$ et $[\![\varphi]\!]_{\mathcal{S}_P^{\Delta}} = [\![\phi]\!]_{\mathcal{S}_P^{\Delta}}$.

□

Nous définissons à présent la notion de successeur temporel d'une région.

Définition 6.11

Une région r' de $\mathcal{R}$ est un *successeur temporel* d'une région r si l'une des deux conditions suivantes est remplie

1. r et r' sont distinctes, et il existe σ dans r et d dans D^* tel que $\sigma + d \in r'$ et pour tout $d' \leq d$, $\sigma + d' \in r \cup r'$;
2. $r = r'$ est une région terminale.

□

On montre facilement que toute région possède un successeur et que de plus il est unique. Le successeur temporel de la région r est noté $\mathtt{succ}_t(r)$.

Exemple 6.3

Nous considérons les deux régions définies dans l'exemple 6.2.

1. Le successeur de la région définie par le prédicat

$$p \wedge \neg q \;\wedge (0 < x < 1) \wedge (1 < y < 2) \wedge (x + 1 < y < x + 2)$$

est la région caractérisée par $p \wedge \neg q \;\wedge (0 < x < 1) \wedge (y = 2) \wedge (x + 1 < y < x + 2)$.

2. Celui de la région caractérisée par

$$p \wedge q \wedge (x > 3) \wedge (1 < y < 2) \wedge (y + 2 < x < y + 3)$$

(avec $c = 3$) est la région définie par $p \wedge q \wedge (x > 3) \wedge (y = 2) \wedge (y + 2 < x < y + 3)$.

□

La proposition suivante fournit une borne supérieure au nombre d'itérations nécessaires de la fonction $\mathtt{succ}_t$ pour passer de la région d'un état σ à celle de $\sigma + d$, quel que soit d.

Proposition 6.13

Soit $\hbar$ le cardinal de $\mathcal{H}$, soit σ un état de Σ, et soit d un élément de D, dont la partie entière supérieure est δ. Il existe $n \leq 2\delta\hbar$ tel que $[\sigma + d] = \mathtt{succ}_t^n([\sigma])$ et pour tout $d' \leq d$ il existe $m \leq n$ tel que $[\sigma + d] = \mathtt{succ}_t^m([\sigma])$.

□

Preuve. Nous supposons que les horloges de $\mathcal{H}$ sont notées x_i avec $1 \leq i \leq \hbar$, et que dans σ elles vérifient

$$\forall i < \hbar : \mathtt{frac}(x_{i+1}) \leq \mathtt{frac}(x_i)$$

Il suffit de prouver la propriété pour $d = 1$. Nous nous plaçons dans le pire des cas. Celui-ci est caractérisé par :

$$\forall i < \hbar : \mathtt{frac}(x_{i+1}) < \mathtt{frac}(x_i)$$
$$\forall i \in [1, \hbar] : x_i \leq c - 1$$

Deux situations sont alors possibles :

1. les parties fractionnelles des x_i sont toutes non nulles ;
2. $x_\hbar$ est entier.

Les deux cas sont traités de manière similaire, nous ne présentons que le premier. Appelons r_0 la région de σ. Outre les valeurs des propositions de base et l'ordre entre les parties fractionnelles défini ci-dessus, elle est caractérisée (cf. rq. 6.1) par la donnée de $\hbar$ entiers $m_i < c - 2$ tels que

$$\forall i \in [1, \hbar] : m_i < x_i < m_i + 1$$

La région $[\sigma + 1]$ est alors caractérisée par les mêmes valeurs des propositions de base et le même ordre entre parties fractionnelles que r_0, et par

$$\forall i \in [1, \hbar] : m_i + 1 < x_i < m_i + 2$$

Pour i entre 1 et $\hbar$, on définit deux régions r_{2i-1} et r_{2i}, caractérisées par le même ordre de parties fractionnelles et les mêmes valeurs de propositions de base que r_0 et telles que, dans r_{2i-1}, les valeurs des horloges vérifient

$$\forall j < i : m_j + 1 < x_j < m_j + 2$$
$$x_i = m_i + 1$$
$$\forall j > i : m_j < x_j < m_j + 1$$

et dans r_{2i} elles vérifient

$$\forall j \leq i : m_j + 1 < x_j < m_j + 2$$
$$\forall j > i : m_j < x_j < m_j + 1$$

On constate alors facilement, d'une part que pour tout k entre 1 et $2\hbar$, $r_k = \mathtt{succ}_t(r_{k-1})$, et d'autre part que $[\sigma + 1] = r_{2\hbar}$. Il suffit donc de $2\hbar$ itérations de la fonction $\mathtt{succ}_t$ pour passer de $[\sigma]$ à $[\sigma + 1]$. On vérifie également facilement que, pour tout $d' \leq 1$, $\sigma + d'$ est dans une des r_k. ■

Le graphe de régions est défini de la manière suivante.

Définition 6.12 (graphe de régions)
Soit P un PCG, et c la constante définissant $\overset{c}{\sim}$. Le *graphe de régions* $G(P,c)$ est un graphe dont

- les éléments de l'ensemble de sommets S sont les régions appartenant à $\phi_P^{\Box}$;
- les éléments de l'ensemble E des arcs sont les couples $([\sigma],[\sigma'])$ tels que $\sigma \mapsto_P \sigma'$ (arc instantané) ou $[\sigma']$ est le successeur temporel de $[\sigma]$ (arc temporel).

□

Remarquer que

- le graphe de régions est fini et à branchement fini ;
- en tout sommet du graphe se trouve un arc instantané bouclant ;
- si (r,r') est un arc instantané de E, alors la proposition 6.11 garantit que pour tout état σ de r il existe un élément σ' de r' tel que $\sigma \mapsto_P \sigma'$.

Nous mettons en évidence une correspondance simple entre les chemins dans $G(P,c)$ et les séquences de pas de $\mathcal{S}_P$.

Définition 6.13 (r-pas)
Un r-pas (pas de régions) est un chemin fini $r_0 r_1 \ldots r_n$ de $G(P,c)$, avec $n \geq 1$, tel que, pour tout $0 \leq i < n-1$, (r_i, r_{i+1}) est un arc temporel de E, et (r_{n-1}, r_n) est un arc instantané de E.

□

Un r-pas correspond à un pas d'une séquence de $\mathcal{S}_P$:

Proposition 6.14
Pour tous σ et σ', $\sigma \rightsquigarrow_P \sigma'$ si et seulement si il existe un r-pas dans $G(P,c)$, dont le sommet initial (resp. final) est $[\sigma]$ (resp. $[\sigma']$), et tel que tous les états intermédiaires entre σ et σ' sont dans une des régions du r-pas.

□

Preuve. Nous montrons les deux sens de l'équivalence.

- $\Rightarrow$: Supposons $\sigma \rightsquigarrow_P \sigma'$. Il existe alors d tel que, pour tout $d' \leq d$, chaque état intermédiaire $\sigma + d'$ satisfait $\phi_P^{\Box}$, et $\sigma + d \mapsto_P \sigma'$.
  Soit δ la partie entière supérieure de d, et $\hbar$ le cardinal de $\mathcal{H}$. D'après la proposition 6.13, il existe $m \leq 2\delta\hbar$ et une séquence finie de régions $r_0 \ldots r_m$, telle que $r_0 = [\sigma]$, $r_m = [\sigma + d]$ et pour tout $i < m$, $r_{i+1} = \mathtt{succ}_t(r_i)$. De plus, pour tout $d' \leq d$ il existe i entre 1 et m telle que $r_i = [\sigma + d']$. La proposition 6.10 implique donc que les r_i sont dans $\phi_P^{\Box}$.
  Finalement, puisque $\sigma + d \mapsto_P \sigma'$, le couple $(r_m, [\sigma'])$ est un arc instantané de E. La séquence $r_0 \ldots r_m\,[\sigma']$ est donc bien un r-pas de $G(P,c)$ tel que tout état entre σ et σ' se trouve dans l'une de ses régions.
- $\Leftarrow$: immédiat par les définitions de $\mathtt{succ}_t$, E et des r-pas, et par la proposition 6.11. ■

La définition suivante applique l'opérateur $\propto$ aux ensembles de régions.

Définition 6.14
Soit R et R' deux ensembles de régions de $\mathcal{R}$. On définit $R \propto R'$ comme l'ensemble des régions r_0 telles qu'il existe un r-pas $r_0 r_1 \ldots r_n$ dans $G(P,c)$, avec $r_n \in R'$ et pour tout i dans $0, n-1$, $r_i \in R \cup R'$.

□

On déduit de cette définition et des propositions 6.9 et 6.10 le corollaire suivant.

Corollaire 6.15

Soit ϕ et ϕ' deux prédicats d'état (deux ensembles de régions) Pour tout état σ,

$$\sigma \models_{\mathcal{S}_P} \phi \propto \phi' \iff [\sigma] \in \phi \propto \phi'$$

□

6.4.2 Algorithme

L'algorithme de model checking de Tμ sur un PCG P consiste à calculer l'ensemble des états qui satisfont une formule φ de Tμ relativement à $\mathcal{S}_P$. Cet ensemble est représenté sous la forme d'un prédicat d'état $|\varphi|$, appelé *prédicat caractéristique* de φ. On note ici encore c la plus grande constante apparaissant dans P et φ. Dans cette section, tous les prédicats d'état ont toutes leurs constantes bornées par c.

Le prédicat caractéristique $|\varphi|$ est défini par induction de la manière suivante.

$$\begin{aligned}
|p| &\stackrel{\text{def}}{=} p \\
|x + n \leq y + m| &\stackrel{\text{def}}{=} x + n \leq y + m \\
|\neg\varphi| &\stackrel{\text{def}}{=} \neg|\varphi| \\
|\varphi_1 \vee \varphi_2| &\stackrel{\text{def}}{=} |\varphi_1| \vee |\varphi_2| \\
|x\downarrow\varphi| &\stackrel{\text{def}}{=} |\varphi|[x := 0] \\
|\varphi_1 \propto \varphi_2| &\stackrel{\text{def}}{=} |\varphi_1| \propto |\varphi_2| \\
|\mu X.\, \varphi(X)| &\stackrel{\text{def}}{=} \bigvee_{i\geq 0} \phi_i, \text{ où } \phi_0 \stackrel{\text{def}}{=} |\varphi(\texttt{false})| \text{ et } \phi_{i+1} \stackrel{\text{def}}{=} \phi_i \vee |\varphi(\phi_i)|
\end{aligned}$$

Cette définition fournit un algorithme de calcul de $|\varphi|$, à condition de savoir effectivement calculer $\phi_1 \propto \phi_2$ pour des prédicats d'état ϕ_1 et ϕ_2.

Le théorème suivant affirme que l'algorithme ainsi obtenu fonctionne correctement.

Théorème 6.16

Pour tout programme temporisé à commandes gardées P et pour toute formule φ de Tμ dont les plus petits points fixes sont définis sur des fonctionnelles monotones, l'algorithme termine et calcule un prédicat d'état ϕ tel que $[\![\phi]\!]_{\mathcal{S}_P} = [\![\varphi]\!]_{\mathcal{S}_P}$

□

Preuve. Nous montrons par induction sur la structure de φ que $|\varphi|$ est un prédicat d'état caractérisant un ensemble de régions, et que $[\![\varphi]\!]_{\mathcal{S}_P} = [\![|\varphi|]\!]_{\mathcal{S}_P}$.

La preuve est évidente pour $\varphi = p, x + n \leq y + m, \neg\varphi'$ et $\varphi_1 \vee \varphi_2$. Pour $\varphi = x\downarrow\varphi'$, on montre facilement que pour tout état σ et tout prédicat d'état ϕ,

$$\sigma[x := 0] \models_{\mathcal{S}_P} \phi \iff \sigma \models_{\mathcal{S}_P} \phi[x := 0]$$

Pour $\varphi = \varphi_1 \propto \varphi_2$ on fait appel au corollaire 6.15. Finalement, le théorème de Kleene assure que le calcul du plus petit point fixe est correct.

La terminaison de l'algorithme est garantie par le fait qu'il n'y a qu'un nombre fini de régions, et donc un nombre fini de prédicats d'états non équivalents. De plus, la finitude du nombre de prédicats d'état permet de ne pas exiger la continuité des fonctionnelles dont on calcule le plus petit point fixe. ■

Il reste donc à donner une méthode effective du calcul de $\phi_1 \propto \phi_2$.

Définition 6.15 (prédécesseur instantané)

Soit ϕ un prédicat d'état.

Pour une commande gardée $G = \psi \xrightarrow{\alpha} L$ de P, on définit le prédicat d'état

$$\mathtt{pre}_G(\phi) \stackrel{\text{def}}{=} \psi \wedge \phi_P^{\Box} \wedge (\phi \wedge \phi_P^{\Box})[L]$$

$\mathtt{pre}_G(\phi)$ caractérise les états à partir desquels un état de ϕ est accessible en exécutant la commande gardée G.

Le *prédécesseur instantané* de ϕ est le prédicat d'état

$$\mathtt{pre}_i(\phi) \stackrel{\text{def}}{=} (\phi \wedge \phi_P^{\Box}) \vee \bigvee_{G \in \mathcal{G}_P} \mathtt{pre}_G(\phi)$$

□

On vérifie aisément que $\mathtt{pre}_i(\phi)$ contient exactement les classes des états σ pour lesquels il existe σ' satisfaisant ϕ tel que $\sigma \mapsto_P \sigma'$.

Définition 6.16 (prédécesseur temporel)

Soit r une région. Le *prédécesseur temporel* de r est le prédicat d'état $\mathtt{pre}_t(r)$ caractérisant la région r' telle que $r = \mathtt{succ}_t(r')$ (ou `false` si r' n'existe pas).

Soit ϕ un prédicat d'état. Le prédécesseur temporel $\mathtt{pre}_t(\phi)$ de ϕ est la disjonction des $\mathtt{pre}_t(r)$, pour toutes les régions r de ϕ.

□

Le prédécesseur temporel d'une région r peut être calculé de la manière suivante à partir du prédicat ϕ_r.

Supposons que ϕ_r est la conjonction des prédicats des huit ensembles

$$\begin{aligned}
E_1 &\stackrel{\text{def}}{=} \{p_b,\ p_b \in \Pi,\ b \in B\} \\
E_2 &\stackrel{\text{def}}{=} \{\neg p_d,\ p_d \in \Pi,\ d \in D\} \\
E_3 &\stackrel{\text{def}}{=} \{x_i = n_i,\ i \in I,\ n_i \leq c\} \\
E_4 &\stackrel{\text{def}}{=} \{n_j < x_j < n_j + 1,\ j \in J,\ n_j < c\} \\
E_5 &\stackrel{\text{def}}{=} \{x_k > c,\ k \in K\} \\
E_6 &\stackrel{\text{def}}{=} \{x_t = x_u + n_{tu},\ (t,u) \in TU,\ n_{tu} \leq c\} \\
E_7 &\stackrel{\text{def}}{=} \{x_v + n_{vw} < x_w < x_v + n_{vw} + 1,\ (v,w) \in VW,\ n_{vw} < c\} \\
E_8 &\stackrel{\text{def}}{=} \{x_e > x_f + c,\ (e,f) \in EF\}
\end{aligned}$$

Un ensemble de prédicats d'état est considéré comme le prédicat conjonction de ses éléments.

Nous distinguons trois cas, selon que I est vide ou non, et dans ce second cas selon qu'un n_i est nul ou non.

1. Si I est non vide et si un n_i est nul, alors $\mathtt{pre}_t(r) = \mathtt{false}$. En effet, ϕ_r ne possède pas de prédécesseur temporel puisqu'au moins une horloge est nulle.
2. Si I est non vide et si tous les n_i sont strictement positifs, alors dès qu'on "recule" d'une infime fraction de temps, les x_i deviennent non entiers, et leur partie fractionnelle (identique pour chacun d'entre eux) est alors supérieure à celle de chacun des x_j et des x_k.

   $\mathtt{pre}_t(r)$ est donc la conjonction de E_1, E_2, $E_4, \ldots, E_8$ et de

   $$\{n_i - 1 < x_i < n_i,\ i \in I\}$$

3. Si I est vide, alors "reculer" d'une région revient à rendre entiers certains x_k et/ou les x_j dont la partie fractionnelle est la plus petite.

   Pour $k \in K$, notons $n_k = c$, et pour tout $H \subseteq J \cup K$, soit

$$\begin{aligned} E_3^H &\stackrel{\text{def}}{=} \{x_h = n_h,\, h \in H\} \\ E_4^H &\stackrel{\text{def}}{=} E_4 - \{n_h < x_h < n_h + 1,\, h \in H\} \\ E_5^H &\stackrel{\text{def}}{=} E_5 - \{x_h > c,\, h \in H\} \end{aligned}$$

   Le prédicat $\mathtt{pre}_t(r)$ est alors

$$\bigvee_{\substack{H \subseteq J \cup K \\ H \neq \emptyset}} \left(E_1 \wedge E_2 \wedge E_3^H \wedge E_4^H \wedge E_5^H \wedge E_6 \wedge E_7 \wedge E_8 \right)$$

Il n'existe qu'un seul sous-ensemble H pour lequel le prédicat entre parenthèses n'est pas réduit à `false`. En effet, pour tous les autres, E_3^H est incompatible avec $E_6 \wedge E_7 \wedge E_8$.

Définition 6.17

Soit ϕ et ϕ' deux prédicats d'état, et soit n un entier naturel. Le prédicat d'état $\phi \rightarrow^n \phi'$ est défini par induction sur n par :

$$\begin{aligned} \phi \rightarrow^0 \phi' &\stackrel{\text{def}}{=} \phi' \wedge \phi_P^{\Box} \\ \phi \rightarrow^{n+1} \phi' &\stackrel{\text{def}}{=} \phi \wedge \phi_P^{\Box} \wedge \mathtt{pre}_t(\phi \rightarrow^n \phi') \end{aligned}$$

□

Les prédicats $\phi \rightarrow^n \phi'$ sont calculables et sont en nombre fini (modulo l'équivalence des prédicats d'état). Nous pouvons alors définir et calculer le prédicat d'état

$$\phi \rightarrow \phi' \stackrel{\text{def}}{=} \bigvee_{n \in \mathbb{N}} \phi \rightarrow^n \phi'$$

Un état σ est donc dans $\phi \rightarrow \phi'$ s'il est possible, à partir de σ, d'atteindre un état de ϕ' en laissant passer le temps et en restant continuellement dans $\phi_P^{\Box} \wedge (\phi \vee \phi')$.

Les transformateurs de prédicats $\mathtt{pre}_i$ et $\rightarrow$ permettent de calculer la relation $\propto$ sur les prédicats d'état :

Proposition 6.17

Pour tous prédicats d'état ϕ et ϕ', $\phi \propto \phi' = \phi \rightarrow \mathtt{pre}_i(\phi')$

□

Nous disposons donc en théorie de tout l'outillage nécessaire pour appliquer l'algorithme de model checking. Sa mise en œuvre nécessite bien entendu de pouvoir effectuer les calculs rapidement, aussi bien pour les transformateurs de prédicats que pour les procédures de décision d'inclusion ou d'égalité de prédicats d'état.

6.4.3 Model checking pour TCTL

Nous montrons à présent comment étendre la notion de prédicat caractéristique aux formules de TCTL, c'est-à-dire aux opérateurs $\exists\mathcal{U}$ et $\forall\mathcal{U}$.

En premier lieu, remarquer que le prédicat caractéristique de la formule EdE est exactement $\phi_P^{\Box}$.

6.4.3.1 Possibilité

La proposition 6.6 permet de définir directement le prédicat caractéristique de $\phi_1 \exists\mathcal{U} \phi_2$ en fonction de ϕ_1 et ϕ_2 :

Proposition 6.18

Pour tous prédicats caractéristiques ϕ_1 et ϕ_2 de TCTL,

$$|\phi_1 \exists\mathcal{U} \phi_2| = \phi_P^{\Box} \wedge |\mu X.\,(\phi_2 \vee (\phi_1 \propto X))|$$

□

6.4.3.2 Inévitabilité bornée

Nous montrons à présent que l'inévitabilité *bornée* s'exprime en termes de l'opérateur $\exists\mathcal{U}$ *si on se restreint aux séquences de pas divergentes de* P.

Proposition 6.19

Soit φ_1 et φ_2 deux formules de TCTL ; soit z une horloge n'appartenant pas à $\mathcal{H}_P$ et non libre dans φ_1 ou φ_2 ; soit n un entier naturel. Alors

$$[\![z{\downarrow}\, \varphi_1 \forall\mathcal{U} (\varphi_2 \wedge z \leq n)]\!]_{\mathcal{S}_P^{\Delta}} = [\![\neg z{\downarrow}\, \neg\varphi_2 \exists\mathcal{U} (\neg(\varphi_1 \vee \varphi_2) \vee z > n)]\!]_{\mathcal{S}_P^{\Delta}}$$

□

En français, cela signifie que la propriété

> "il est inévitable de satisfaire φ_2 avant n unités de temps, $\varphi_1 \vee \varphi_2$ étant auparavant continûment satisfaite"

est équivalente à la propriété

> "il n'est pas possible de ne satisfaire ni φ_1 ni φ_2 ou de dépasser l'instant n, sans avoir précédemment satisfait φ_2 avant l'instant n."

Preuve. Soit K_1 le premier ensemble caractéristique ci-dessus, et K_2 le second. Nous montrons que $K_1 \subseteq K_2$ puis que $K_2 \subseteq K_1$.

Soit σ dans K_1. Pour toute séquence $\overline{\sigma}$ de $\mathcal{S}_P^{\Delta}$ dont l'état initial est $\sigma[z := 0]$, il existe γ dans $,^{\overline{\sigma}}$ dont l'état satisfait $\varphi_2 \wedge z \leq n$, et de plus l'état de toute occurrence $\gamma' \leq \gamma$ satisfait $\varphi_1 \vee (\varphi_2 \wedge x \leq n)$. Puisque $\gamma' \leq \gamma$ et que z n'est pas une horloge de P, on est assuré que ${\gamma'}^{\mathsf{e}}(z) \leq n$. Tous les γ' sont dont tels que ${\gamma'}^{\mathsf{e}}$ satisfait $(\varphi_1 \vee \varphi_2) \wedge z \leq n$.

Supposons que σ n'appartient pas à K_2. Il existe alors une séquence $\overline{\sigma}$ dans $\mathcal{S}_P^{\Delta}$ dont l'état initial est $\sigma[z : 0]$, et il existe une occurrence γ_1 dans $,^{\overline{\sigma}}$ dont l'état satisfait $\neg(\varphi_1 \vee \varphi_2) \vee z > n$, et de plus toute occurrence $\gamma_1' \leq \gamma_1$ est telle que ${\gamma_1'}^{\mathsf{e}}$ satisfait $\neg\varphi_2 \vee \neg(\varphi_1 \vee \varphi_2) \vee z > n$; c'est-à-dire :

$$\forall \gamma_1' \leq \gamma_1 : {\gamma_1}^{\mathsf{e}}(z) \leq n \Rightarrow {\gamma_1}^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_2$$

Pour cette séquence σ particulière, on a soit $\gamma \leq \gamma_1$, soit $\gamma_1 \leq \gamma$.

Le premier cas est impossible, car alors γ est un γ_1', et on devrait avoir $\gamma^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_2$, ce qui est faux.

Le second cas est également impossible. En effet γ_1 serait alors un γ'. Son état devrait donc satisfaire $\varphi_1 \vee \varphi_2$ et $z \leq n$, ce qui est également faux.

On aboutit à une contradiction, et donc σ appartient à K_2.

Soit σ dans K_2. Pour toute séquence $\overline{\sigma}$ de $\mathcal{S}_P^{\Delta}$ dont l'état initial est $\sigma[z := 0]$, pour tout γ dans $,^{\overline{\sigma}}$, γ^{e} satisfait $(\varphi_1 \vee \varphi_2) \wedge z \leq n$ ou il existe $\gamma' \leq \gamma$ dont l'état satisfait $\varphi_2 \wedge (\varphi_1 \vee \varphi_2) \wedge z \leq n$, qui est équivalent à $\varphi_2 \wedge z \leq n$. On a donc :

$$\begin{array}{l} \forall \overline{\sigma} \in \mathcal{S}_P^{\Delta} \text{ avec } \sigma_0 = \sigma[z := 0], \\ \forall \gamma \in ,^{\overline{\sigma}} : (\gamma^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} (\varphi_1 \vee \varphi_2) \wedge z \leq n) \ \vee \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \varphi_2 \wedge z \leq n \end{array} \tag{6.2}$$

Supposons que σ n'est pas dans K_1. Il existe alors une séquence $\overline{\sigma}$ dans $\mathcal{S}_P^{\Delta}$, dont l'état initial est $\sigma_0 = \sigma[z := 0]$, telle que

$$\forall \gamma \in ,^{\overline{\sigma}} : (\gamma^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_2 \vee z > n) \ \vee \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_1 \wedge (\neg\varphi_2 \vee z > n) \tag{6.3}$$

Par définition, cette séquence $\overline{\sigma}$ est divergente. La valeur de z n'étant jamais remise à zéro, il existe donc une occurrence γ dans $,^{\overline{\sigma}}$ telle que $\gamma^{\mathsf{e}}(z) > n$. En appliquant la formule 6.2 à γ, on trouve qu'il existe $\gamma_0 \leq \gamma$ telle que

$$\gamma_0{}^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \varphi_2 \wedge z \leq n$$

On peut déduire des formules 6.2 et 6.3 les formules 6.4 et 6.5 suivantes :

$$\forall \gamma \in ,^{\overline{\sigma}} : \left(\gamma^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_1 \wedge \neg\varphi_1 \wedge z \leq n \ \Rightarrow \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \varphi_2 \wedge z \leq n\right) \tag{6.4}$$

$$\forall \gamma \in ,^{\overline{\sigma}} : \left(\gamma^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \varphi_2 \wedge z \leq n \ \Rightarrow \ \exists \gamma' \leq \gamma : \gamma'^{\mathsf{e}} \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_1 \wedge \neg\varphi_2 \wedge z \leq n\right) \tag{6.5}$$

Supposons que l'indice de γ_0 est k. L'état de γ_0 se trouve donc dans un des $k+1$ premiers pas de $\overline{\sigma}$.

D'après la proposition 6.14, il existe $k+1$ r-pas dans le graphe de régions, chacun d'entre eux correspondant à un des $k+1$ premiers pas de $\overline{\sigma}$.

On peut partitionner l'ensemble de toutes les régions en trois sous-ensembles :

$$\begin{array}{rcl} R_0 & \stackrel{\text{def}}{=} & \{r \mid \forall \sigma \in r : \sigma \models_{\mathcal{S}_P^{\Delta}} \varphi_2\} \\ R_1 & \stackrel{\text{def}}{=} & \{r \mid \forall \sigma \in r : \sigma \models_{\mathcal{S}_P^{\Delta}} \neg\varphi_1 \wedge \neg\varphi_2\} \\ R_2 & \stackrel{\text{def}}{=} & \{r \mid \forall \sigma \in r : \sigma \models_{\mathcal{S}_P^{\Delta}} \varphi_1 \wedge \neg\varphi_2\} \end{array}$$

(le théorème 6.12 garantit que les R_i sont bien des ensembles de régions.)

L'état de γ_0 se trouve dans une région de R_0. Il existe donc des régions de $R_0 \cup R_1$ dans la séquence des k_1 r-pas. À partir de σ_0 et en avançant le long de ces r-pas, il existe donc nécessairement une région r dans R_0 ou R_1 telle que toutes les régions précédentes sont dans R_2. Si r est dans R_0 (resp. R_1), alors tout état de r viole la propriété 6.5 (resp. 6.4).

On obtient donc une contradiction ; par conséquent σ est bien dans K_1. ■

Nous exhibons à présent un contre-exemple de cette proposition dans le cas des systèmes temporels quelconques.

On considère $\Pi = \{p, q\}$, $\mathcal{H} = \{x, y\}$ et $D \stackrel{\text{def}}{=} \mathbb{Q}_+$.

Un état σ est un quadruplet $(\sigma(p), \sigma(q), \sigma(x), \sigma(y))$.

Soit les formules de TCTL

$$\begin{array}{lll} \varphi_1 \stackrel{\text{def}}{=} \exists \Diamond_{\leq 1} p & & \varphi_2 \stackrel{\text{def}}{=} \exists \Diamond_{\leq 1} q \\ \varphi \stackrel{\text{def}}{=} z{\downarrow}\, \varphi_1 \forall \mathcal{U} (\varphi_2 \wedge z \leq 1) & \text{et} & \varphi' \stackrel{\text{def}}{=} \neg(z{\downarrow}\, \neg\varphi_2 \exists \mathcal{U} \neg(\varphi_1 \vee \varphi_2) \vee z > 1) \end{array}$$

Nous définissons le système temporel T_1 de la manière suivante.

Pour tout σ, $\sigma \xrightarrow{d}_{T_1} \sigma + d$, c'est-à-dire que le temps n'est jamais bloqué.

Soit $\sigma_0 = (\mathtt{ff}, \mathtt{ff}, 0, 0)$ et, pour tout d dans $\mathbb{Q}_+$, $\sigma_d = \sigma_0 + d$. Les seules transitions instantanées sont :

$$t_1 \ : \ \sigma_0 \mapsto_{T_1} (\mathtt{tt}, \mathtt{ff}, 0, 0) \qquad \forall n \in \mathbb{N}^* \ : \ t_1^n \ : \ \sigma_{\frac{1}{n}} \mapsto_{T_1} (\mathtt{ff}, \mathtt{ff}, \tfrac{1}{n}, 0)$$

$$t_2 \ : \ (\mathtt{tt}, \mathtt{ff}, 2, 2) \mapsto_{T_1} (\mathtt{tt}, \mathtt{tt}, 2, 2) \qquad \forall n \in \mathbb{N}^* \ : \ t_2^n \ : \ (\mathtt{ff}, \mathtt{ff}, 1 + \tfrac{1}{n}, 1) \mapsto_{T_1} (\mathtt{ff}, \mathtt{tt}, 1 + \tfrac{1}{n}, 1)$$

La transition t_1 garantit que $\sigma_0 \models_{\mathcal{S}^{\Delta}_{T_1}} \varphi_1$. De même, les transitions t_1^n et t_2^n permettent de déduire

$$\forall n \in \mathbb{N}^* \ : \ \sigma_{\frac{1}{n}} \models_{\mathcal{S}^{\Delta}_{T_1}} \varphi_2$$

De plus, pour tout $d > 0$, aucune exécution à partir de σ_d ne permet d'atteindre un état où p est vraie. Donc

$$\forall d > 0 \ : \ \sigma_d \models_{\mathcal{S}^{\Delta}_{T_1}} \neg\varphi_1$$

Finalement, les seuls s_d à partir desquels on peut atteindre un état satisfaisant q dans un délai ≤ 1 sont les $\sigma_{\frac{1}{n}}$. Par conséquent, on obtient :

$$\begin{array}{lll} \sigma_0 & \models_{\mathcal{S}^{\Delta}_{T_1}} & \varphi_1 \wedge \neg\varphi_2 \\ \sigma_{\frac{1}{n}} & \models_{\mathcal{S}^{\Delta}_{T_1}} & \varphi_2 \qquad \forall n \in \mathbb{N}^* \\ \sigma_d & \models_{\mathcal{S}^{\Delta}_{T_1}} & \neg(\varphi_1 \vee \neg\varphi_2) \qquad \forall d \notin \{0\} \cup \{\frac{1}{n} \mid n \in \mathbb{N}^*\} \end{array}$$

Il existe trois types de séquences de pas divergentes dont l'état initial est σ_0 :

1. les séquences qui "commencent" par la transition instantanée t_1. Celles-ci ne passent par aucun des σ_d pour $d > 0$;
2. les séquences qui, lorsqu'elles atteignent un certain état $\sigma_{\frac{1}{n}}$, continuent en prenant la transition t_1^n ;
3. les séquences qui passent par tous les σ_d.

Pour les séquences de type 1, l'état $(\mathtt{tt}, \mathtt{ff}, 1, 1)$ est atteint après une unité de temps, et il satisfait φ_2. De plus, tous les états qui de ces séquences satisfont φ_1. Le long de ces séquences, il n'est donc pas possible d'atteindre un état satisfaisant $\neg(\varphi_1 \vee \varphi_2) \vee z > 1$ sans avoir précédemment satisfait φ_2.

Pour les séquences des deux autres types, tout état satisfaisant $\neg\varphi_1 \wedge \neg\varphi_2$ ou tel que $z > 1$ est précédé d'un état satisfaisant φ_2 car, d'une part l'état initial de ces séquences est σ_0 qui ne satisfait pas $\neg\varphi_1 \wedge \neg\varphi_2$, et d'autre part

$$\forall d \in \mathbb{Q}_+^* \ \exists n \in \mathbb{N}^* \ : \ \frac{1}{n} < d$$

Pour ces séquences, il n'est donc pas non plus possible d'atteindre un état satisfaisant $\neg(\varphi_1 \vee \varphi_2) \vee z > 1$ sans avoir précédemment satisfait φ_2.

On en conclut que l'état σ_0 satisfait φ'. Par contre il ne satisfait pas φ. En effet, le long des séquences de type 2 ou 3, il n'est pas possible d'atteindre un état satisfaisant φ_2 (un $\sigma_{\frac{1}{n}}$) sans avoir précédemment satisfait $\neg\varphi_1 \vee \varphi_2$, car

$$\forall n \in \mathbb{N}^* \ \exists d \in \mathbb{Q}_+^* \ : \ d < \frac{1}{n}$$

On a donc trouvé un système temporel T_1 tel que

$$[\![z\downarrow \varphi_1 \forall\mathcal{U}(\varphi_2 \wedge z \leq 1)]\!]_{\mathcal{S}_P^\Delta} \neq [\![\neg(z\downarrow \neg\varphi_2 \exists\mathcal{U}\neg(\varphi_1 \vee \varphi_2) \vee z > 1)]\!]_{\mathcal{S}_{T_1}^\Delta}$$

Si la conjecture 6.8 est fausse, alors il doit exister ψ dans Tμ telle que

$$\forall \text{ système temporel } T \ : \ [\![\varphi]\!]_{\mathcal{S}_T^\Delta} = [\![\psi]\!]_{\mathcal{S}_T^\Delta}$$

D'autre part, la proposition 6.6 garantit que φ' est équivalente à une formule ψ' de Tμ. La proposition 6.19 affirme alors que

$$\forall \text{GCP} P \ : \ [\![\varphi]\!]_{\mathcal{S}_P^\Delta} = [\![\psi']\!]_{\mathcal{S}_P^\Delta}$$

Pour réfuter la conjecture, il faudrait donc trouver une formules ψ' de Tμ qui "coïncide" avec ψ sur *tous* les GCP, mais pas sur certains systèmes temporels, en particulier sur T_1 défini ci-dessus, c'est-à-dire telle que

$$\forall \text{GCP} P \ : \ [\![\psi]\!]_{\mathcal{S}_P^\Delta} = [\![\psi']\!]_{\mathcal{S}_P^\Delta} \quad \text{et} \quad [\![\psi]\!]_{\mathcal{S}_{T_1}^\Delta} \neq [\![\psi']\!]_{\mathcal{S}_{T_1}^\Delta}$$

Une telle éventualité semble hautement improbable ; c'est pourquoi nous pouvons raisonnablement espérer que la conjecture est vraie.

Soit P un PCG bien temporisé. Lorsqu'on souhaite vérifier une propriété d'inévitabilité, il faut bien entendu le faire en ne considérant que les séquences de pas divergentes de P (voir proposition 6.4). En d'autres termes, on ne s'intéresse pour les inévitabilités qu'aux ensembles caractéristiques relatif aux séquences de pas divergentes de P. Le prédicat caractéristique de $\varphi_1 \forall\mathcal{U} \varphi_2$ doit donc être défini de telle sorte que

$$[\![\varphi_1 \forall\mathcal{U} \varphi_2]\!]_{\mathcal{S}_P^\Delta} = [\![|\varphi_1 \forall\mathcal{U} \varphi_2|]\!]_{\mathcal{S}_P^\Delta}$$

La proposition 6.3 affirme que si P est bien temporisé, alors pour toute formule φ de Tμ, $[\![\varphi]\!]_{\mathcal{S}_P^\Delta} = [\![\varphi]\!]_{\mathcal{S}_P}$. La proposition 6.19 permet donc de définir le prédicat caractéristique de l'inévitabilité bornée :

Proposition 6.20

Pour tout PCG P, pour toutes formules φ_1 et φ_2 de TCTL, pour toute horloge z n'apparaissant pas dans P et non libre dans φ_1 ou φ_2, pour tout entier positif n, si P est bien temporisé alors

$$|z\downarrow \varphi_1 \forall\mathcal{U}(\varphi_2 \wedge z \leq n)| = |\neg z\downarrow \neg\varphi_2 \exists\mathcal{U}(\neg(\varphi_1 \vee \varphi_2) \vee z > n)|$$

ou de manière équivalente, d'après la proposition 6.18 :

$$|z\downarrow \varphi_1 \forall\mathcal{U}(\varphi_2 \wedge z \leq n)| = |z\downarrow \neg\,(\phi_P^{\Box} \wedge \mu X.\,(\neg(|\varphi_1| \vee |\varphi_2|) \vee z > n \vee (\neg|\varphi_2| \propto X)))|$$

□

6.4.3.3 Inévitabilité

Pour obtenir le prédicat caractéristique de l'inévitabilité générale $\varphi_1 \forall\mathcal{U} \varphi_2$, on fait alors appel à la proposition 6.7, qui affirme que l'ensemble caractéristique de $\varphi_1 \forall\mathcal{U} \varphi_2$ relativement aux séquences divergentes est le plus petit point fixe d'une fonctionnelle où apparaît une inévitabilité bornée. Si on remplace celle-ci par son prédicat caractéristique (qu'on sait calculer grâce à la proposition 6.20), on obtient alors la proposition suivante, qui définit le prédicat caractéristique de l'inévitabilité en fonction de celui d'une formule de Tμ.

Proposition 6.21

Pour tout PCG P, pour toutes formules φ_1 et φ_2 de TCTL, pour toute horloge z n'apparaissant pas dans P et non libre dans φ_1 ou φ_2, pour tout entier strictement positif n, si P est bien temporisé alors

$$|\varphi_1 \forall\mathcal{U} \varphi_2| = |\mu X.\ (|\varphi_2| \vee z{\downarrow}\,(|\varphi_1| \forall\mathcal{U} (X \wedge z \leq n)))|$$

On peut obtenir une formule de Tμ en utilisant la caractérisation de l'inévitabilité bornée présentée ci-dessus :

$$|\varphi_1 \forall\mathcal{U} \varphi_2| = |\mu X.\ (|\varphi_2| \vee z{\downarrow}\,\neg(\phi_P^{\Box} \wedge \mu Y.\ (\neg(|\varphi_1| \ \vee\ X) \ \vee\ z > n \vee (\neg X \propto Y))))|$$

□

Le calcul de $|\varphi_1 \forall\mathcal{U} \varphi_2|$ consiste donc à itérer celui de $\mu X.\, \varphi_2 \vee \varphi_1 \forall\mathcal{U}_{\leq n}\, X$. On constate une correspondance avec le cas non temporisé, pour lequel on itère $\mu X.\, \varphi_2 \vee \boxed{\varphi_1 \wedge \bigcirc X \wedge \neg\bigcirc\neg X}$. La partie encadrée de cette fonctionnelle exprime une inévitabilité "à l'état suivant", φ_1 étant vérifiée "à l'état courant". Ne disposant pas de notion d'état suivant, nous faisons correspondre à cette sous-formule l'inévitabilité bornée $\varphi_1 \forall\mathcal{U}_{\leq n}\, X$, dont le prédicat caractéristique est, lui, directement calculable comme la négation d'un $\exists\mathcal{U}$.

La valeur choisie pour n est sans influence sur le résultat final ; elle peut par contre affecter le nombre d'itérations nécessaires pour atteindre le point fixe. Plus elle est grande, moins il est nécessaire d'itérer pour le calcul de $|\mu X.\, \varphi_2 \vee \varphi_1 \forall\mathcal{U}_{\leq n}\, X|$. Cependant, le calcul de chacun des $|\varphi_1 \forall\mathcal{U}_{\leq n}\, \phi_i|$ nécessite également une itération. Celle-ci est en générale plus longue et compliquée lorsque n croît. La valeur optimale de n peut être très difficile à trouver. Choisir $n = 1$ facilite en général les calculs, même s'ils sont plus longs, car la taille des ensembles de régions est alors plus petite qu'avec une valeur plus importante.

6.4.4 Détection des programmes bien temporisés

Il est important à double titre de pouvoir détecter si un PCG est bien temporisé : d'une part à cause des arguments exposés dans la section 2.5 du chapitre 2, et d'autre part pour pouvoir vérifier les propriétés d'inévitabilité de TCTL, comme on vient de le montrer.

D'après la définition 6.5, un PCG n'est pas bien temporisé s'il existe une séquence de $\mathcal{S}_P$ dont un préfixe fini n'est pas préfixe d'une séquence de pas divergente. Il est clair qu'il existe alors un état σ qui satisfait $\phi_P^{\Box}$, à partir duquel toute exécution ne dépasse pas une unité de temps.

Un PCG P est donc bien temporisé si et seulement si l'ensemble caractéristique de $\phi_P^{\Box}$ est inclus dans celui de $\exists\Diamond_{=1}\,\mathtt{true}$, c'est-à-dire si le prédicat d'état

$$\phi_P^{\Box} \Rightarrow |\exists\Diamond_{=1}\,\mathtt{true}|$$

est valide. Ce prédicat peut être écrit en utilisant Tμ :

$$\phi_P^{\Box} \Rightarrow |z{\downarrow}\,\mu X.\, z = 1 \vee \mathtt{true} \propto X|$$

Nous notons ϕ_{bt} le prédicat d'état $|z{\downarrow}\,\mu X.\, z = 1 \vee \mathtt{true} \propto X|$.

À titre d'exemple, nous vérifions que les programmes P_1 et P_2 présentés dans la section 6.3.2 sont respectivement bien temporisé et non bien temporisé. Nous nous plaçons dans le cas $D = \mathbb{Q}_+$.

Soit $\varphi(X)$ la formule $z = 1 \vee \mathtt{true} \propto X$.

Le programme P_1 est défini par

$$\begin{array}{rcl} H_1 & \stackrel{\text{def}}{=} & \{x\} \\ \phi_1^0 & \stackrel{\text{def}}{=} & (s=0) \wedge (x=0) \\ \phi_1^\Box & \stackrel{\text{def}}{=} & ((s=0) \wedge (x \leq 4)) \vee ((s=1) \wedge (x \leq 4)) \vee (s=2) \\ \mathcal{G}_1 & \stackrel{\text{def}}{=} & \left\{ s=0 \xrightarrow{a} s:=1 \ , \ s=1 \xrightarrow{b} s:=0 \ , \ (s=0) \wedge (x \geq 1) \xrightarrow{c} s:=2 \right\} \end{array}$$

Les trois commandes gardées sont appelées, dans l'ordre, G_1, G_2 et G_3.

Nous devons en premier lieu calculer $|\mu X.\varphi(X)|$.

La première itération du calcul du point fixe donne

$$\phi_0 \stackrel{\text{def}}{=} |\varphi(\texttt{false})| = z = 1$$

On doit donc calculer

$$\phi_1 \stackrel{\text{def}}{=} |\varphi(\phi_0) = (z=1) \vee \texttt{true} \propto (z=1)$$

Il faut pour cela connaître $\texttt{pre}_{G_i}(z=1)$ pour chaque commande G_i.

$$\begin{array}{rcl} \texttt{pre}_{G_1}(z=1) & = & (s=0) \wedge \phi_1^\Box \wedge ((z=1) \wedge \phi_1^\Box)[s:=1] \\ & = & (s=0) \wedge (x \leq 4) \wedge (z=1) \wedge ((1=1) \wedge (x \leq 4)) \\ & = & (s=0) \wedge (x \leq 4) \wedge (z=1) \\ \texttt{pre}_{G_2}(z=1) & = & (s=1) \wedge \phi_1^\Box \wedge ((z=1) \wedge \phi_1^\Box)[s:=0] \\ & = & (s=1) \wedge (x \leq 4) \wedge (z=1) \wedge ((0=0) \wedge (x \leq 4)) \\ & = & (s=1) \wedge (x \leq 4) \wedge (z=1) \\ \texttt{pre}_{G_3}(z=1) & = & (s=0) \wedge (x \geq 1) \wedge \phi_1^\Box \wedge ((z=1) \wedge \phi_1^\Box)[s:=2] \\ & = & (s=0) \wedge (1 \leq x \leq 4) \wedge (z=1) \wedge (2=2) \\ & = & (s=0) \wedge (1 \leq x \leq 4) \wedge (z=1) \end{array}$$

Nous obtenons alors

$$\begin{array}{rcl} \texttt{pre}_i(z=1) & = & (z=1) \wedge \phi_1^\Box \vee \texttt{pre}_{G_1}(z=1) \vee \texttt{pre}_{G_2}(z=1) \vee \texttt{pre}_{G_3}(z=1) \\ & = & (z=1) \wedge \phi_1^\Box \end{array}$$

On trouve ensuite

$$\begin{array}{rcl} |\texttt{true} \propto (z=1)| & = & \texttt{true} \rightharpoonup \texttt{pre}_i(z=1) \\ & = & (z \leq 1) \wedge (\ (s=0) \wedge (x \leq 4) \wedge (x \leq z+3) \ \vee \\ & & \qquad (s=1) \wedge (x \leq 4) \wedge (x \leq z+3) \ \vee \\ & & \qquad (s=2)) \end{array}$$

On en déduit

$$\begin{array}{rcl} \phi_1 = (z=1) \ \vee \ (z \leq 1) \wedge (\ (s=0) \wedge (x \leq 4) \wedge (x \leq z+3) \ \vee \\ \qquad (s=1) \wedge (x \leq 4) \wedge (x \leq z+3) \ \vee \\ \qquad (s=2)) \end{array}$$

On doit à présent calculer les $\mathtt{pre}_{G_i}(\phi_1)$:

$$\begin{aligned}
\mathtt{pre}_{G_1}(\phi_1) &= \mathtt{pre}_{G_1}(z=1) \ \vee \ (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \\
&= (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \\
\mathtt{pre}_{G_2}(\phi_1) &= \mathtt{pre}_{G_2}(z=1) \ \vee \ (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \\
&= (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \\
\mathtt{pre}_{G_3}(\phi_1) &= \mathtt{pre}_{G_3}(z=1) \ \vee \ (s=0) \wedge (1 \leq x \leq 4) \wedge (z \leq 1) \\
&= (s=0) \wedge (1 \leq x \leq 4) \wedge (z \leq 1)
\end{aligned}$$

On a alors

$$\begin{aligned}
\mathtt{pre}_i(\phi_1) &= \phi_1 \wedge \phi_1^{\Box} \ \vee \ \mathtt{pre}_{G_1}(\phi_1) \ \vee \ \mathtt{pre}_{G_2}(\phi_1) \ \vee \ \mathtt{pre}_{G_3}(\phi_1) \\
&= (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \ \vee \\
&\quad (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \ \vee \\
&\quad (s=2) \wedge (z \leq 1)
\end{aligned}$$

Ce qui donne

$$\begin{aligned}
|\mathtt{true} \propto \phi_1| &= \mathtt{tt} \rightarrow \mathtt{pre}_i(\phi_1) \\
&= (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \ \vee \\
&\quad (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \ \vee \\
&\quad (s=2) \wedge (z \leq 1) \\
\phi_2 &= (z=1) \ \vee \\
&\quad (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \ \vee \\
&\quad (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \ \vee \\
&\quad (s=2) \wedge (z \leq 1)
\end{aligned}$$

Nous passons à la troisième itération.

$$\begin{aligned}
\mathtt{pre}_{G_1}(\phi_2) &= (s=0) \wedge (x \leq 4) \wedge (z \leq 1) \wedge (x \leq z+3) \\
\mathtt{pre}_{G_2}(\phi_2) &= (s=1) \wedge (x \leq 4) \wedge (z \leq 1) \\
\mathtt{pre}_{G_3}(\phi_2) &= (s=0) \wedge (1 \leq x \leq 4) \wedge (z \leq 1) \\
\mathtt{pre}_i(\phi_2) &= \phi_2 \wedge \phi_1^{\Box} \ \vee \ \mathtt{pre}_{G_1}(\phi_2) \ \vee \ \mathtt{pre}_{G_2}(\phi_2) \ \vee \ \mathtt{pre}_{G_3}(\phi_2) \\
&= (z \leq 1) \ \wedge \ ((s=0) \wedge (x \leq 4) \ ou \ (s=1) \wedge (x \leq 4) \ \vee \ (s=2)) \\
&= (z \leq 1) \wedge \phi_1^{\Box} \\
|\mathtt{true} \propto \phi_2| &= \mathtt{tt} \rightarrow \mathtt{pre}_i(\phi 2) \\
&= (z \leq 1) \wedge \phi_1^{\Box} \\
\phi_2 &= (z=1) \ \vee \ (z \leq 1) \wedge \phi_1^{\Box}
\end{aligned}$$

L'itération suivante donne $\phi_3 = \phi_2$. On obtient donc

$$\begin{aligned}
|\mu X.\varphi(X)| &= \phi_0 \ \vee \ \phi_1 \ \vee \ \phi_2 \\
&= (z=1) \ \vee \ (z \leq 1) \wedge \phi_1^{\Box}
\end{aligned}$$

Par conséquent,

$$\begin{aligned}\phi_{bt} &= |\mu X.\,\varphi(X)|[z := 0] \\ &= \texttt{false} \vee \texttt{true} \wedge \phi_1^{\Box} \\ &= \phi_1^{\Box}\end{aligned}$$

Le programme P_1 est donc bien temporisé, puisque $\phi_1^{\Box} \Rightarrow \phi_{bt}$ est trivialement valide.

Le programme P_2 est défini par

$$\begin{aligned}H_2 &\stackrel{\text{def}}{=} \{x\} \\ \phi_2^0 &\stackrel{\text{def}}{=} (s = 0) \wedge (x = 0) \\ \phi_2^{\Box} &\stackrel{\text{def}}{=} ((s = 0) \wedge (x \leq 5)) \vee ((s = 1) \wedge (x \leq 5)) \vee (s = 2) \\ \mathcal{G}_2 &\stackrel{\text{def}}{=} \left\{ s = 0 \xrightarrow{a} s := 1 \;,\; s = 1 \xrightarrow{b} s := 0 \;,\; (s = 0) \wedge (1 \leq x \leq 4) \xrightarrow{c} s := 2 \right\}\end{aligned}$$

Les trois commandes sont appelées G_1', G_2' et G_3'. Il faut effectuer le même calcul que pour P_1. On a encore $\phi_0 = (z = 1)$. Pour les $\texttt{pre}_{G_i'}(\phi_0)$, on trouve cette fois :

$$\begin{aligned}\texttt{pre}_{G_1'}(z = 1) &= (s = 0) \wedge (x \leq 5) \wedge (z = 1) \\ \texttt{pre}_{G_2'}(z = 1) &= (s = 1) \wedge (x \leq 5) \wedge (z = 1) \\ \texttt{pre}_{G_3'}(z = 1) &= (s = 0) \wedge (1 \leq x \leq 4) \wedge (z = 1) \\ \texttt{pre}_t(z = 1) &= (z = 1) \wedge \phi_2^{\Box} \;\vee\; \texttt{pre}_{G_1'}(z = 1) \;\vee\; \texttt{pre}_{G_2'}(z = 1) \;\vee\; \texttt{pre}_{G_3'}(z = 1) \\ &= (z = 1) \wedge \phi_2^{\Box} \\ \texttt{true} \propto \texttt{pre}_t(z = 1) &= (s = 0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \;\vee \\ &\quad (s = 1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \;\vee \\ &\quad (s = 2) \wedge (z \leq 1)\end{aligned}$$

On obtient donc pour la première itération :

$$\begin{aligned}\phi_1 = (z = 1) \;\vee\; &(s = 0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \;\vee \\ &(s = 1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \;\vee \\ &(s = 2) \wedge (z \leq 1)\end{aligned}$$

On passe ensuite à la seconde itération :

$$\begin{aligned}\texttt{pre}_{G_1'}(\phi_1) &= (s = 0) \wedge (x \leq 5) \wedge (z = 1) \;\vee \\ &\quad (s = 0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \\ &= (s = 0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \\ \texttt{pre}_{G_2'}(\phi_1) &= (s = 1) \wedge (x \leq 5) \wedge (z = 1) \;\vee \\ &\quad (s = 1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \\ &= (s = 1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z + 4) \\ \texttt{pre}_{G_3'}(\phi_1) &= (s = 0) \wedge (1 \leq x \leq 4) \wedge (z = 1) \;\vee \\ &\quad (s = 0) \wedge (1 \leq x \leq 4) \wedge (z \leq 1) \\ &= (s = 0) \wedge (1 \leq x \leq 4) \wedge (z \leq 1)\end{aligned}$$

$$\begin{aligned}
\mathtt{pre}_t(\phi_1) &= \phi_1 \wedge \phi_2^{\Box} \ \vee \ \mathtt{pre}_{G'_1}(\phi_1) \ \vee \ \mathtt{pre}_{G'_2}(\phi_1) \ \vee \ \mathtt{pre}_{G'_3}(\phi_1) \\
&= (s=0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=2) \wedge (z \leq 1) \\
\mathtt{true} \propto \mathtt{pre}_t(\phi_1) &= (s=0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=2) \wedge (z \leq 1)
\end{aligned}$$

On obtient donc

$$\phi_1 \ = \ (z=1) \ \vee \ \mathtt{true} \propto \mathtt{pre}_t(\phi_1) \ = \ \phi_1$$

L'itération est donc stabilisée, et on trouve

$$\begin{aligned}
|\mu X.\varphi(X)| &= \phi_0 \ \vee \ \phi_1 \\
&= (z=1) \ \vee \\
&\quad (s=0) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=1) \wedge (x \leq 5) \wedge (z \leq 1) \wedge (x \leq z+4) \ \vee \\
&\quad (s=2) \wedge (z \leq 1)
\end{aligned}$$

On a donc

$$\begin{aligned}
\phi_{bt} &= |\mu X.\varphi(X)|[z:=0] \\
&= ((s=0) \wedge (x \leq 4)) \ \vee \ ((s=1) \wedge (x \leq 4)) \ \vee \ (s=2)
\end{aligned}$$

Rappelons que

$$\phi_2^{\Box} = ((s=0) \wedge (x \leq 5)) \ \vee \ ((s=1) \wedge (x \leq 5)) \ \vee \ (s=2)$$

Le prédicat $\phi_2^{\Box} \Rightarrow \phi_{bt}$ n'est donc pas valide, car il n'est pas satisfait par les états définis par

$$((s=0) \ \vee \ (s=1)) \wedge (4 < x \leq 5)$$

Ces états sont ceux pour lesquels il n'est plus jamais possible de passer dans le nœud 2.

Le programme P_2 n'est donc pas bien temporisé.

6.4.5 Exemple de vérification : le contrôleur de passage à niveau

Nous reprenons ici l'exemple présenté aux paragraphes 3.3.4 et 5.1.3.3, en le simplifiant légèrement. Nous intégrons le contrôleur et la barrière en une seule entité, et nous utilisons une seule action o pour indiquer la sortie de la zone du passage à niveau (au lieu de o pour la sortie effective et s pour son signalement). Nous modifions également certaines valeurs des délais. Les graphes temporisés du train et du contrôleur sont présentés sur la figure 6.4. Les communications sont définies par $a_t|a_c = a$ et $o_t|o_c = o$. On force celles-ci à avoir lieu en encapsulant le système par l'ensemble $\{a_t, a_c, o_t, o_c\}$.

Nous considérons que l'unité de temps est la minute, et nous nous situons dans le cas $D = \mathbb{Q}_+$.

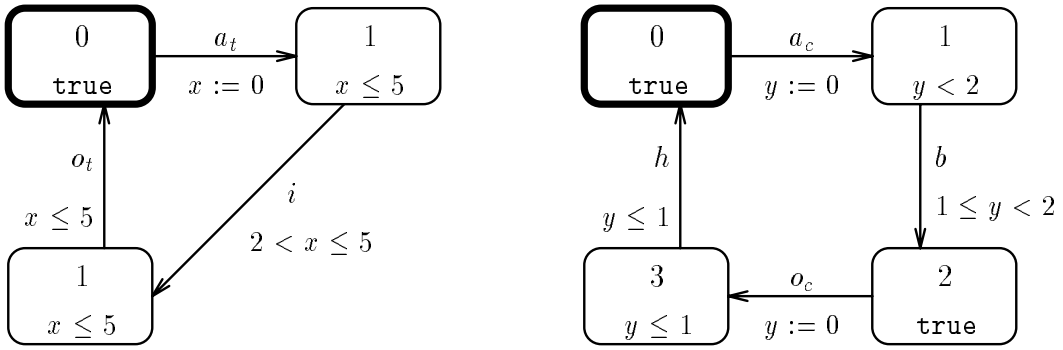


Figure 6.4 : graphes temporisés du train et du contrôleur

Lorsque le train approche, il envoie le signal a au moins 2 minutes avant d'entrer dans le passage (événement i). L'événement o signale la sortie du passage ; on impose qu'il ait lieu au plus 5 minutes après le signal a.

Lorsqu'il reçoit le signal d'approche, le contrôleur ferme la barrière. Cette action prend au moins une minute, mais moins de 2 minutes. L'événement b a lieu lorsque la barrière arrive en bas. Le contrôleur attend ensuite le signal o. Il ouvre alors la barrière en moins d'une minute (l'événement h signale la fin de la remontée).

Nous utilisons une variable T dont les valeurs sont 0, 1 ou 2 pour modéliser les nœuds du train, et une variable C (à valeurs dans $\{0,1,2,3\}$) pour ceux du contrôleur.

Le système complet est décrit par le PCG $S = (\{x,y\}, \phi^0, \phi^\Box, \mathcal{G})$ qui représente la composition parallèle encapsulée des deux graphes temporisés :

$$\begin{aligned}
\phi^0 &= (T=0) \wedge (x=0) \wedge (C=0) \wedge (y=0) \\
\phi^\Box &= (T=0) \ \vee \ ((T=1) \vee (T=2)) \wedge (x \leq 5) \ \vee \\
&\quad (C=0) \ \vee \ (C=1) \wedge (y<2) \ \vee \ (C=2) \ \vee \ (C=3) \wedge (y \leq 1) \\
\mathcal{G} &= \{ \ (T=0) \wedge (C=0) \xrightarrow{a} \{T:=1, C:=1, x:=0, y:=0\} \\
&\quad (T=1) \wedge (2<x\leq 5) \xrightarrow{i} \{T:=2\} \\
&\quad (C=1) \wedge (1 \leq y < 2) \xrightarrow{b} \{C:=2\} \\
&\quad (T=2) \wedge (C=2) \wedge (x \leq 5) \xrightarrow{o} \{T:=0, C:=3, y:=0\} \\
&\quad (C=3) \wedge (y \leq 1) \xrightarrow{h} \{C:=0\} \ \}
\end{aligned}$$

Il faut vérifier que le programme est bien temporisé, ce que nous ne faisons pas ici.

Nous souhaitons prouver qu'à partir d'un état initial, la barrière n'est jamais fermée pendant plus de 5 minutes. Cette propriété est exprimée par la formule de TCTL

$$\varphi \stackrel{\text{def}}{=} \phi^0 \Rightarrow \forall\Box((C=2) \Rightarrow \forall\Diamond_{\leq 5}(C=0))$$

Le système étant bien temporisé, cette formule est équivalente à la formule de Tμ

$$\phi^0 \Rightarrow \neg\mu Y.((C=2) \wedge z{\downarrow}\, \mu X.(z>5) \ \vee \ ((C \neq 0) \propto X)) \ \vee \ (\texttt{true} \propto Y)$$

Nous pouvons restreindre tous les calculs par $\phi^\Box$, car $\phi^0 \Rightarrow \phi^\Box$ est valide.

Le prédicat caractéristique du point fixe interne est calculé itérativement sous la forme $\phi = \bigvee_i \phi_i$, avec

$$\begin{aligned} \phi_0 &= \phi^{\Box} \wedge (z > 5) \\ \phi_{i+1} &= \phi_i \vee ((C \neq 0) \propto \phi_i) \end{aligned}$$

On a alors :

$$\begin{aligned}
\phi_1 &= \phi_0 \vee ((C \neq 0) \propto \phi_0) \\
&= \phi^{\Box} \wedge (z > 5) \vee \\
&\quad ((T = 0) \vee ((T = 1) \vee (T = 2)) \wedge (x \leq 5) \wedge z > x) \\
&\quad \wedge ((C = 1) \wedge (y < 2) \wedge (z > y + 3) \vee) \\
&\qquad (C = 2) \vee \\
&\qquad (C = 3) \wedge (y \leq 1) \wedge (z > y + 4) \\
\phi_2 &= \phi_1 \vee ((C \neq 0) \propto \phi_1) \\
&= \phi_1 \vee \\
&\quad (T = 0) \wedge (C = 1) \wedge (y < 2) \vee \\
&\quad ((T = 1) \vee (T = 2)) \wedge (C = 1) \wedge (x \leq 5) \wedge (y < 2) \wedge (z > x) \wedge (y \geq x - 4) \vee \\
&\quad (T = 2) \wedge (C = 2) \wedge (x \leq 5) \wedge (z > x - 1) \\
\phi_3 &= \phi_2 \vee ((C \neq 0) \propto \phi_2) \\
&= \phi_2 \vee \\
&\quad (T = 1) \wedge (C = 2) \wedge (x \leq 5) \wedge (z > x - 1) \vee \\
&\quad (T = 2) \wedge (C = 1) \wedge (x \leq 5) \wedge (y < 2) \wedge (y \geq x - 4) \wedge (z > x - 1) \\
\phi_4 &= \phi_3 \vee ((C \neq 0) \propto \phi_3) \\
&= \phi_3 \vee (T = 1) \wedge (C = 1) \wedge (x \leq 5) \wedge (y < 2) \wedge (y \geq x - 4) \wedge (z > x - 1)
\end{aligned}$$

On trouve alors $\phi_5 = \phi_4$. Donc

$$\phi = \phi_0 \vee \phi_1 \vee \phi_2 \vee \phi_3 \vee \phi_4 = \phi_4$$

Nous calculons ensuite le prédicat caractéristique de l'opérateur d'initialisation :

$$\begin{aligned}
|z{\downarrow}\phi| &= \phi[z := 0] \\
&= ((T = 0) \vee ((T = 1) \vee (T = 2)) \wedge (x < 1)) \wedge ((C = 1) \wedge (y < 2) \vee (C = 2))
\end{aligned}$$

Nous calculons à présent le point fixe externe sous la forme $\psi = \bigvee_i \psi_i$, avec

$$\begin{aligned} \psi_0 &= (C = 2) \wedge |z{\downarrow}\phi| \\ \psi_{i+1} &= \psi_i \vee (\texttt{true} \propto \psi_i) \end{aligned}$$

Le calcul s'arrête à la seconde itération ($\psi_2 = \psi_1$), et on trouve

$$\psi = \psi_0 \vee ((T = 0) \vee ((T = 1) \vee (T = 2)) \wedge (x < 1) \wedge (x < y)) \wedge (C = 1) \wedge (y < 2)$$

On vérifie alors facilement que le prédicat $\phi^0 \Rightarrow \neg\psi$ est valide. Par conséquent $|\varphi| = \texttt{true}$, qui est le résultat voulu.

Conclusion

Bilan

L'objectif de ce travail était, d'une part, l'étude de la spécification des systèmes temps réel par le biais des algèbres de processus et, d'autre part, la recherche de techniques de vérification prenant en compte les particularités de ces systèmes, c'est-à-dire la présence de délais dans les descriptions.

Spécification

Nous avons en premier lieu défini l'algèbre ATP et ses extensions pour des domaines temporels quelconques ATP_g et ATP_D.

ATP est basée sur l'hypothèse de synchronisme, ce qui permet de raisonner à partir d'une notion de temps *abstrait*. Ce principe est maintenant adopté dans toutes les autres algèbres de processus temporisés, Timed CSP exceptée. Les actions s'exécutant en temps nul, la distinction entre leurs occurrences et le passage du temps est clairement définie ; elle conduit à un formalisme simple d'expression de la sémantique opérationnelle et à une structure de modèles facilement manipulable, en particulier dans le cas discret.

Le choix des opérateurs et de leur sémantique a été guidé par un souci de *minimalité* du nombre de constructions, de *simplicité* des règles de sémantique et d'*expressivité* de l'algèbre. Nous avons montré comment étendre une algèbre non temporisée AUP au moyen des deux opérateurs de préfixage urgent et de délai unitaire, de manière à préserver toutes les propriétés des termes d'AUP.

À partir de l'algèbre de processus temporisés ainsi obtenue, nous avons défini un certain nombre d'autre opérateurs permettant de spécifier des contraintes temporelles. En particulier, l'opérateur de *chien de garde* est très général ; il permet de dériver diverses constructions comme les retards, les délais de début ou d'exécution, ... L'opérateur de *restriction temporelle* présente un grand intérêt dans le cas de systèmes sous-spécifiés, dont il restreint l'ensemble des comportements.

La classe des systèmes de transitions obtenus est caractérisée par le *déterminisme temporel* et la possibilité de modéliser l'*urgence*, et de plus par l'*additivité temporelle* et la *persistance partielle* lorsque le domaine temporel est dense. À cet égard, le pouvoir d'expression d'ATP_D est parmi les plus importants parmi les algèbres existantes. Seule ACP_ρ de Baeten et Bergstra présente des constructions qu'on ne peut définir dans ATP_D sans modifier la classe de modèles. La différence entre les deux algèbres est due à la possibilité dans ACP_ρ d'absence de persistance partielle. Ce critère de comparaison n'est pas réellement significatif, puisqu'il ne concerne que les domaines

temporels denses. L'intérêt de ceux-ci ne se situe pas au niveau du comportement des systèmes, pour l'implantation desquels il faut de toutes façons se ramener à un domaine discret. Il réside bien plutôt dans la *préservation des propriétés de sûreté* lorsqu'on discrétise le domaine. Cette préservation est prouvée dans le cas d'ATP_D ; il n'est pas sûr qu'elle reste vraie pour des classes de modèles plus générales.

Vérification

Le problème essentiel à résoudre pour la vérification est celui de l'explosion de la taille des modèles. Nous nous sommes intéressés à la cause d'explosion spécifique aux systèmes temps réel, qui est la variation de cette taille en fonction des valeurs des délais.

Pour résoudre ce problème, nous avons étudié les *graphes temporisés*. On peut les considérer comme des automates étendus au moyen de compteurs de temps (horloges). Ils sont donc candidats au rôle de *modèles intermédiaires* des systèmes temporisés : l'expansion du parallélisme et des autres opérateurs de structure est effectuée, mais les contraintes temporelles sont encore exprimées de façon symbolique par des conditions sur les valeurs des horloges. La taille d'un graphe temporisé ne dépend donc pas des valeurs des délais, mais uniquement de la structure du système.

Nous avons présenté une *méthode de traduction* d'un processus d'ATP_D en un graphe temporisé, indépendante du domaine temporel considéré, et qui garantit la conservation de la sémantique du système.

L'utilité de tels modèles intermédiaires est mise en évidence par la conception d'un algorithme de *model checking symbolique* qui s'applique aux graphes temporisés quel que soit le domaine temporel. La définition d'un *μ-calcul temporel* $\text{T}\mu$ et de la sémantique de ses formules sur des modèles denses était un préalable à l'élaboration de l'algorithme. Il était en particulier crucial d'identifier une relation "état suivant" dont l'itération garantit que tous les états d'une exécution sont visités. Elle doit donc permettre d'avancer au-delà de tout instant ; dans le cas dense, elle ne doit cependant pas imposer d'avancer de plus d'une fraction infinitésimale de temps. Nous avons montré que ces deux exigences a priori contradictoires peuvent cependant être satisfaites.

L'algorithme lui-même repose sur la notion de *prédicat d'état*, dont la validité est décidable. L'ensemble caractéristique d'une formule est défini comme un point fixe de fonctionnelle sur les prédicats d'état. La calculabilité de ce point fixe est obtenue grâce aux *graphes de régions* définis par Alur, Courcoubetis et Dill [ACD90], qui permettent de raisonner sur un quotient fini du modèle éventuellement infini non dénombrable. L'algorithme pour $\text{T}\mu$ permet d'en déduire un pour la logique TCTL. En effet, si ces deux logiques sont incomparables dans le cas général, nous avons montré que $\text{T}\mu$ est strictement plus expressive que TCTL si on se restreint aux modèles des graphes temporisés *bien temporisés*. Il faut donc savoir décider si un système est bien temporisé, ce qui est possible en évaluant une formule de $\text{T}\mu$.

L'intérêt de la méthode symbolique par rapport à l'algorithme énumératif présenté dans [ACD90] réside dans le fait qu'on évite à double titre l'explosion du nombre des états : d'une part, on effectue les calculs sur des ensembles de régions au lieu de considérer chaque région individuellement et on ne scinde un ensemble que lorsque cela est nécessaire ; d'autre part, on ne construit pas réellement de modèle, mais on manipule symboliquement la relation de transition comme un transformateur de prédicats.

Perspectives

La validation des principes théoriques exposés dans cette thèse requiert l'élaboration de logiciels qui les mettent en pratique. S. Yovine a développé un compilateur de processus ATP en graphes temporisés appliquant la méthode effective présentée dans [NSY92]. L'algorithme de model checking symbolique est en cours d'implantation par S. Yovine et A. Olivero, ce dernier s'attachant particulièrement à mettre au point des méthodes de décision efficaces dans la théorie des prédicats d'état. Nous souhaitons ainsi obtenir une chaîne de compilation-vérification de systèmes décrits en ATP.

Les techniques spécifiques aux problèmes temporels mises en œuvre dans ce prototype devraient pouvoir servir de référence pour la conception d'outils permettant de travailler en grandeur réelle à partir de langages de programmation plus sophistiqués. En particulier, la phase de vérification est relativement indépendante du langage initial, pour peu que les programmes puissent être compilés en graphes temporisés ou dans un formalisme équivalent.

L'algorithme de model checking présenté est sans doute trop général pour être réellement efficace dans tous les cas. Si on veut obtenir des résultats de vérification en temps raisonnable, il est souhaitable d'identifier certaines classes de propriétés temps réel pour lesquelles on peut obtenir des algorithmes plus efficaces.

L'algèbre n'est évidemment pas destinée à devenir un langage de programmation. Il paraît plus réaliste d'appliquer les résultats obtenus à des langages existants. Dans cette optique, deux directions de travail sont à envisager :

- La première concerne l'introduction de constructions temporelles dans des langages n'en possédant pas. La méthode utilisée pour l'algèbre AUP semble pour cela facilement généralisable. Le candidat idéal à une telle intégration est certainement le langage LOTOS. G. Leduc, de l'Université de Liège, en a déjà proposé une extension au moyen de certains opérateurs d'ATP (délai de début et délai d'exécution) [Led91]. L'addition d'autres opérateurs tels que le chien de garde généralisé ou la restriction temporelle ne devrait pas poser de problème particulier.

  Le compilateur LOTOS développé par H. Garavel [Gar89] utilise comme modèles intermédiaires les réseaux de Petri. On peut envisager de les étendre avec des horloges, afin d'obtenir un formalisme comparable (en ce qui concerne l'expression des contraintes temporelles) à celui des programmes temporisés à commandes gardées. Il sera ensuite nécessaire d'adapter les algorithmes de model checking pour les appliquer à de tels réseaux de Petri "temporisés".

- La seconde direction consiste à analyser les rapports entre les graphes temporisés et le code OC [CPPS90] produit par les compilateurs des langages synchrones ESTEREL, LUSTRE et ARGOS. L'objectif est d'étudier comment l'approche symbolique présentée dans cette thèse peut être adaptée pour vérifier des programmes synchrones.

  Il semble facile d'identifier dans les automates OC les compteurs qui correspondent à des horloges dans les graphes temporisés. Nous nous heurtons cependant au problème du temps multiforme, qui n'a pas du tout été examiné dans ce document. Le temps multiforme permet de définir des constructions temporelles dont les délais sont relatifs à des échelles de temps indépendantes.

  La vérification de propriétés dans un tel cadre passe en premier lieu par la définition de logiques temporelles temps réel à temps multiforme. Les formules doivent comporter pour chaque horloge l'indication de l'échelle de temps sur laquelle elle fonctionne. Il faut ensuite

pouvoir définir une (ou des) relations "état suivant" sur les modèles et un (ou des) transformateurs de prédicats $\propto$ qui prennent en compte le fait que les vitesses des horloges ne sont pas toutes corrélées. Une première approche simplifiée consiste peut-être à ne vérifier que des propriétés relatives à une seule échelle de temps. On peut dans ce cas envisager de faire abstraction (dans une mesure qui reste à définir) des valeurs des horloges relatives aux autres échelles de temps.

L'intégration du temps monoforme dans un langage tel que LOTOS ne soulève donc pas a priori de problème particulier, et devrait être réalisée d'ici peu. Il n'en va pas de même pour la mise au point de méthodes de vérification de propriétés temps réel dans le cadre multiforme des langages synchrones. Les idées émises ci-dessus relèvent pour l'instant du domaine de la conjecture. Une étude approfondie des modèles dans le cas du temps multiforme, telle que celle que nous avons menée dans le cas monoforme, est sans doute nécessaire pour mettre au point des techniques de vérification adaptées.

Bibliographie

[AB84] D. Austry et G. Boudol. Algèbre de processus et synchronisation. *Theoretical Computer Science*, 30, 1984.

[ACD90] R. Alur, C. Courcoubetis et D. Dill. Model-checking for real-time systems. Dans *Proceedings of the fifth annual IEEE symposium on Logics In Computer Science*, pages 414–425, Philadelphie, PA, États-Unis, juin 1990.

[AD90] R. Alur et D. Dill. Automata for modeling real-time systems. Dans M. S. Paterson, éditeur, *Proceedings of ICALP '90*, pages 322–335. Springer Verlag, 1990.

[Ada] The programming language ADA reference manual. LNCS 155, Springer Verlag, 1983.

[AFH91] R. Alur, T. Feder et T. Henzinger. The benefits of relaxing punctuality. Dans *Proceedings of the tenth annual symposium on Principles Of Distributed Computing*, pages 139–152. ACM Press, 1991.

[AH89] R. Alur et T. Henzinger. A really temporal logic. Dans *Proceedings of the 30^{th} Symposium on Foundations of Computer Science*, pages 164–169. IEEE Computer Society Press, 1989.

[AH91] R. Alur et T. Henzinger. Logics and models of real time: a survey. Dans J. W. de Bakker, C. Huizing, W. P. de Roever et G. Rozenberg, éditeurs, *LNCS 600: Proceedings of REX workshop "Real-time: theory in practice"*, Mook, Pays-Bas, juin 1991. Springer Verlag.

[AL88] M. Abadi et L. Lamport. The existence of refinement mappings. SRC 29, Digital Equipement Corporation, août 1988.

[Alu91] R. Alur. *Techniques for automatic verification of real-time systems.* Thèse, Department of Computer Science, Université de Stanford, CA, États-Unis, 1991.

[BB90] J. C. M. Baeten et J. A. Bergstra. Real time process algebra. Rapport de recherche CS-R9053, Centre for Mathematics and Computer Science, Amsterdam, Pays-Bas, 1990.

[BC84] G. Berry et L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. Dans *LNCS 197: Proceedings CMU Seminar on Concurrency*, pages 389–448. Springer Verlag, 1984.

[BCD+90] J. B. Burch, E. M. Clarke, D. Dill, L. J. Hwang et K. L. McMillan. Symbolic model checking: 10^{20} states and beyond. Dans *Proceedings of LICS '90*, pages 428–439. IEEE Computer Society Press, 1990.

[BCG87] G. Berry, P. Couronné et G. Gonthier. Programmation synchrone des systèmes réactifs : le langage ESTEREL. *Technique et Science Informatiques*, 4:305–316, 1987.

[BD90] E. Badouel et P. Darondeau. A note on guarded recursion. Rapport de recherche 1249, INRIA, Rennes, France, juin 1990.

[BFG+91] A. Bouajjani, J.-C. Fernandez, S. Graf, C. Rodriguez et J. Sifakis. Safety for branching time semantics. Dans J. Leach Albert, B. Monien et M. Rodríguez Artalejo, éditeurs, *LNCS 510: Proceedings of ICALP '91*, pages 76–92, Madrid, Espagne, juillet 1991. Springer Verlag.

[BFH90] A. Bouajjani, J.-C. Fernandez et N. Halbwachs. On the verification of safety properties. Rapport technique SPECTRE L12, LGI-IMAG, Grenoble, France, mars 1990.

[BHR84] S. D. Brookes, C. A. R. Hoare et A. W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BIM88] B. Bloom, S. Istrail et A. R. Meyer. Bisimulation can't be traced: preliminary report. Dans *Conference records of the 15th ACM symposium on Principles Of Programming Languages*, pages 229–239, San Diego, Californie, États-Unis, 1988.

[BK84] J. A. Bergstra et J. W. Klop. Algebra of Communicating Processes. Rapport de recherche CS-R8420, Centre for Mathematics and Computer Science, Amsterdam, Pays-Bas, 1984.

[BK86] J. A. Bergstra et J. W. Klop. Process algebra: specification and verification in bisimulation semantics. Dans M. Hazewinkel, J. K. Lenstra et L. G. L. T. Meertens, éditeurs, *Proceedings of the CWI symposium Mathematics and Computer Science II (CWI Monograph 4)*, pages 61–94, Amsterdam, Pays-Bas, 1986. North Holland.

[BKP86] H. Barringer, R. Kuiper et A. Pnueli. A really abstract concurrent model and its temporal logic. Dans *Proceedings of the 13th ACM symposium on Principles Of Programming Languages*, pages 173–183, 1986.

[BL91] T. Bolognesi et F. Lucidi. LOTOS-like process algebra with urgent or timed interactions. Dans K. Parker et G. Rose, éditeurs, *Proceedings of the fourth international conference on Formal Description Techniques (FORTE)*. North-Holland, novembre 1991.

[Bry86] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35 (8), 1986.

[CBM89] O. Coudert, C. Berthet et J.-C. Madre. Verification of synchronous sequential machines based on symbolic execution. Dans J. Sifakis, éditeur, *LNCS 407: Proceedings of the international workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France, juin 1989. Springer Verlag.

[CE81] E. M. Clarke et E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. Dans *LNCS 131: Proceedings of workshop on Logic of Programs*. Springer Verlag, 1981.

[CES86] E. M. Clarke, E. A. Emerson et A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8, 1986.

[CHPP87] P. Caspi, N. Halbwachs, D. Pilaud et J. Plaice. LUSTRE: a declarative language for programming synchronous systems. Dans *14^{th} symposium on Principles Of Programming Languages*, Münich, Allemagne, janvier 1987.

[CLM89] E. M. Clarke, D. E. Long et K. L. McMillan. Compositional model checking. Dans *Proceedings of LICS '89*. IEEE Computer Society Press, 1989.

[Cou90] P. Couronné. *Le système* ESTEREL V2. Thèse, Université de Paris VII, 1990.

[CPPS90] P. Couronné, J.-P. Paris, J. Plaice et J.-B. Saint. The LUSTRE-ESTEREL portable format (version oc3). Rapport technique, École Nationale Supérieure de Mines de Paris, 1990.

[CPS89] R. Cleaveland, J. G. Parrow et B. Steffen. The concurrency workbench. Dans J. Sifakis, éditeur, *LNCS 407: Proceedings of the international workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France, juin 1989. Springer Verlag.

[CS91] R. Cleaveland et B. Steffen. A linear-time model checking algorithm for the alternation-free modal μ-calculus. Dans K. G. Larsen, éditeur, *Proceedings of the 3^{rd} workshop on Computer-Aided Verification*, Ålborg, Danemark, juillet 1991.

[CVWY90] C. Courcoubetis, M. Vardi, P. Wolper et M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. Dans R. P. Kurshan et E. M. Clarke, éditeurs, *Proceedings of the 2^{nd} workshop on Computer-Aided Verification*, Rutgers, États-Unis, juin 1990.

[DS89] J. Davies et S. Schneider. An introduction to Timed CSP. Rapport de recherche PRG-75, Oxford University Computing Laboratory, GB, août 1989.

[dSV89] R. de Simone et D. Vergamini. Aboard AUTO. Rapport technique 111, INRIA, 1989.

[EC80] E. A. Emerson et E. M. Clarke. Characterizing correctness properties of parallel programs as fixpoints. Dans *LNCS 85: Proceedings of ICALP '80*, pages 169–181. Springer Verlag, 1980.

[EC82] E. A. Emerson et E. M. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[EH86] E. A. Emerson et J. Halpern. "Sometimes" and "not never" revisited: on branching time versus linear time temporal logic. *Journal of the ACM*, 33 (1):151–178, 1986.

[EL86] E. A. Emerson et C. L. Lei. Efficient model checking in fragments of the propositional μ-calculus. Dans *Proceedings of LICS '86*. IEEE Computer Society Press, 1986.

[Eme91] E. A. Emerson. Real time and the μ-calculus. Dans J. W. de Bakker, C. Huizing, W. P. de Roever et G. Rozenberg, éditeurs, *LNCS 600: Proceedings of REX workshop "Real-time: theory in practice"*, Mook, Pays-Bas, juin 1991. Springer Verlag.

[EMSS89] E. A. Emerson, A. Mok, A. P. Sistla et J. Srinivasan. Quantitative temporal reasoning. Dans J. Sifakis, éditeur, *LNCS 407: Proceedings of the international workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France, juin 1989. Springer Verlag.

[Fer88] J.-C. Fernandez. ALDÉBARAN, *vérification de processus communicants*. Thèse, Université Joseph Fourier, Grenoble, France, 1988.

[Gar89] H. Garavel. *Compilation et vérification de programmes LOTOS.* Thèse, Université Joseph Fourier, Grenoble, France, 1989.

[Gro89] J.-F. Groote. Transition system specification with negative premisses. Rapport de recherche CS-R8950, Centre for Mathematics and Computer Science, Amsterdam, Pays-Bas, décembre 1989.

[GS90] S. Graf et B. Steffen. Compositional minimization of finite state processes. Dans R. P. Kurshan et E. M. Clarke, éditeurs, *Proceedings of the 2^{nd} workshop on Computer-Aided Verification*, Rutgers, États-Unis, juin 1990.

[Han91] H. Hansson. *Time and probability in formal design of distributed systems.* Thèse, Université d'Uppsala, Suède, 1991.

[Har87] D. Harel. Statecharts : a visual approach to complex systems. *Science of Computer Programming*, 8–3:231–275, 1987.

[HJ90] H. Hansson et B. Jonsson. A Calculus for Communicating Systems with time and probabilitiesDans *Proceedings of IEEE RTSS*, 1990.

[HLW91] U. Holmer, K. Larsen et Wang Yi. Deciding properties of regular real timed processes. Dans *Proceedings of the 3^{rd} workshop on Computer-Aided Verification.* Université d'Ålborg, Danemark, 1991.

[HMP91] T. Henzinger, Z. Manna et A. Pnueli. Temporal proof methodologies for real-time systems. Dans *Proceedings of the 18^{th} symposium on Principles Of Programming Languages*, pages 353–366. ACM Press, 1991.

[HNSY92] T. Henzinger, X. Nicollin, J. Sifakis et S. Yovine. Symbolic model-checking for real-time systems. Dans *Proceedings of LICS '92*, juin 1992.

[Hoa78] C. A. R. Hoare. Communicating Sequential Processes. *CACM*, 21(8), 1978.

[HR90] M. Hennessy et T. Regan. A Temporal Process Algebra. Rapport de recherche 2/90, Université de Sussex, GB, avril 1990.

[HR91] M. Hennessy et T. Regan. A process algebra for timed systems. Rapport de recherche 5/91, Université de Sussex, GB, avril 1991.

[ISO88a] ISO. ESTELLE — *A Formal Description Technique based on an extended state transition model.* International Standard 9074, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, Suisse, septembre 1988.

[ISO88b] ISO. LOTOS — *A Formal Description Technique based on the temporal ordering of observational behaviour.* International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, Suisse, septembre 1988.

[JJ89] C. Jard et T. Jéron. On-line model-checking for finite linear temporal logic specification. Dans J. Sifakis, éditeur, *LNCS 407: Proceedings of the international workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France, juin 1989. Springer Verlag.

[JJ91] C. Jard et T. Jéron. Bounded-memory algorithms for verification "on-the-fly". Dans K. G. Larsen, éditeur, *Proceedings of the 3^{rd} workshop on Computer-Aided Verification*, Ålborg, Danemark, juillet 1991.

[Klu91] A. S. Klusener. Completeness in real time process algebra. Rapport de recherche CS-R9106, Centre for Mathematics and Computer Science, Amsterdam, Pays-Bas, janvier 1991.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27, 1983.

[Lam83] L. Lamport. What good is temporal logic? Dans R. E. A. Mason, éditeur, *Information Processing 83: proceedings of the 9th IFIP World Computer Congress*, pages 657–668. Elsevier Science Publishers, 1983.

[LBBG85] P. Le Guernic, A. Benveniste, P. Bournal et T. Gauthier. Signal: a data flow oriented language for signal processing. Rapport de recherche 246, IRISA, Rennes, France, 1985.

[Led91] G. Leduc. On the design and properties of TLOTOS. Rapport de recherche S.A.R.T. 91/04/13, Université de Liège, Belgique, 1991.

[Mar90] F. Maraninchi. ARGOS *: un langage graphique pour la conception, la description et la validation des systèmes réactifs.* Thèse, Université Joseph Fourier, Grenoble, France, 1990.

[MF76] P. Merlin et D. J. Farber. Recoverability of communication protocols – implications of a theoretical study. *IEEE Transactions on Communications*, COM-24(9), septembre 1976.

[Mil80] R. Milner. A Calculus of Communicating Systems. Dans *LNCS 92*. Springer Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25, 1983.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.

[Mil89] R. Milner. *Communication and concurrency.* Prentice-Hall, 1989.

[Mou92] L. Mounier. *Méthodes de vérification de spécifications comportementales : étude et mise en œuvre.* Thèse, Université Joseph Fourier, Grenoble, France, 1992.

[MP81] Z. Manna et A. Pnueli. The temporal framework for concurrent programs. Dans R. S. Boyer et J. S. Moore, éditeurs, *The correctness problem in computer science*, pages 215–274. Academic Press, 1981.

[MP89] Z. Manna et A. Pnueli. The anchored version of the temporal framework. Dans J. W. de Bakker, W. P. de Roever et G. Rozenberg, éditeurs, *LNCS 354: Linear time, branching time, and partial order in logics and models for concurrency.* Springer Verlag, 1989.

[MT90] F. Moller et C. Tofts. A temporal calculus of communicating processes. Dans J. C. M. Baeten et J. W. Klop, éditeurs, *LNCS 458: Proceedings of CONCUR '90 (Theories of concurrency: unification and extension)*, pages 401–415, Amsterdam, Pays-Bas, août 1990. Springer Verlag.

[MW84] Z. Manna et P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6(1):68–93, 1984.

[Nic88] X. Nicollin. *Étude de la notion de temps dans les algèbres de processus communicants : application au langage* ESTELLE/R. Rapport de DEA, Institut National Polytechnique de Grenoble, France, 1988.

[NRSV90] X. Nicollin, J.-L. Richier, J. Sifakis et J. Voiron. ATP: an algebra for timed processes. Dans *Proceedings of the IFIP TC 2 working conference on Programming Concepts and Methods*, Mer de Galilée, Israël, avril 1990.

[NS90] X. Nicollin et J. Sifakis. The algebra of timed processes ATP: theory and application. *Information and Computation*, (à paraître), décembre 1990.

[NS91] X. Nicollin et J. Sifakis. An overview and synthesis on timed process algebras. Dans K. G. Larsen, éditeur, *Proceedings of the 3rd workshop on Computer-Aided Verification*, Ålborg, Danemark, juillet 1991.

[NSY91] X. Nicollin, J. Sifakis et S. Yovine. From ATP to timed gaphs and hybrid systems. Dans J. W. de Bakker, C. Huizing, W. P. de Roever et G. Rozenberg, éditeurs, *LNCS 600: Proceedings of REX workshop "Real-time: theory in practice"*, Mook, Pays-Bas, juin 1991. Springer Verlag.

[NSY92] X. Nicollin, J. Sifakis et S. Yovine. Compiling real-time specifications into timed automata. *IEEE Transactions on Software Engineering, special issue on specification and analysis of real-time systems*, 18(9):794–804, septembre 1992.

[OL82] S. Owicki et L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455–595, 1982.

[Par80] D. M. R. Park. Concurrency and automata on infinite sequences. Dans *LNCS 104*. Springer Verlag, 1980.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Rapport de recherche DAIMI FN-19, Université d'Århus, Computer Science department, Århus, Danemark, 1981.

[Pnu77] A. Pnueli. The temporal logic of concurrent programs. Dans *LNCS 194: Proceedings of 12th ICALP*. Springer Verlag, 1977.

[Pnu86] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. Dans *LNCS 224: Current trends in concurrency*, pages 510–584. Springer Verlag, 1986.

[QS81] J. Queille et J. Sifakis. Specification and verification of concurrent systems in CESAR. Dans *LNCS 137: Proceedings of 5th International Symposium in Programming*, pages 337–351. Springer Verlag, 1981.

[Rat92] C. Ratel. LESAR *: un outil pour la vérification d'invariants de programmes* LUSTRE. Thèse, Université Joseph Fourier, Grenoble, France, 1992.

[Ray91] P. Raymond. *Compilation efficace d'un langage déclaratif synchrone : le générateur de code* LUSTRE-*V3*. Thèse, Institut National Polytechnique de Grenoble, France, 1991.

[Rod88] C. Rodriguez. *Spécification et validation de systèmes en* XESAR. Thèse, Institut National Polytechnique de Grenoble, France, 1988.

[RR88] G. M. Reed et A. W. Roscoe. A timed model for Communicating Sequential Processes. *Theoretical Computer Science*, 58 (pp 249–261), 1988.

[RRSV87] J.-L. Richier, C. Rodriguez, J. Sifakis et J. Voiron. XESAR*: A Tool for Protocol Validation. User's Guide*. LGI-IMAG, Grenoble, France, 1987.

[RSV87] J.-L. Richier, J. Sifakis et J. Voiron. Une algèbre des processus temporisés. Dans A. Arnold, éditeur, *Actes du deuxième colloque* C^3, Angoulême, France, mai 1987.

[Sch91] S. Schneider. *An operational semantics for Timed CSP*. Programming Research Group, Université d'Oxford, GB, février 1991.

[Sif82] J. Sifakis. A unified approach for studying the properties of transition systems. *Theoretical Computer Science*, 18, 1982.

[Ver87] D. Vergamini. *Vérification de réseaux d'automates finis par équivalences observationnelles : le système AUTO*. Thèse, Université de Nice, France, 1987.

[VT91] A. Valmari et M. Tienari. An improved failure equivalence for finite-state systems with a reduction algorithm. Dans B. Jonsson, J. Parrow et B. Pehrson, éditeurs, *Proceedings of the 11^{th} IFIP international workshop on Protocol Specification, Testing and Verification*. North-Holland, juin 1991.

[Wan90] Wang Yi. Real-time behaviour of asynchronous agents. Dans J. C. M. Baeten et J. W. Klop, éditeurs, *LNCS 458: Proceedings of CONCUR '90 (Theories of concurrency: unification and extension)*, pages 502–520, Amsterdam, Pays-Bas, août 1990. Springer Verlag.

[Wan91] Wang Yi. CCS + time = an interleaving model for real time systems. Dans J. Leach Albert, B. Monien et M. Rodríguez Artalejo, éditeurs, *LNCS 510: Proceedings of ICALP '91*, Madrid, Espagne, juillet 1991. Springer Verlag.

Annexes

Annexe A

Consistance de l'axiomatisation de $\mathcal{P}_t^r$

L'objet de cette annexe est la preuve du théorème 2.3 :

$$\forall P, Q \in \mathcal{P}_t^r,\ P \equiv Q \ \Rightarrow P \sim Q$$

Il faut donc considérer chacun des axiomes, et montrer que son correspondant sémantique est vrai.

Pour tous les axiomes sauf ceux de la récursion, on procède en montrant qu'une certaine relation, contenant les couples constitués des deux membres de l'axiome, est une bisimulation forte.

À titre d'exemple, nous montrons l'axiome [+6]. Il s'agit donc de prouver que pour tous P_1, P_2, Q_1 et Q_2,

$$\lfloor P_1 \rfloor Q_1 \ + \ \lfloor P_2 \rfloor Q_2 \ \equiv \ \lfloor P_1 + P_2 \rfloor (Q_1 + Q_2)$$

Pour cela, nous définissons la relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(\lfloor P_1 \rfloor Q_1 \ + \ \lfloor P_2 \rfloor Q_2, \lfloor P_1 + P_2 \rfloor (Q_1 + Q_2)) \mid P_1, P_2, Q_1, Q_2 \in \mathcal{P}_t^r\} \cup \ \mathtt{Id}$$

et nous prouvons que c'est une bisimulation forte. `Id` est la relation d'égalité sur $\mathcal{P}_t^r$, pour laquelle aucun travail n'est à faire, puisqu'elle est trivialement une bisimulation forte.

Soit $R \stackrel{\text{def}}{=} \lfloor P_1 \rfloor Q_1 + \lfloor P_2 \rfloor Q_2$ et $S \stackrel{\text{def}}{=} \lfloor P_1 + P_2 \rfloor (Q_1 + Q_2)$.

- Si $R \xrightarrow{a} R'$, alors la règle $[+_l^a]$ ou $[+_r^a]$ imposent que $\lfloor P_1 \rfloor Q_1 \xrightarrow{a} R'$ ou $\lfloor P_2 \rfloor Q_2 \xrightarrow{a} R'$. La règle $[\lfloor\ \rfloor^a]$ exige alors $P_1 \xrightarrow{a} R'$ ou $P_2 \xrightarrow{a} R'$. On en déduit $P_1 + P_2 \xrightarrow{a} R'$ (règle $[+_l^a]$ ou $[+_r^a]$), et en appliquant à nouveau $[\lfloor\ \rfloor^a]$, on obtient $S \xrightarrow{a} R'$.
- Si $S \xrightarrow{a} S'$, le raisonnement inverse donne $R \xrightarrow{a} S'$.
- Si $R \xrightarrow{\chi} R'$, la règle $[+^\chi]$ impose $\lfloor P_1 \rfloor Q_1 \xrightarrow{\chi} R_1$, $\lfloor P_2 \rfloor Q_2 \xrightarrow{\chi} R_2$, et $R' = R_1 + R_2$. Par application de $[\lfloor\ \rfloor^\chi]$, on obtient $R_1 = Q_1$ et $R_2 = Q_2$. D'autre part, cette même règle dit que $S \xrightarrow{\chi} Q_1 + Q_2$, donc $S \xrightarrow{\chi} R'$.
- Si $S \xrightarrow{\chi} S'$, le raisonnement inverse conduit à $R \xrightarrow{\chi} S'$.
- Supposons $R \xrightarrow{X^+} \delta$. L'une des règles $[+_l^{X^+}]$ ou $[+_r^{X^+}]$ doit alors s'appliquer, et donc $\lfloor P_1 \rfloor Q_1 \xrightarrow{X^+} \delta$ ou $\lfloor P_2 \rfloor Q_2 \xrightarrow{X^+} \delta$. Mais cela est impossible, car un délai unitaire n'a pas de transition par X^+. Le même argument est valable également pour S.

Si $R \xrightarrow{X^-} \delta$, alors l'une des règles $[+_l^{X^-}2]$ ou $[+_r^{X^-}2]$ doit s'appliquer, et donc soit $\lfloor P_1 \rfloor Q_1$, soit $\lfloor P_2 \rfloor Q_2$ a une transition par X^-. D'après $[\lfloor\rfloor^{X^-}]$, on en déduit que P_1 ou Q_1 a une transition par $X^?$. L'une des six règles $[+^{X^?}]$ peut forcément être utilisée pour obtenir que $P_1 + P_2$ a une transition par la variable X. En utilisant une fois encore la règle $[\lfloor\rfloor^{X^-}]$, on conclut que $S \xrightarrow{X^-} \delta$.

Si $R \xrightarrow{X^-} \delta$, un argument similaire permet de conclure que $R \xrightarrow{X^-} \delta$.

Finalement, nous avons montré que

$$\forall \ell \in \mathcal{L}_t, \forall R',\ R \xrightarrow{\ell} R' \ \Rightarrow\ \exists S'\ S \xrightarrow{\ell} S' \text{ et } R' \mathcal{R} S'$$

et

$$\forall \ell \in \mathcal{L}_t, \forall S',\ S \xrightarrow{\ell} S' \ \Rightarrow\ \exists R'\ R \xrightarrow{\ell} R' \text{ et } R' \mathcal{R} S'$$

Nous supposons donc prouvée la consistance de tous les axiomes hormis ceux de la récursion.

Afin de montrer la consistance de ces derniers, nous définissons la *substitution sémantique* $P\{Q/X\}$, et nous prouvons qu'elle coïncide avec la substitution syntaxique $P[Q/X]$.

Les règles définissant la sémantique de $P\{Q/X\}$ sont relativement nombreuses, du fait des deux types de transition par une variable :

$[\{\}_1^a] : \dfrac{P \xrightarrow{a} P'}{P\{Q/X\} \xrightarrow{a} P'\{Q/X\}}$	$[\{\}_2^a] : \dfrac{P \xrightarrow{X^?} P', Q \xrightarrow{a} Q'}{P\{Q/X\} \xrightarrow{a} Q'}$
$[\{\}_1^\chi] : \dfrac{P \xrightarrow{\chi} P', P \xrightarrow{X^+}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{\chi} P'\{Q/X\}}$	$[\{\}_2^\chi] : \dfrac{P \xrightarrow{\chi} P', P \xrightarrow{X^+} P'', Q \xrightarrow{\chi} Q'}{P\{Q/X\} \xrightarrow{\chi} P'\{Q/X\} + Q'}$
$[\{\}_1^{Y^+}] : \dfrac{P \xrightarrow{Y^+} P', P \xrightarrow{X^+}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{Y^+} P'}\ Y \neq X$ $[\{\}_3^{Y^+}] : \dfrac{P \xrightarrow{X^+} P', Q \xrightarrow{Y^+} Q'}{P\{Q/X\} \xrightarrow{Y^+} Q'}$	$[\{\}_2^{Y^+}] : \dfrac{P \xrightarrow{Y^+} P', P \xrightarrow{X^+} P'', Q \xrightarrow{\chi} Q'}{P\{Q/X\} \xrightarrow{Y^+} P'}\ Y \neq X$
$[\{\}_1^{Y^-}] : \dfrac{P \xrightarrow{Y^-} P', P \xrightarrow{X^+}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{Y^-} P'}\ Y \neq X$ $[\{\}_3^{Y^-}] : \dfrac{P \xrightarrow{Y^-} P', P \xrightarrow{X^+} P'', Q \xrightarrow{\chi}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{Y^-} P'}$ $[\{\}_5^{Y^-}] : \dfrac{P \xrightarrow{X^+} P', Q \xrightarrow{Y^-} Q'}{P\{Q/X\} \xrightarrow{Y^-} Q'}$	$[\{\}_2^{Y^-}] : \dfrac{P \xrightarrow{Y^-} P', P \xrightarrow{X^+} P'', Q \xrightarrow{Y^+}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{Y^-} P'}$ $[\{\}_4^{Y^-}] : \dfrac{P \xrightarrow{Y^+} P', P \xrightarrow{X^+} P'', Q \xrightarrow{\chi}\!\!\!\!\!\!/}{P\{Q/X\} \xrightarrow{Y^-} P'}\ Y \neq X$ $[\{\}_6^{Y^-}] : \dfrac{P \xrightarrow{X^-} P', Q \xrightarrow{Y^?} Q'}{P\{Q/X\} \xrightarrow{Y^-} Q'}$

Toutes ces règles sont dans le format GSOS, et donc l'équivalence forte est encore une congruence.

Dans la règle d'action et la règle temporelle de la récursion, nous remplaçons la substitution syntaxique par la substitution sémantique :

$$[\mathtt{rec}^a] : \frac{P \xrightarrow{a} P'}{\mathtt{rec}X \cdot P \xrightarrow{a} P'\{\mathtt{rec}X \cdot P/X\}} \qquad [\mathtt{rec}^\chi] : \frac{P \xrightarrow{\chi} P'}{\mathtt{rec}X \cdot P \xrightarrow{\chi} P'\{\mathtt{rec}X \cdot P/X\}}$$

Cette modification se justifie a posteriori une fois démontrée l'équivalence des substitutions syntaxique et sémantique.

Nous prouvons en premier lieu le lemme suivant :

Lemme A.1

$\forall P \in \mathcal{P}_t^r, \forall X,\ \mathtt{rec}X \cdot P \sim P\{\mathtt{rec}X \cdot P/X\}$

□

Preuve. Nous montrons que la relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(\mathtt{rec}X \cdot P, P\{\mathtt{rec}X \cdot P/X\}) \mid P \in \mathcal{P}_t^r\} \cup \mathtt{Id}$$

est une bisimulation forte modulo $\sim$ (cf. définition 1.13, p. 27). Aucun travail n'est à faire pour $\mathtt{Id}$.

Soit $R \stackrel{\text{def}}{=} \mathtt{rec}X \cdot P$ et $S \stackrel{\text{def}}{=} P\{\mathtt{rec}X \cdot P/X\} = P\{R/X\}$.

Nous examinons chacun des cas possibles pour les transitions de R et S.

1. $R \xrightarrow{a} R'$. D'après la règle $[\mathtt{rec}^a]$, cela n'est possible que si $P \xrightarrow{a} P'$, avec de plus $R' = P'\{\mathtt{rec}X \cdot P/X\}$. On en déduit que $S \xrightarrow{a} R'$ (règle $[\{\}_1^a]$).
2. $S \xrightarrow{a} S'$. Si cette transition est obtenue par application de la règle $[\{\}_1^a]$, alors $P \xrightarrow{a} P'$ et $S' = P'\{R/X\}$; en appliquant $[\mathtt{rec}^a]$, on obtient alors $R \xrightarrow{a} S'$. Si elle est obtenue en utilisant $[\{\}_2^a]$, alors $R \xrightarrow{a} S'$ (seconde prémisse).
3. $R \xrightarrow{X} R'$. La règle $[\mathtt{rec}^X]$ impose $P \xrightarrow{X} P'$, et $R' = P'\{\mathtt{rec}X \cdot P/X\}$. Si P n'a pas de transition par X^+, nous obtenons alors $S \xrightarrow{X} P'$ par la règle $[\{\}_1^X]$. Dans le cas contraire, l'application de la règle $[\{\}_2^X]$ donne $S \xrightarrow{X} P' + P'$. Les axiomes de $+$ étant consistants, nous avons donc $P' \sim P' + P'$.
4. $S \xrightarrow{X} S'$. Nous obtenons par le raisonnement inverse $R \xrightarrow{X} R'$, avec $R' = S'$ ou $R' \sim S' = R' + R'$.
5. $R \xrightarrow{Y^+} \delta$. En premier lieu, cela impose que R a une transition par χ (prop. 2.2 (2)). La règle $[\mathtt{rec}^X]$ est la seule applicable, et donc $P \xrightarrow{Y^+} \delta$, et $Y \neq X$. Si P n'a pas de transition par X^+, la règle $[\{\}_1^{Y^+}]$ s'applique, et donc $S \xrightarrow{Y^+} \delta$. Si P a une transition par X^+, on applique cette fois $[\{\}_2^{Y^+}]$ pour obtenir à nouveau $S \xrightarrow{Y^+} \delta$.
6. $S \xrightarrow{Y^+} \delta$. Si P n'a pas de transition par X^+, alors la transition par Y^+ ne peut provenir que de la règle $[\{\}_1^{Y^+}]$; donc Y est différent de X, et $P \xrightarrow{Y^+} \delta$; on en conclut que $R \xrightarrow{Y^+} \delta$. Si P possède une transition par X^+ et que la transition par Y^+ de S provient de la règle $[\{\}_3^{Y^+}]$, on a immédiatement $R \xrightarrow{Y^+} \delta$. Si la transition provient de la règle $[\{\}_2^{Y^+}]$, la première prémisse de celle-ci impose $P \xrightarrow{Y^+} \delta$, et de plus $Y \neq X$. On obtient donc $R \xrightarrow{Y^+} \delta$.
7. $R \xrightarrow{Y^-} \delta$. De même que précédemment, cela n'est possible que si Y est différent de X, et si $P \xrightarrow{Y^-} \delta$. La contraposée de la proposition 2.2 (1) assure que R n'a pas de transition par Y^+ ; donc l'une des règles $[\{\}_1^{Y^-}]$ et $[\{\}_2^{Y^-}]$ est applicable, et on en déduit que $S \xrightarrow{Y^-} \delta$
8. $S \xrightarrow{Y^-} \delta$. Divers cas sont possibles selon la règle appliquée pour obtenir cette transition :
   - Si elle provient d'une des règles $[\{\}_1^{Y^-}]$, $[\{\}_2^{Y^-}]$ ou $[\{\}_3^{Y^-}]$, alors $P \xrightarrow{Y^-} \delta$, et Y est différent de X (dans les deux dernières de ces trois règles, cela est garanti par la proposition 2.2 (2), puisque P a une transition par X^+) ; donc $R \xrightarrow{Y^-} \delta$
   - Elle ne peut provenir de la règle $[\{\}_4^{Y^-}]$, puisque celle-ci impose, premièrement que Y soit différent de X, ensuite que P ait une transition par Y^+, et donc R aussi, et finalement que R n'ait pas de transition par χ, ce qui contredit l'exigence précédente (prop. 2.2 (1)).
   - Si elle provient de la règle $[\{\}_5^{Y^-}]$, on a immédiatement $R \xrightarrow{Y^-} \delta$.
   - Finalement, si elle provient de la règle $[\{\}_6^{Y^-}]$, Y ne peut être égal à X, puisque R n'a pas de transition par $X^?$. Donc Y est différent de X, et R a une transition par Y^- ou Y^+. Ça ne peut pas être par Y^+, S en aurait aussi une (cf. cas 5 ci-dessus), ce qui contredit la proposition 2.2 (2). Donc $R \xrightarrow{Y^-} \delta$.

Dans chacun des huit cas, on obtient à partir de R' (resp. S') une terme S' (resp. R'), but d'une transition issue de S (resp. R) portant la même étiquette, et tel que $R' \sim\mathcal{R}\sim S'$. $\mathcal{R}$ est donc une bisimulation forte modulo $\sim$. ■

La démonstration du lemme suivant est évidente :

Lemme A.2

$$\forall P, Q \in \mathcal{P}_t^r, \forall X, P\{Q/X\} \sim P \text{ si } X \text{ n'est pas libre dans } P$$

□

Preuve. D'après la proposition 1.2 (2), si X n'est pas libre dans P, alors X n'y est pas activable. Donc le système de transitions de P ne comporte aucune transition X^+ ou X^-. En conséquence, chacune des transitions du modèle de $P\{Q/X\}$ ne peut provenir que d'une des règles $[\{\}_1^a]$, $[\{\}_1^\chi]$, $[\{\}_1^{Y^+}]$ ou $[\{\}_1^{Y^-}]$. On en déduit immédiatement que P et $P\{Q/X\}$ sont fortement équivalents. ■

Nous avons encore besoin du lemme suivant :

Lemme A.3

(1) $\forall P \in \mathcal{P}_t^r, \forall Y$, $\texttt{rec}Y \cdot P \sim \texttt{rec}Z \cdot P\{Z/Y\}$ si Z n'est pas activable dans $\texttt{rec}Y \cdot P$

(2) $\forall P, Q \in \mathcal{P}_t^r, \forall X, Y$, $(\texttt{rec}Y \cdot P)\{Q/X\} \sim \texttt{rec}Y \cdot P\{Q/X\}$
si $Y \neq X$ et Y n'est pas activable dans Q

□

Nous n'en donnons pas la preuve. Son principe est le suivant : pour (1), si on pose

$$P_1 \stackrel{\text{def}}{=} \texttt{rec}X \cdot P \quad \text{et} \quad P_2 \stackrel{\text{def}}{=} \texttt{rec}Y \cdot P\{Y/X\}$$

alors on a $P_1 \sim P\{P_1/X\}$ et $P_2 \sim P\{P_2/X\}$ (lemme A.1). Il suffit alors de montrer que la relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(Q\{P_1/X\}, Q\{Y/X\}\{P_2/Y\})\} \cup \texttt{Id}$$

est une bisimulation forte modulo $\sim$.

Pour (2), on pose

$$P_1 \stackrel{\text{def}}{=} \texttt{rec}Y \cdot P \quad \text{et} \quad P_2 \stackrel{\text{def}}{=} \texttt{rec}Y \cdot P\{Q/X\}$$

On a alors $P_1 \sim P\{P_1/Y\}$ et $P_2 \sim P\{Q/X\}\{P_2/Y\}$. Il suffit de montrer que la relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(R\{P_1/Y\}\{Q/X\}, R\{Q/X\}\{P_2/Y\})\} \cup \texttt{Id}$$

est une bisimulation forte modulo $\sim$.

Nous sommes à présent en mesure de prouver la proposition suivante :

Proposition A.4

$$\forall P, Q \in \mathcal{P}_t^r, \forall X,\ P[Q/X] \sim P\{Q/X\}$$

□

Preuve. Nous raisonnons par induction sur le nombre de symboles de P. Pour $P = P_1 \| P_2$, $P = \partial_H(P_1)$ et $P = P_1 \triangledown P_2$, il faut faire appel au fait que les termes de $\mathcal{P}_t^r$ sont régulier et au lemme A.2.

Les cas intéressants sont ceux où $P = X$ et $P = \mathtt{rec}Y \cdot R$.

Si $P = X$, alors $P[Q/X] = Q$. D'autre part, les seules transitions de P sont $P \xrightarrow{X^+} \delta$ et $P \xrightarrow{X} \delta$. Par conséquent, les transitions de $P\{Q/X\}$ ne peuvent provenir que de l'application des règles $[\{\}_2^a]$, $[\{\}_2^X]$, $[\{\}_3^{Y^+}]$ et $[\{\}_5^{Y^-}]$. $P\{Q/X\}$ a donc une transition par a, Y^+, Y^- si et seulement si Q en a aussi une ; il a une transition par χ si et seulement si son but est $\delta + Q'$, où $Q \xrightarrow{X} Q'$, et $Q' \sim \delta + Q'$ (axiomes [+1] et [+4]). On en conclut donc que $P[Q/X] \sim P\{Q/X\}$

Si $P = \mathtt{rec}Y \cdot R$ et $Y = X$, alors $P[Q/X] = P$, et $P\{Q/X\} \sim P$ d'après le lemme A.2.

Si $P = \mathtt{rec}Y \cdot R$ avec $Y \neq X$, alors $P[Q/X] = \mathtt{rec}Z \cdot R[Z/Y][Q/X]$, où Z est différent de X et n'est libre ni dans P ni dans Q. Le terme $R[Z/Y]$ a exactement le même nombre de symboles que R ; donc par hypothèse d'induction, $R[Z/Y][Q/X]$ est équivalent à $R[Z/Y]\{Q/X\}$. De plus, par hypothèse d'induction, $R[Z/Y]$ est équivalent à $R\{Z/Y\}$. L'équivalence forte étant une congruence, on en déduit que $R[Z/Y][Q/X]$ est équivalent à $R\{Z/Y\}\{Q/X\}$, et donc, en utilisant encore une fois la congruence, $P[Q/X] \sim \mathtt{rec}Z \cdot R\{Z/Y\}\{Q/X\}$.

D'autre part, Z n'étant pas activable dans P, $\mathtt{rec}Y \cdot R$ est équivalent à $\mathtt{rec}Z \cdot R\{Z/Y\}$ (lemme A.3 (1)). Finalement, comme Z est différent de X et n'est pas activable dans Q, la partie (2) du même lemme nous donne $P\{Q/X\} \sim \mathtt{rec}Z \cdot R\{Z/Y\}\{Q/X\}$, ce qui termine la preuve. ■

La consistance de l'axiome [rec1] se déduit instantanément du lemme A.1 et de la proposition A.4.

Le fait que $\sim$ soit une congruence et que les substitutions syntaxique et sémantique coïncident nous autorise à raisonner sur les classes d'équivalence de $\mathcal{P}_t^r/\sim$ au lieu de $\mathcal{P}_t^r$. La bisimulation forte sur les classes est alors l'égalité, notée $\tilde{=}$.

Pour montrer la consistance de l'axiome [rec2] (si X est gardé dans P et $P[Q/X] \equiv Q$ alors $\mathtt{rec}X \cdot P \equiv Q$), il est suffisant de prouver que si $Q_1 \sim P\{Q_1/X\}$ et $Q_2 \sim P\{Q_2/X\}$ alors $Q_1 \sim Q_2$. En effet, puisque $\mathtt{rec}X \cdot P$ est équivalent à $P\{\mathtt{rec}X \cdot P/X\}$, nous obtenons alors $Q_1 \sim Q_2 \sim \mathtt{rec}X \cdot P$.

Nous raisonnons ici sur les classes d'équivalence. Pour plus de lisibilité, la classe d'un terme P est notée P, et la relation de transition est notée $\rightarrow$.

Nous définissons la relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(R\{Q_1/X\}, R\{Q_2/X\}), R \in \mathcal{P}_t^r/\sim\}$$

et nous montrons qu'elle est une bisimulation forte. Le résultat ci-dessus est alors obtenu en choisissant $R \tilde{=} X$.

Remarquer en premier lieu que pour tout ℓ dans $\mathcal{L}_t$,

$$\forall Q_1' \in \mathcal{P}_t^r,\ \left(Q_1 \xrightarrow{\ell} Q_1' \ \Rightarrow\ \exists Q_2' \in \mathcal{P}_t^r,\ Q_2 \xrightarrow{\ell} Q_2' \text{ et } Q_1'\ \mathcal{R}\ Q_2'\right)$$
$$\forall Q_2' \in \mathcal{P}_t^r,\ \left(Q_2 \xrightarrow{\ell} Q_2' \ \Rightarrow\ \exists Q_1' \in \mathcal{P}_t^r,\ Q_1 \xrightarrow{\ell} Q_1' \text{ et } Q_1'\ \mathcal{R}\ Q_2'\right)$$

Cela est dû au fait que X est gardé dans P, et que $Q_1 \tilde{=} P\{Q_1/X\}$ et $Q_2 \tilde{=} P\{Q_2/X\}$. Une transition $Q_1 \xrightarrow{\ell} Q_1'$ provient donc d'une transition $P \xrightarrow{\ell} P'$, avec $Q_1' \tilde{=} P'\{Q_1/X\}$. Il existe alors une transition $Q_2 \xrightarrow{\ell} P'\{Q_2/X\}$.

Remarquer aussi que $\delta \; \mathcal{R} \; \delta$, puisque

$$\delta \cong \delta[Q_1/X] \cong \delta\{Q_1/X\} \; \mathcal{R} \; \delta\{Q_2/X\} \cong \delta[Q_2/X] \cong \delta$$

Nous examinons à présent les différentes transitions possibles pour $R\{Q_1/X\}$.

Supposons $R\{Q_1/X\} \xrightarrow{a} S$. Alors

- soit $R \xrightarrow{a} R'$ et $S \cong R'\{Q_1/X\}$, auquel cas nous obtenons $R\{Q_2/X\} \xrightarrow{a} R'\{Q_2/X\}$ et $(S, R'\{Q_2/X\}) \in \mathcal{R}$;
- soit $R \xrightarrow{X^+} \delta$ et $Q_1 \xrightarrow{a} S$. On en déduit $Q_2 \xrightarrow{a} S_2$, avec $S \; \mathcal{R} \; S_2$, et aussi $R\{Q_2/X\} \xrightarrow{a} S_2$.

Supposons $R\{Q_1/X\} \xrightarrow{\chi} S$. Nous devons considérer deux cas :

- $R \xrightarrow{\chi} R'$ et $R \not\xrightarrow{X^+}$, avec $S \cong R'\{Q_1/X\}$.
  Nous obtenons alors $R\{Q_2/X\} \xrightarrow{\chi} R'\{Q_2/X\}$, et $S \; \mathcal{R} \; R'\{Q_2/X\}$.
- $R \xrightarrow{\chi} R'$, $R \xrightarrow{X^+} \delta$ et $Q_1 \xrightarrow{\chi} Q_1'$, avec $S \cong R'\{Q_1/X\} + Q_1'$. Comme expliqué ci-dessus, $Q_1' \cong P'\{Q_1/X\}$ pour un certain P' ; donc $S \cong (R' + P')\{Q_1/X\}$. Mais alors, on a aussi $Q_2 \xrightarrow{\chi} P'\{Q_2/X\}$, et donc $R\{Q_2/X\}$ a une transition par χ vers $R'\{Q_2/X\} + P'\{Q_2/X\}$, qui est égal (dans $\mathcal{P}_t^r/\sim$) à $(R' + P')\{Q_2/X\}$. Finalement, $(R' + P')\{Q_1/X\} \; \mathcal{R} \; (R' + P')\{Q_2/X\}$.

Supposons $R\{Q_1/X\} \xrightarrow{Y^+} \delta$. Quelle que soit le règle de sémantique employée pour obtenir cette transition, le fait que Q_1 et Q_2 ont les mêmes étiquettes de transition nous donne immédiatement $R\{Q_2/X\} \xrightarrow{Y^+} \delta$.

Supposons $R\{Q_1//X\} \xrightarrow{Y^-} \delta$. Comme pour le cas précédent, on a forcément $R\{Q_2/X\} \xrightarrow{Y^-} \delta$.

On utilise des arguments symétriques pour passer des transitions de $R\{Q_2/X\}$ à celles de $R\{Q_1/X\}$. On peut donc conclure que $\mathcal{R}$ est une bisimulation forte, ce qui termine la preuve de la consistance de l'axiome [rec2].

Pour les axiomes [rec3] et [rec4], il faut à nouveau prouver que des relations sont des bisimulations fortes. Nous nous contentons de présenter une preuve informelle de leur consistance :

- Les modèles de $P + X$ et P ne peuvent différer que par des transitions par X^+ ou X^- de leur état initial : P peut n'avoir pas de transition par X^+, alors que $P + X$ en a une. À l'inverse, P peut avoir une transition par X^-, ce qui n'est pas le cas pour $P + X$. Or le comportement de $\mathtt{rec}X \cdot P + X$ et $\mathtt{rec}X \cdot P$ est indépendant de ces transitions. Les deux termes sont donc équivalents.
- Le même raisonnement est valable pour $\lfloor P + X \rfloor Q$ et $\lfloor P \rfloor Q$: le premier a une transition par X^- que le second peut ne pas posséder. On en conclut que les deux termes $\mathtt{rec}X \cdot \lfloor P + X \rfloor Q$ et $\mathtt{rec}X \cdot \lfloor P \rfloor Q$ sont équivalents.

La preuve de consistance des axiomes de $\mathcal{P}_t^r$ est à présent terminée.

Annexe B

Complétude de l'axiomatisation de $\mathcal{P}_t^r$

Cette annexe présente les preuves de la proposition définissant la forme canonique des termes de $\mathcal{P}_t^r$, et du théorème de complétude 2.6.

B.1 Forme canonique

L'extension à $\mathcal{P}_t^r$ de la proposition 2.5 est la suivante :

Proposition B.1

Soit P un terme de $\mathcal{P}_t^r$, dont les variables libres sont dans $\overline{Y} = Y_1, \ldots, Y_p$. Il existe $n \geq 1$ et des termes P_1, ..., P_n de $\mathcal{P}_t^r$, ayant leurs variables libres dans $\overline{Y}$, tels que $P \equiv P_1$ et chaque P_i est solution d'une équation dans une et une seule des formes [1] et [2] définies par :

$$[1] \quad P_i \equiv \sum_{b \in B_i} a_b \, P_{b_i} \; + \; \sum_{c \in C_i} Y_c \qquad\qquad b_i \in [1,n],\; C_i \subseteq [1,p]$$

$$[2] \quad P_i \equiv \left[\sum_{d \in D_i} a_d \, P_{d_i} \; + \; \sum_{e \in E_i} Y_e \right] P_{f_i} \; + \; \sum_{g \in G_i} Y_g \qquad d_i, f_i \in [1,n],\; E_i, G_i \subseteq [1,p],\; E_i \cap G_i = \emptyset$$

Selon nos conventions, si B_i ou D_i est vide, le terme $\sum$ correspondant est $\mathbf{0}$, et si C_i, E_i ou G_i est vide, le terme $\sum$ correspondant est δ.

□

Preuve. Remarquer d'abord que $\mathbf{0}$ est en forme [1], avec B et C vide. La preuve utilise de manière intensive le processus δ. On constate qu'il est équivalent à un P_j en forme [2] où D_j, E_j et G_j sont vides : $P_j \equiv \lfloor 0 \rfloor P_j$. Nous pouvons donc toujours l'ajouter sans problème à un ensemble de P_i.

Nous raisonnons par induction sur la structure du terme P. Nous n'indiquons pas toujours les axiomes employés lorsqu'ils sont évidents.

1. $P = X$. Alors X doit appartenir à $\overline{Y}$. Soit $P_1 \stackrel{\text{def}}{=} \lfloor 0 \rfloor \delta \; + \; X$. P_1 est en forme [2], et de plus

$$\begin{aligned} P_1 &\equiv \delta + X \\ &\equiv X \end{aligned}$$

2. $P = a\,Q$. Les variables libres de P et Q sont les mêmes. Donc, par induction, il existe Q_1, Q_m dont les variables libres sont dans $\overline{Y}$, chacun d'entre eux étant solution d'une équation d'une des deux formes, et tels que $Q \equiv Q_1$. Si on pose $P_1 \stackrel{\text{def}}{=} a\,Q_1$, alors $P \equiv P_1$ et P_1 est solution d'une équation en forme [1].
3. $P = Q + R$. Les variables libres de Q et R sont dans $\overline{Y}$. Par induction, il existe donc Q_1, $\ldots$, Q_m (resp. R_1, $\ldots$, R_q) dont les variables libres sont dans $\overline{Y}$, solutions d'équations dans une des deux formes, et tels que $Q \equiv Q_1$ (resp. $R \equiv R_1$). Nous avons donc $P \equiv Q_1 + R_1$. Posons $Q_{m+1} \stackrel{\text{def}}{=} R_{q+1} \stackrel{\text{def}}{=} \delta$. Nous définissons, pour $i' \in [1, m+1]$ et $i'' \in [1, q+1]$, le terme

$$P_{(i',i'')} \stackrel{\text{def}}{=} Q_{i'} + R_{i''}$$

Noter que

$$P_{(i',q+1)} = Q_{i'} \quad \text{et} \quad P_{(m+1,i'')} = R_{i''} \qquad (1)$$

Dans les équations des Q_i, nous pouvons donc substituer à chacun des Q_j le terme $P_{(j,q+1)}$. Nous obtenons alors un ensemble d'équations en forme requise pour les $P_{(i',q+1)}$, où n'interviennent que des $P_{(i',i'')}$. Le même raisonnement vaut pour les $P_{(m+1,i'')}$.
Nous prouvons maintenant que pour $i' \in [1, m]$ et $i'' \in [1, q]$, $P_{(i',i'')}$ est solution d'une équation d'une des deux formes, où seuls de tels termes apparaissent. Pour cela, nous considérons les formes des équations de $Q_{i'}$ et $R_{i''}$.

- Si les deux équations sont en forme [1], les identités (1) mènent directement à une équation en forme [1] pour $P_{(i',i'')}$.
- Si $Q_{i'} \equiv \sum_{b' \in B'_{i'}} a_{b'}\, Q_{b'_{i'}} + \sum_{c' \in C'_{i'}} Y_{c'}$ et

  $R_{i''} \equiv \left[\sum_{d'' \in D''_{i''}} a_{d''}\, R_{d''_{i''}} + \sum_{e'' \in E''_{i''}} Y_{e''} \right] R_{f''_{i''}} + \sum_{g'' \in G''_{i''}} Y_{g''}$, les axiomes [+5] et [+6] permettent d'obtenir

$$P_{(i',i'')} \equiv \sum_{b' \in B'_{i'}} a_{b'}\, P_{(b'_{i'},p+1)} + \sum_{d'' \in D''_{i''}} a_{d''}\, P_{(m+1,d''_{i''})} + \sum_{j \in C'_{i'} \cup E''_{i''} \cup F''_{i''}} Y_j$$

  qui est une équation en forme [1].
- Le cas symétrique mène au même résultat.
- Si $Q_{i'}$ et $R_{i''}$ satisfont une équation en forme [2], alors on peut utiliser les axiomes [+7] et [+8] afin d'obtenir pour $P_{(i',i'')}$ une équation en forme [2] (l'axiome [+7] sert à éliminer du délai les variables qui se trouvent en dehors).

Nous terminons le cas du choix non déterministe en remarquant que $P \equiv P_{(1,1)}$.

4. $P = \lfloor Q \rfloor R$. Les variables libres de Q et R sont dans $\overline{Y}$. Par induction, il existe donc Q_1,$\ldots$,Q_m et R_1,$\ldots$,R_q satisfaisant les conditions de la propriété, et tels que $Q \equiv Q_1$ et $R \equiv R_1$. Nous avons de plus $P \equiv \lfloor Q_1 \rfloor R_1$. Il faut trouver un P_1 équivalent à P et solution d'une équation dans une des deux formes. Pour cela, nous considérons l'équation satisfaite par Q_1 :

- $Q_1 \equiv \sum_{b \in B1} a_b\, Q_{b_1} + \sum_{c \in C_1} Y_c$. Dans ce cas, P_1 est défini comme solution de l'équation

$$P_1 \equiv \left[\sum_{b \in B_1} a_b\, Q_{b_1} + \sum_{c \in C_1} Y_c \right] R_1$$

  qui est en forme [2].

- $Q_1 \equiv \left\lfloor \sum_{d\in D_1} a_d\, Q_{d_1} \;+\; \sum_{e\in E_1} Y_e \right\rfloor Q_{f_1} \;+\; \sum_{g\in G_1} Y_g$. Nous définissons alors P_1 comme solution de l'équation en forme [2]

$$P_1 \equiv \left\lfloor \sum_{d\in D_1} a_d\, Q_{d_1} \;+\; \sum_{e\in E_1} Y_e \;+\; \sum_{g\in G_1} Y_g \right\rfloor R_1$$

  Les axiomes [+7] et [⌊⌋] assurent alors que $P \equiv P_1$.

5. $P = Q\|R$. P appartenant à $\mathcal{P}_t^r$, c'est un terme régulier ; il n'a donc ici pas de variable libre, pas plus que Q ou R. Par induction, il existe $Q_1,\ldots,Q_m$ et $R_1,\ldots,R_q$ fermés, tels que $Q \equiv Q_1$ et $R \equiv R_1$, et les Q_i et R_i sont solution d'une équation en forme [1] ou [2]. Puisqu'ils sont fermés, aucune variable n'apparaît dans les équations. De plus, nous avons $P \equiv Q_1\|R_1$.

   Nous définissons, pour $i' \in [1,m]$ et $i'' \in [1,q]$, le terme $P_{(i',i'')}$ par

$$P_{(i',i'')} \stackrel{\text{def}}{=} Q_{i'}\|R_{i''}$$

   En étudiant les équations de $Q_{i'}$ et $R_{i''}$, nous montrons que $P_{(i',i'')}$ est solution d'une équation d'une des deux formes.

   - Si $Q_{i'} \equiv \sum_{b'\in B'_{i'}} a_{b'}\, Q_{b'_{i'}}$ et $R_{i''} \equiv \sum_{b''\in B''_{i''}} a_{b''}\, R_{b''_{i''}}$, alors

$$\begin{aligned} P_{(i',i'')} &\equiv \sum_{b'\in B'_{i'}} a_{b'}\,(Q_{b'_{i'}}\|R_{i''}) + \sum_{b''\in B''_{i''}} a_{b''}\,(Q_{i'}\|R_{b''_{i''}}) + \sum_{\substack{b'\in B'_{i'},\, b''\in B''_{i''}\\ a_{b'}|a_{b''}\neq\bot}} (a_{b'}|a_{b''})\,(Q_{b'_{i'}}\|R_{b''_{i''}}) \\ &\equiv \sum_{b'\in B'_{i'}} a_{b'}\,P_{(b'_{i'},i'')} + \sum_{b''\in B''_{i''}} a_{b''}\,P_{(i',b''_{i''})} + \sum_{\substack{b'\in B'_{i'},\, b''\in B''_{i''}\\ a_{b'}|a_{b''}\neq\bot}} (a_{b'}|a_{b''})\,P_{(b'_{i'},b''_{i''})} \end{aligned}$$

     (axiomes [||, ⟦1, ⟦2 ou ⟦3, |2, |3 ou |4, |5])

   - Si $Q_{i'} \equiv \left\lfloor \sum_{d'\in D'_{i'}} a_{d'}\, Q_{d'_{i'}} \right\rfloor Q_{f'_{i'}}$ et $R_{i''} \equiv \sum_{b''\in B''_{i''}} a_{b''}\, R_{b''_{i''}}$, alors

$$\begin{aligned} P_{(i',i'')} &\equiv \sum_{d'\in D'_{i'}} a_{d'}\,(Q_{d'_{i'}}\|R_{i''}) + \sum_{b''\in B''_{i''}} a_{b''}\,(Q_{i'}\|R_{b''_{i''}}) + \sum_{\substack{d'\in D'_{i'},\, b''\in B''_{i''}\\ a_{d'}|a_{b''}\neq\bot}} (a_{d'}|a_{b''})\,(Q_{d'_{i'}}\|R_{b''_{i''}}) \\ &\equiv \sum_{d'\in D'_{i'}} a_{d'}\,P_{(d'_{i'},i'')} + \sum_{b''\in B''_{i''}} a_{b''}\,P_{(i',b''_{i''})} + \sum_{\substack{d'\in D'_{i'},\, b''\in B''_{i''}\\ a_{d'}|a_{b''}\neq\bot}} (a_{d'}|a_{b''})\,P_{(d'_{i'},b''_{i''})} \end{aligned}$$

     (axiomes [||, ⟦4, ⟦2 ou ⟦3, |1, |2, |6, |3 ou |4, |5])

   - Le cas symétrique se traite de la même manière.

   - Si $Q_{i'} \equiv \left\lfloor \sum_{d'\in D'_{i'}} a_{d'}\, Q_{d'_{i'}} \right\rfloor Q_{f'_{i'}}$ et $R_{i''} \equiv \left\lfloor \sum_{d''\in D''_{i''}} a_{d''}\, R_{d''_{i''}} \right\rfloor R_{f''_{i''}}$, alors

$$P_{(i',i'')} \equiv \left\lfloor \begin{array}{l} \sum_{d'\in D'_{i'}} a_{d'}\,(Q_{d'_{i'}}\|R_{i''}) + \sum_{d''\in D''_{i''}} a_{d''}\,(Q_{i'}\|R_{d''_{i''}}) + \\ \sum_{\substack{d'\in D'_{i'},\, d''\in D''_{i''}\\ a_{d'}|a_{d''}\neq\bot}} (a_{d'}|a_{d''})\,(Q_{d'_{i'}}\|R_{d''_{i''}}) \end{array} \right\rfloor (Q_{f'_{i'}}\|R_{f''_{i''}})$$

$$\equiv \left[\begin{array}{l} \displaystyle\sum_{d' \in D'_{i'}} a_{d'} \, P_{(d'_{i'}, i'')} + \sum_{d'' \in D''_{i''}} a_{d''} \, P_{(i', d''_{i''})} + \\ \displaystyle\sum_{\substack{d' \in D'_{i'}, d'' \in D''_{i''} \\ a_{d'} | a_{d''} \neq \bot}} (a_{d'} | a_{d''}) \, P_{(d'_{i'}, d''_{i''})} \end{array} \right] P_{(f'_{i'}, f''_{i''})}$$

(axiomes $[\|, \llbracket 5, \llbracket 1, \llbracket 2 \text{ ou } \llbracket 3, |7, |2, |3 \text{ ou } |4, |5, +7]$)

Dans tous les cas, nous obtenons une équation dans une des deux formes requises. On termine en constatant que $P \equiv P_{(1,1)}$.

6. $P = \partial_H(Q)$. Par hypothèse, P est régulier, donc fermé dans ce cas. En conséquence, ni P ni Q n'a de variable libre. Par induction, il existe donc $Q_1, \ldots, Q_m$ satisfaisant les conditions de la propriété, où les équations sont sans variable, tels que $Q \equiv Q_1$.

   Pour chaque i entre 1 et m, on définit

$$P_i \stackrel{\text{def}}{=} \partial_H(Q_i)$$

   On constate alors facilement que les P_i sont solutions d'équations dans une des formes [1] ou [2], et que $P \equiv P_1$.

7. $P = Q \triangledown R$. Les variables libres de R sont dans $\overline{Y}$, et Q n'en a pas, puisque P est régulier. Par induction, il existe $Q_1, \ldots, Q_m$ et $R_1, \ldots, R_q$ solutions d'équations en forme [1] ou [2], tels que $Q \equiv Q_1$ et $R \equiv R_1$. De plus, les variables libres des R_i sont dans $\overline{Y}$, et les Q_i n'en ont pas. Pour tout i entre 1 et m, nous définissons

$$P_i \stackrel{\text{def}}{=} Q_i \triangledown R_1$$

   Nous montrons que chaque P_i est solution d'une équation d'une des deux formes, où n'interviennent que des P_j et des R_k. Pour cela, nous distinguons les cas en fonction de l'équation satisfaite par Q_i

   - Si $Q_{i'} \equiv \displaystyle\sum_{b' \in B'_{i'}} a_{b'} \, Q_{b'_{i'}} + \sum_{c' \in C'_{i'}} \xi^{n_{c'}+1} \, Q_{c'_{i'}}$, où les $a_{b'}$ ne sont pas dans Ξ, alors

$$\begin{aligned} P_i &\equiv \sum_{b' \in B'_{i'}} a_{b'} \, (Q_{b'_{i'}} \triangledown R_1) + \sum_{c' \in C'_{i'}} \xi^{n_{c'}} \, R_1 \\ &\equiv \sum_{b' \in B'_{i'}} a_{b'} \, P_{b'_{i'}} + \sum_{c' \in C'_{i'}} \xi^{n_{c'}} \, R_1 \end{aligned}$$

     (axiomes $[\triangledown 1, \triangledown 2 \text{ ou } \triangledown 3, \triangledown 4]$)

   - Si $Q_{i'} \equiv \left[\displaystyle\sum_{d' \in D'_{i'}} a_{d'} \, Q_{d'_{i'}} + \sum_{c' \in C'_{i'}} \xi^{n_{c'}+1} \, Q_{c'_{i'}} \right] Q_{f'_{i'}}$, où les $a_{d'}$ ne sont pas dans Ξ, alors

$$\begin{aligned} P_i &\equiv \left[\sum_{d' \in D'_{i'}} a_{d'} \, (Q_{d'_{i'}} \triangledown R_1) + \sum_{c' \in C'_{i'}} \xi^{n_{c'}} \, R_1 \right] (Q_{f'_{i'}} \triangledown R_1) \\ &\equiv \left[\sum_{d' \in D'_{i'}} a_{d'} \, P_{d'_{i'}} + \sum_{c' \in C'_{i'}} \xi^{n_{c'}} \, R_1 \right] P_{f'_{i'}} \end{aligned}$$

     (axiomes $[\triangledown 5, \triangledown 1, \triangledown 1 \text{ ou } \triangledown 2, \triangledown 3, \triangledown 4]$)

Dans les deux cas, on obtient une équation en forme [1] ou [2], où n'interviennent que les P_i et R_1. Finalement, $P \equiv P_1$.

8. $P = \texttt{rec}X \cdot Q$. Les variables de Q sont dans $\langle \overline{Y}, X \rangle$. Par induction, il existe donc $Q_1,\ldots,Q_m$ solutions d'équations d'une des deux formes, dont les variables libres sont dans $\langle \overline{Y}, X \rangle$, et tels que $Q \equiv Q_1$. Puisque $\equiv$ est une congruence, nous avons $P \equiv \texttt{rec}X \cdot Q_1$.
Nous distinguons deux cas, selon l'équation satisfaite par Q_1.
   - $Q_1 \equiv \sum_{b \in B_1} a_b\, Q_{b_1} \;+\; \sum_{c \in C_1} Y_c\ \{+X\}$ ($\{+X\}$ signifie que X est présent ou non comme élément du choix non déterministe).
   On définit alors $Q_1' \stackrel{\text{def}}{=} \sum_{b \in B_1} a_b\, Q_{b_1} \;+\; \sum_{c \in C_1} Y_c$. Donc, soit $Q_1 \equiv Q_1'$, soit $Q_1 \equiv Q_1' + X$.
   En utilisant les axiomes [rec3] et [rec1], on obtient alors $P \equiv Q_1'[P/X]$, c'est-à-dire :
   $$P \equiv \sum_{b \in B_1} a_b\, Q_{b_1}[P/X] \;+\; \sum_{c \in C_1} Y_c$$
   Pour chaque i entre 1 et m, posons
   $$P_i \stackrel{\text{def}}{=} Q_i[P/X]$$
   Remarquer que $P \equiv P_1$. Les variables libres de chaque P_i sont dans $\overline{Y}$; nous montrons maintenant qu'il est solution d'une équation d'une des deux formes, où seuls des P_j apparaissent, en considérant la forme de l'équation satisfaite par Q_i, et selon la présence et la position de la variable X dans cette équation.
   – Si $Q_i \equiv \sum_{h \in H_i} a_h\, Q_{h_i} \;+\; \sum_{l \in L_i} Y_l\ \{+X\}$, alors
   $$P_i \equiv \sum_{h \in H_i} a_h\, P_{h_i} \;+\; \sum_{l \in L_i} Y_l \left\{ + \sum_{b \in B_1} a_b\, P_{b_1} \;+\; \sum_{c \in C_1} Y_c \right\}$$
   qui est une équation de la forme [1].
   – Si $Q_i \equiv \left[\sum_{h \in H_i} a_h\, Q_{h_i} \;+\; \sum_{l \in L_i} Y_l\ \{+X\} \right] Q_{m_i} \;+\; \sum_{k \in K_i} Y_k$, alors
   $$P_i \equiv \left[\sum_{h \in H_i} a_h\, Q_{h_i} \;+\; \sum_{l \in L_i} Y_l \left\{ + \sum_{b \in B_1} a_b\, P_{b_1} \;+\; \sum_{\substack{c \in C_1 \\ Y_c \notin \{Y_k\}}} Y_c \right\} \right] P_{m_i} \;+\; \sum_{k \in K_i} Y_k$$
   qui est une équation en forme [2]. Noter que dans la somme des Y_c, on peut éliminer les variables présentes parmi les Y_k grâce à l'axiome [+8].
   – Si $Q_i \equiv \left[\sum_{h \in H_i} a_h\, Q_{h_i} \;+\; \sum_{l \in L_i} Y_l \right] Q_{m_i} \;+\; \sum_{k \in K_i} Y_k \;+\; X$, alors
   $$P_i \equiv \sum_{h \in H_i} a_h\, Q_{h_i} \;+\; \sum_{l \in L_i} Y_l \;+\; \sum_{k \in K_i} Y_k \;+\; \sum_{b \in B_1} a_b\, P_{b_1} \;+\; \sum_{c \in C_1} Y_c$$
   qui est une équation en forme [1] (le délai disparaît grâce aux axiomes [+5] et [+6]).

- $Q_1 \equiv \left\lfloor \sum_{d\in D_1} a_d\, Q_{d_1} \;+\; \sum_{e\in E_1} Y_e\ \{+X\} \right\rfloor Q_{f_1} \;+\; \sum_{g\in G_1} Y_g\ \{+X\}$ (X peut être présent ou non, mais au plus à une des deux positions indiquées).

  On définit cette fois $Q_1' \equiv \left\lfloor \sum_{d\in D_1} a_d\, Q_{d_1} \;+\; \sum_{e\in E_1} Y_e \right\rfloor Q_{f_1} \;+\; \sum_{g\in G_1} Y_g$. Les axiomes [rec3], [rec4] et [rec1] permettent d'obtenir à nouveau $P \equiv Q_1'[P/X]$, c'est-à-dire :

$$P \equiv \left\lfloor \sum_{d\in D_1} a_1\, Q_{d_1}[P/X] \;+\; \sum_{e\in E_1} Y_e \right\rfloor Q_{f_1}[P/X] \;+\; \sum_{g\in G_1} Y_g$$

  Pour $i \in [1,m]$, on pose

$$\begin{aligned} P_i &\stackrel{\text{def}}{=} Q_i[P/X] \\ R_i &\stackrel{\text{def}}{=} P_i \;+\; P_{f_1} \end{aligned}$$

  Nous avons donc $P \equiv P_1$. Les P_i et R_i ont leurs variables libres dans $\overline{Y}$, et $P \equiv P_1$. Nous montrons que les P_i sont solutions d'équations en forme [1] ou [2], où n'interviennent que des P_j et R_j. Le même résultat est alors obtenu pour les R_i en utilisant le raisonnement effectué pour le choix non déterministe (cas 3 ci-dessus).

  – Si $Q_i \equiv \sum_{h\in H_i} a_h\, Q_{h_i} \;+\; \sum_{l\in L_i} Y_l\ \{+X\}$, alors

$$P_i \equiv \sum_{h\in H_i} a_h\, P_{h_i} \;+\; \sum_{l\in L_i} Y_l \left\{ + \sum_{d\in D_1} a_d\, P_{d_1} \;+\; \sum_{e\in E_1} Y_e \;+\; \sum_{g\in G_1} Y_g \right\}$$

  qui est une équation en forme [1].

  – Si $Q_i \equiv \left\lfloor \sum_{h\in H_i} a_h\, Q_{h_i} \;+\; \sum_{l\in L_i} Y_l\ \{+X\} \right\rfloor Q_{m_i} \;+\; \sum_{k\in K_i} Y_k$, alors

$$P_i \equiv \left\lfloor \sum_{h\in H_i} a_h\, Q_{h_i} \;+\; \sum_{l\in L_i} Y_l \left\{ + \sum_{d\in D_1} a_d\, P_{d_1} \;+\; \sum_{\substack{e\in E_1 \\ Y_e\notin\{Y_k\}}} Y_e \;+\; \sum_{\substack{g\in G_1 \\ Y_g\notin\{Y_k\}}} Y_g \right\} \right\rfloor P_{m_i} \;+\; \sum_{k\in K_i} Y_k$$

  qui est une équation en forme [2].

  – Si $Q_i \equiv \left\lfloor \sum_{h\in H_i} a_h\, Q_{h_i} \;+\; \sum_{l\in L_i} Y_l \right\rfloor Q_{m_i} \;+\; \sum_{k\in K_i} Y_k \;+\; X$, alors

$$P_i \equiv \left\lfloor \sum_{h\in H_i} a_h\, Q_{h_i} \;+\; \sum_{\substack{l\in L_i \\ Y_l\notin\{Y_g\}}} Y_l \;+\; \sum_{d\in D_1} a_d\, P_{d_1} \;+\; \sum_{\substack{e\in E_1 \\ Y_e\notin\{Y_k\}}} Y_e \right\rfloor R_{m_i} \;+\; \sum_{k\in K} Y_k \;+\; \sum_{g\in G_1} Y_g$$

  qui est une équation en forme [2] (nous rappelons que $R_{m_i} = Q_{m_i} + P_{f_1}$).

Dans chacun des cas, on obtient bien un ensemble d'équations en forme [1] ou [2] avec variables libres dans $\overline{Y}$, et $P \equiv P_1$, ce qui termine la preuve de la proposition. ■

B.2 Complétude

Le théorème de complétude énonce que si deux termes P et Q de $\mathcal{P}_t^r$ sont équivalents dans la relation $\sim$, alors ils le sont également dans la relation $\equiv$. Nous en présentons la preuve, qui est inspirée de [Mil84].

Supposons que les termes P et Q ont leurs variables libres dans $\overline{Y}$ (union des variables libres de P et de celles de Q). D'après la proposition B.1, il existe des termes $P_1,\ldots,\ P_n$ (resp. $Q_1,\ldots,\ Q_m$), dont les variables libres sont dans $\overline{Y}$, tels que $P \equiv P_1$ (resp. $Q \equiv Q_1$), et chaque P_i (resp. Q_i) est solution d'une équation dans une et une seule des formes [1] et [2], où n'apparaissent que des P_j (resp. des Q_j) et des Y_k de $\overline{Y}$.

On peut remarquer d'emblée que :

(1) l'équation de P_i (resp. Q_i) est en forme [1] $\iff$ P_i (resp. Q_i) n'a pas de transition par χ

(2) l'équation de P_i (resp. Q_i) est en forme [2] $\iff$ P_i (resp. Q_i) a une unique transition par χ

Soit $I \stackrel{\text{def}}{=} \{(i,i') \mid P_i \sim Q_{i'}\}$. Puisque $P \equiv P_1$ et $Q \equiv Q_1$ et puisque l'axiomatisation est consistante, nous avons $P_1 \sim Q_1$, c'est-à-dire $(1,1) \in I$.

De plus, si $(i,i') \in I$, P_i et $Q_{i'}$ ont les mêmes transitions. Donc, d'après **(1)** et **(2)**, P_i et $Q_{i'}$ sont solutions d'équations de la même forme. Nous partitionnons l'ensemble I en I_1 et I_2 selon la forme de ces équations.

- Pour $(i,i') \in I_1$, les équations sont

$$P_i \equiv \sum_{b \in B_i} a_b\, P_{b_i} \ + \sum_{c \in C_i} Y_c \qquad \text{et} \qquad Q_{i'} \equiv \sum_{b' \in B'_{i'}} a_{b'}\, Q_{b'_{i'}} \ + \sum_{c' \in C'_{i'}} Y_{c'}$$

  Pour chaque Y_c (resp. $Y_{c'}$), P_i (resp. $Q_{i'}$) a une transition par Y_c^- (resp. $Y_{c'}^-$). Puisque $P_i \sim Q_{i'}$, on a donc nécessairement $C_i = C'_{i'}$. De même, il existe une relation surjective totale $\mathcal{R}^1_{ii'}$ entre B_i et $B'_{i'}$, définie par

$$\mathcal{R}^1_{ii'} = \{(b,b') \mid a_b = a_{b'} \text{ et } P_{b_i} \sim Q_{b'_{i'}}\}$$

  Nous définissons alors les termes suivants, un pour chaque $(i,i') \in I_1$:

$$R_{ii'} \stackrel{\text{def}}{=} \sum_{(b,b') \in \mathcal{R}^1_{ii'}} a_b\, X_{b_i b'_{i'}} \ + \sum_{c \in C_i} Y_c$$

  où les $X_{jj'}$ sont des variables distinctes des éléments de $\overline{Y}$.

- Pour $(i,i') \in I_2$, les équations sont

$$P_i \equiv \left\lfloor \sum_{d \in D_i} a_d\, P_{d_i} \ + \sum_{e \in E_i} Y_e \right\rfloor P_{f_i} + \sum_{g \in G_i} Y_g \qquad \text{et} \qquad Q_i \equiv \left\lfloor \sum_{d' \in D'_{i'}} a_{d'}\, Q_{d'_{i'}} \ + \sum_{e' \in E'_{i'}} Y_{e'} \right\rfloor Q_{f'_{i'}} + \sum_{g' \in G'_i} Y_{g'}$$

  P_i (resp. $Q_{i'}$) a une transition par Y_e^- (resp. $Y_{e'}^-$) pour chaque Y_e (resp. $Y_{e'}$), et par Y_g^+ (resp. $Y_{g'}^+$) pour chaque Y_g (resp. $Y_{g'}$). Puisque $P_i \sim Q_{i'}$, on en déduit que $E_i = E'_{i'}$ et $G_i = G'_{i'}$. De plus, il existe une relation surjective totale $\mathcal{R}^2_{ii'}$ entre D_i et $D'_{i'}$, définie par

$$\mathcal{R}^2_{ii'} = \{(d,d') \mid a_d = a_{d'} \text{ et } P_{d_i} \sim Q_{d'_{i'}}\}$$

Nous définissons, pour chaque $(i, i') \in I_2$, le terme

$$R_{ii'} \stackrel{\text{def}}{=} \left\lfloor \sum_{(d,d') \in \mathcal{R}^2_{ii'}} a_d \, X_{d_i d'_{i'}} \; + \; \sum_{e \in E_i} Y_e \right\rfloor x_{f_i f'_{i'}} \; + \; \sum_{g \in G_i} Y_g$$

où les $X_{jj'}$ sont des variables n'appartenant pas à $\overline{Y}$.

Nous obtenons donc un ensemble de termes $R_{ii'}$, à variables libres dans $\langle \overline{X}, \overline{Y} \rangle$, dans lesquels les $X_{jj'}$ sont gardées.

D'après la proposition 2.4, il existe alors des termes $\overline{S} = \{S_{ii'} \mid (i, i') \in I\}$ de $\mathcal{P}_t^r$, dont les variables libres sont dans $\overline{Y}$, tels que pour tout (i, i'),

$$S_{ii'} \equiv R_{ii'}[\overline{S}/\overline{X}]$$

Ces équations sont satisfaisables si on choisit $S_{ii'} = P_i$. En effet, selon sa forme, l'équation de $S_{ii'}$ devient alors

$$P_i \equiv \sum_{(b,b') \in \mathcal{R}^1_{ii'}} a_b \, P_{b_i} \; + \; \sum_{c \in C_i} Y_c \qquad \text{ou} \qquad P_i \equiv \left\lfloor \sum_{(d,d') \in \mathcal{R}^2_{ii'}} a_d \, P_{d_i} \; + \; \sum_{e \in E_i} Y_e \right\rfloor P_{f_i} \; + \; \sum_{g \in G_i} Y_g$$

Dans les deux cas, puisque $\mathcal{R}^1_{ii'}$ et $\mathcal{R}^2_{ii'}$ sont totales, les membres droit de l'équation diffèrent au plus par des éléments répétés (dans les sommes) des membres droits de l'équation satisfaite par P_i. De même, les équations sont satisfaisables lorsqu'on prend $S_{ii'} = Q_{i'}$. Cela est dû au fait que $\mathcal{R}^1_{ii'}$ et $\mathcal{R}^2_{ii'}$ sont surjectives.

L'affirmation d'unicité de la proposition 2.4 permet alors de conclure que $P_i \equiv Q_{i'}$ pour tout (i, i') dans I. En particulier, on a $P_1 \equiv Q_1$, et donc $P \equiv Q$, ce qui termine la preuve. ■

Annexe C

Additivité temporelle des modèles des graphes temporisés

Nous montrons que la sémantique opérationnelle des graphes temporisés garantit l'additivité des transitions temporelles, c'est-à-dire :

Proposition C.1

$$\forall N, N', \forall d, d' \in D^*, \forall \vec{v}, \vec{v'} :$$
$$\left(\exists N_1, \vec{v_1} : (N, \vec{v}) \xrightarrow[D]{d} (N_1, \vec{v_1}) \wedge (N_1, \vec{v_1}) \xrightarrow[D]{d'} (N', \vec{v'})\right) \iff (N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

□

Nous rappelons les règles de sémantique des graphes temporisés, en les numérotant :

$$(R1)\ \frac{\forall d' \leq d : \mathtt{act}(N)(\vec{v} + \vec{d'}) = \mathtt{tt}}{(N, \vec{v}) \xrightarrow[D]{d} (N, \vec{v} + \vec{d})} \qquad (R2)\ \frac{N \xrightarrow{a,c,f} N',\ (\mathtt{act}(N) \wedge c)(\vec{v}) = \mathtt{tt}}{(N, \vec{v}) \xrightarrow[D]{a} (N', f(\vec{v}))}$$

$$(R3)\ \frac{N \xrightarrow{\varepsilon,c,f} N',\ (N', f(\vec{v})) \xrightarrow[D]{a} (N'', \vec{v'}),\ (\mathtt{act}(N) \wedge c)(\vec{v}) = \mathtt{tt}}{(N, \vec{v}) \xrightarrow[D]{a} (N'', \vec{v'})}$$

$$(R4)\ \frac{N \xrightarrow{\varepsilon,c,f} N',\ (N', f(\vec{v})) \xrightarrow[D]{d} (N'', \vec{v'}),\ (\mathtt{act}(N) \wedge c)(\vec{v}) = \mathtt{tt}}{(N, \vec{v}) \xrightarrow[D]{d} (N'', \vec{v'})}$$

$$(R5)\ \frac{(N, \vec{v}) \xrightarrow[D]{d} (N', \vec{v'}),\ N' \xrightarrow{\varepsilon,c,f} N'',\ (N'', f(\vec{v'})) \xrightarrow[D]{d'} (N''', \vec{v''}),\ (\mathtt{act}(N') \wedge c)(\vec{v'}) = \mathtt{tt}}{(N, \vec{v}) \xrightarrow[D]{d+d'} (N''', \vec{v''})}$$

Dans cette annexe, nous faisons plusieurs preuves par récurrence sur $R(T)$, nombre minimal nécessaire d'applications de ces règles pour obtenir une transition T donnée. Si ce nombre est 1, alors la transition est obtenue par application d'une des règles $R1$ ou $R2$; s'il est plus grand que 1, alors la dernière règle appliquée est $R3$, $R4$ ou $R5$.

Nous prouvons successivement les deux directions de l'équivalence de la propriété C.1.

$\Rightarrow$ Nous raisonnons par récurrence sur $n = R(T_1) + R(T_2)$, où

$$T_1 \stackrel{\text{def}}{=} (N, \vec{v}) \xrightarrow[D]{d} (N_1, \vec{v_1}) \qquad \text{et} \qquad T_2 \stackrel{\text{def}}{=} (N_1, \vec{v_1}) \xrightarrow[D]{d'} (N', \vec{v'})$$

- Si $n = 2$, alors T_1 et T_2 sont obtenues par application de R_1. On a donc :

  $\forall d_1 \leq d : \mathtt{act}(N)(\vec{v} + \vec{d_1}),\ N_1 = N,\ \vec{v_1} = \vec{v} + \vec{d},\ \forall d_2 \leq d' : \mathtt{act}(N_1)(\vec{v_1} + \vec{d_2}),\ \vec{v'} = \vec{v_1} + \vec{d'}$

On en déduit immédiatement $\forall d_3 \leq d + d' : \mathtt{act}(N)(\vec{v} + \vec{d_3})$, et donc $(N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$.

- Supposons que le sens $\Rightarrow$ est prouvé pour tout couple de transitions T_1' et T_2' tel que $R(T_1') + R(T_2') < n$, et que $n > 2$. L'une au moins des transitions T_1 ou T_2 est obtenue par application de $R4$ ou $R5$. Nous examinons les quatre cas possibles :
  - T_1 est obtenue par application de $R4$. On a alors :

    $$N \xrightarrow{\varepsilon,c,f} N_1',\ (\mathtt{act}(N) \wedge c)(\vec{v}) = \mathtt{tt},\ \text{et}\ (N_1', f(\vec{v})) \xrightarrow[D]{d} (N_1, \vec{v_1})$$

    Pour cette dernière transition T_3, on a $R(T_3) < R(T_1)$, et donc $R(T_3) + R(T_2) < n$. Par hypothèse de récurrence, on obtient alors

    $$(N_1', f(\vec{v})) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

    On applique alors la règle $R4$, ce qui donne

    $$(N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

  - T_1 est obtenue par application de $R5$. On a alors :

    $$(N, \vec{v}) \xrightarrow[D]{d_1} (N_1', \vec{v_1'}),\ N_1' \xrightarrow{\varepsilon,c,f} N_1'',\ (\mathtt{act}(N_1') \wedge c)(\vec{v_1'}) = \mathtt{tt},\ (N_1'', f(\vec{v_1'})) \xrightarrow[D]{d_2} (N_1, \vec{v_1})$$

    et $d = d_1 + d_2$. Comme précédemment, on peut appliquer l'hypothèse de récurrence pour obtenir

    $$(N_1'', f(\vec{v_1'})) \xrightarrow[D]{d_2+d'} (N', \vec{v'})$$

    En réutilisant $R5$, du fait que $d_1 + d_2 + d' = d + d'$, on obtient finalement

    $$(N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

  - T_2 est obtenue par application de $R4$. C'est-à-dire :

    $$N_1 \xrightarrow{\varepsilon,c,f} N_1',\ (\mathtt{act}(N) \wedge c)(\vec{v_1}) = \mathtt{tt},\ \text{et}\ (N_1', f(\vec{v})) \xrightarrow[D]{d} (N', \vec{v'})$$

    On peut alors directement appliquer $R5$ à $(N, \vec{v})$ pour obtenir

    $$(N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

  - T_2 est obtenue par application de $R5$. On a alors :

    $$(N_1, \vec{v_1}) \xrightarrow[D]{(N_1', \vec{v_1'})} ,N_1' \xrightarrow{\varepsilon,c,f} N_1'',\ (\mathtt{act}(N_1') \wedge c)(\vec{v_1'}) = \mathtt{tt},\ (N_1'', f(\vec{v_1'})) \xrightarrow[D]{d_2} (N', \vec{v'})$$

    et $d' = d_1 + d_2$. On peut appliquer l'hypothèse de récurrence pour la première de ces transitions, ce qui donne

    $$(N, \vec{v}) \xrightarrow[D]{d+d_1} (N_1', \vec{v_1'})$$

    En utilisant $R5$, on obtient à nouveau

    $$(N, \vec{v}) \xrightarrow[D]{d+d_1+d_2} (N', \vec{v'})$$

    qui est le résultat voulu, puisque $d + d_1 + d_2 = d + d'$.

$\Leftarrow$ Nous raisonnons par récurrence sur $R(T)$, ou'

$$T = (N, \vec{v}) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

- $R(T) = 1$. T est alors obtenue par application de $R1$. On a donc :

  $$N' = N,\ \vec{v'} = \vec{v} + \vec{d} + \vec{d'},\ \text{et}\ \forall d'' \leq d + d', \mathtt{act}(N)(\vec{v} + \vec{d''}) = \mathtt{tt}$$

  On en déduit, d'une part

  $$\forall d'' \leq d, \mathtt{act}(N)(\vec{v} + \vec{d''}) = \mathtt{tt} \text{ et donc } (N, \vec{v}) \xrightarrow[D]{d} (N, \vec{v} + \vec{d})$$

  et d'autre part

  $$\forall d'' \leq d', \mathtt{act}(N)(\vec{v} + \vec{d} + \vec{d''}) = \mathtt{tt} \text{ et donc } (N, \vec{v} + \vec{d}) \xrightarrow[D]{d'} (N, \vec{v} + \vec{d} + \vec{d'}) = (N, \vec{v} + d \dot{+} d')$$

- Supposons $R(T) > 1$, et que le sens $\Leftarrow$ est prouvé pour toute transition T' telle que $R(T') < R(T)$. Nous distinguons deux cas, selon la dernière règle appliquée pour obtenir T :
  - Si T est obtenue par application de $R4$, alors on a

    $$N \xrightarrow{\varepsilon, c, f} N'', (\mathtt{act}(N) \wedge c)(\vec{v}) = \mathtt{tt}, (N'', f(\vec{v})) \xrightarrow[D]{d+d'} (N', \vec{v'})$$

    On applique l'hypothèse d'indution à cette dernière transition, pour obtenir :

    $$(N'', f(\vec{v})) \xrightarrow[D]{d} (N_1, \vec{v_1}) \wedge (N_1, \vec{v_1}) \xrightarrow[D]{d'} (N', \vec{v'})$$

    En utilisant la règle $R4$, on a donc aussi

    $$(N, \vec{v}) \xrightarrow[D]{d} (N_1, \vec{v_1})$$

  - Si T est obtenue par application de $R5$, alors

    $$(N, \vec{v}) \xrightarrow[D]{d_1} (N_1', \vec{v_1'}),\ N_1' \xrightarrow{\varepsilon, c, f} N_2',\ (\mathtt{act}(N_1') \wedge c)(\vec{v_1'}) = \mathtt{tt},\ (N_2', f(\vec{v_1'})) \xrightarrow[D]{d_2} (N', \vec{v'})$$

    avec $d_1 + d_2 = d + d'$. Nous distinguons encore trois cas, selon le rapport entre d_1 et d.
    - Si $d_1 = d$, alors $d_2 = d'$, et donc, d'une part

      $$(N, \vec{v}) \xrightarrow[D]{d} (N_1', \vec{v_1'})$$

      et d'autre part, en appliquant $R4$,

      $$(N_1', \vec{v_1'}) \xrightarrow[D]{d'} (N', \vec{v'})$$

    - Si $d_1 < d$, alors $d_2 = d_3 + d'$ et $d_1 + d_3 = d$. On peut alors appliquer l'hypothèse de récurrence pour obtenir

      $$(N_2', f(\vec{v_1'})) \xrightarrow[D]{d_3} (N_1, \vec{v_1}) \text{ et } (N_1, \vec{v_1}) \xrightarrow[D]{d'} (N', \vec{v'})$$

      On a donc une partie du résultat. De plus, la règle $R5$ nous donne alors

      $$(N, \vec{v}) \xrightarrow[D]{d_1 + d_3} (N_1, \vec{v_1})$$

      qui constitue la seconde partie du résultat, puisque $d = d_1 + d_3$.
    - Si $d_1 > d$, alors $d_1 = d + d_3$ et $d' = d_3 + d_2$. On peut de nouveau appliquer l'hypothèse de récurrence pour obtenir

      $$(N, \vec{v}) \xrightarrow[D]{d} (N_1, \vec{v_1}) \text{ et } (N_1, \vec{v_1}) \xrightarrow[D]{d_3} (N_1', \vec{v_1'})$$

      La seconde partie du résultat est alors obtenue en appliquant la règle $R5$, qui nous donne

      $$(N_1, \vec{v_1}) \xrightarrow[D]{d_3 + d_2} (N', \vec{v'})$$

      qui est la transition voulue, puisque $d_3 + d_2 = d'$. ■

La preuve de l'additivité des transitions temporelles est à présent terminée. La règle de sémantique

$$(R6)\quad \frac{(N,\vec{v}) \xrightarrow[D]{d} (N',\vec{v'}),\ (N',\vec{v'}) \xrightarrow[D]{d'} (N'',\vec{v''})}{(N,\vec{v}) \xrightarrow[D]{d+d'} (N'',\vec{v''})}$$

est donc valide.

On constate facilement que la règle $R5$ peut être déduite de $R4$ et $R6$, et qu'on peut donc la supprimer si on ajoute $R6$.