Université de Montréal

ACTC – une algèbre de processus temporisée pour la spécification et vérification d'interfaces matérielles

par

Simona Gandrabur

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures

en vue de l'obtention du grade de

PhilosophiæDoctor (Ph.D.)

en informatique

Mars, 2000

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

ACTC – une algèbre de processus temporisée pour la spécification et vérification d'interfaces matérielles

présentée par:
Simona Gandrabur

a été évaluée par un jury composé des personnes suivantes:

président-rapporteur:	Mostapha Aboulhamid
directeur de recherche:	Eduard Cerny
membre du jury:	Pierre McKEnzie
examinateur externe:	André Arnold
représentant du doyen:	Paul Gauthier

Thèse acceptée le: 12 juin 2000

À mes enfants Sandrine et Alexandre

Sommaire

Le but de cet ouvrage est de définir une algèbre de processus ACTC pour formaliser les méthodes de spécification et vérification d'interfaces matérielles basées sur les *chronogrammes hiérarchiques* ou HAAD (de l'anglais *Hierarchical Annotated Action Diagrams*). Les HAAD sont un formalisme graphique qui permet de représenter des interfaces matérielles par des modules de base qui sont ensuite composés avec des opérateurs de composition hiérarchiques.

On dénote par *interface matérielle* tout système qui assure la communication de composants informatiques interconnectés, par des processus de communications spécifiques. La communication est établie par l'exécution d'événements, appelés *actions*. Une interface matérielle est un *système temps-réel*, c'est à dire un système qui doit produire ses résultats à l'intérieur d'intervalles de temps spécifiés par un ensemble de *contraintes temporelles*.

L'algèbre de processus ACTC est définie à partir d'un langage de spécification $\mathcal{LA}$ avec une syntaxe précise, dont les éléments sont appelés *termes*. Parmi les particularités de ce langage, spécifiquement conçu pour répondre aux besoins des développeurs d'interfaces matérielles, on note le support du dualisme des actions d'*entrée* et de *sortie*, des contraintes temporelles min, max, et conjonctives de type *supposition*, ou de type *engagement*, (en anglais "*assume/commit* contraints"), et du principe de *causalité* dans la sémantique de divers opérateurs de composition, tels que les *contraintes temporelles réactives*, la *composition parallèle temporelle avec communication*, le *choix retardé temporisé*, et l'opérateur d'*exception temporisé*. Dans le

langage $\mathcal{LA}$ les opérateurs de composition parallèle, avec ou sans communication, et de choix, simple ou retardé, sont d'arité arbitraire.

Nous associons au langage $\mathcal{LA}$ une *représentation équationnelle* et une *représentation opérationnelle*. La spécification équationelle permet de déduire des égalités de termes d'une manière purement syntaxique à partir d'un ensemble de règles de déduction et un ensemble **A** d'égalités de base, appelé *axiomatisation*. Autrement dit, on définit une *relation d'équivalence syntaxique* de termes, décidable à partir de la spécification équationnelle du langage. La représentation opérationnelle est définie par un ensemble de *règles de sémantique opérationnelle structurée* à la manière de Plotkin [74] et permet d'associer à tout terme un *modèle opérationnel* qui est un *système de transitions étiquetées* ou STE.

Nous définissons une *relation d'équivalence sémantique* ou *opérationnelle* $\approx$ sur le langage $\mathcal{LA}$, à partir de critères sémantiques définis sur les modèles opérationnels des termes et prouvons des propriétés sémantiques importantes du langage $\mathcal{LA}$, telles que la *congruence* de $\approx$ sur $\mathcal{LA}$ et l'*avancement du temps* dans les modèles opérationnels associés aux termes du langage.

Par la suite nous prouvons que l'axiomatisation **A** est *cohérente* avec $\approx$ sur tout le langage $\mathcal{LA}$. Autrement dit, si deux termes quelconques de $\mathcal{LA}$ sont équivalents syntaxiquement, ils sont aussi équivalents d'un point vue opérationnel. L'axiomatisation **A** n'est pas *complète*, c'est à dire que l'implication inverse n'est pas toujours vraie. Nous prouvons, par contre, la décidabilité de $\approx$ sur un sous-ensemble de termes $\mathcal{L}^r_\beta \subset \mathcal{LA}$ et donnons une procédure de type *vérification de modèle* (de l'anglais *model-checking*) de décision de l'équivalence $\approx$.

Mots clés: temps réel, spécification et vérification formelles, interface matérielle, algèbre de processus.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Je veux d'abord exprimer ma gratitude envers mon directeur de thèse, Eduard Cerny, pour m'avoir permis de bénéficier de ses conseils, de son expertise, et du support de son laboratoire de recherche durant mes études de baccalauréat, de maîtrise, et de doctorat.

Je tiens à remercier les fonds FCAR et CRSNG pour leur soutien financier tout au long de mes études.

Je remercie Karim Khordoc pour m'avoir initiée à la problématique de la spécification et vérification d'interfaces matérielles, Bachir Berkane pour m'avoir permis de découvrir le monde de la vérification formelle, et Abdelkader Dekdouk pour avoir approfondi mes connaissances sur les algèbres de processus.

Entreprendre des études doctorales est en soi une épreuve exigeante. Avoir deux bébés et travailler pendant les études doctorales aurait carrément été impossible sans le soutien moral et physique de beaucoup de personnes de mon entourage. Je tiens particulièrement à remercier mes parents, Magdalena et Ioan Gandrabur: sans leur confiance et leur aide, je n'aurais pas pu dire aujourd'hui: "mission accomplie". Merci à ma sœur, Ioana, d'avoir toujours été ma meilleure amie et d'avoir su me faire sourire.

Finalement, je remercie mon mari, Pierre L'Écuyer, de m'avoir servi de modèle d'auto-dépassement et de quête de l'excellence. Merci pour son amour et sa patience.

Je dédie cet ouvrage à mes enfants, Sandrine et Alexandre, qui ont porté sur leur petits dos si fragiles le gros du poids de mes longues journées de travail et d'études.

Chapitre 1

Introduction

Un système informatique est, en général, une interconnexion de différents composants communicants. La conception d'un tel système est alors un processus modulaire et hiérarchique. Les modules avec lesquels un composant est connecté forment l'*environnement* du composant. Les différents modules d'un système sont souvent conçus de manière isolée, sans connaissances précises sur le fonctionnement interne des autres composants. Lors de leur conception on doit alors faire un nombre d'hypothèses sur les propriétés de l'environnement et définir le comportement des modules en fonction de ces suppositions. L'*interface* d'un composant avec son environnement décrit l'ensemble de ces hypothèses ainsi que les réactions du composant face aux actions de l'environnement. Un *processus de communication* spécifie les règles d'échange d'informations du composant avec son environnement via l'*interface*.

Une interface est un *système temps-réel*, c'est à dire un système qui doit produire ses résultats à l'intérieur d'intervalles de temps spécifiés par un ensemble de *contraintes temporelles*.

Un système temps-réel est *correct* s'il est à la fois *fonctionnellement correct* et *temporellement correct*, c'est à dire qu'il produit des résultats corrects et qu'il satisfait aux contraintes temporelles spécifiées. La vérification d'une interface matérielle consiste à déterminer si les protocoles de communication des composants inter-connectés

sont fonctionnellement et temporellement *compatibles.*

Malgré l'importance des systèmes temps-réel dans la vie quotidienne, leurs méthodes de développement restent, en grande partie, assez primitives. Les concepteurs de tels systèmes passent souvent d'une spécification informelle directement à l'implantation dans un langage de programmation de systèmes temps-réels et la vérification se résume à la simulation non-exhaustive. Ces limitations sont dues d'une part à la grande complexité des systèmes à développer et, d'autre part au fait que l'utilisation des techniques de spécification et vérification formelles est non-triviale et que ces techniques sont très coûteuses en temps de calcul et espace mémoire.

Si, il y a quelques années seulement, l'utilisation des méthodes formelles dans le développement de systèmes temps-réel était encore relativement controversée dans l'industrie, aujourd'hui il est largement accepté que les techniques traditionnelles de spécification et vérification ne suffisent plus et que des méthodes formelles sont nécessaires. Ce besoin de formalisme se manifeste surtout dans l'industrie du développement de matériel informatique, où la complexité des systèmes croit à une vitesse exponentielle et les coûts associés à des "bogues" non-détectés ou détectés trop tard dans le processus de développement sont extrêmement importants.

L'avantage des méthodes formelles consiste en ce qu'elles peuvent garantir la "correctitude" d'un système, plus précisément elles peuvent garantir que les systèmes aient des propriétés précises. De plus, elles peuvent détecter les incohérences dans la spécification d'un système très tôt dans le processus de développement, minimisant ainsi les coûts associés à la modification et correction d'un système. Pour contourner le problème de la complexité on utilise des techniques hiérarchiques et modulaires ainsi que l'abstraction de données à différents niveaux.

Nos travaux s'incrivent dans la lignée de publications [32, 54, 21, 22, 31] qui présentent des techniques de spécification d'interfaces matérielles basées sur des *chronogrammes hiérarchiques annotés* ou HAAD (de l'anglais *Hierarchical Annotated Action Diagrams*). Les chronogrammes hiérarchiques sont un formalisme graphique qui

permet de représenter des interfaces matérielles par des modules de base, appelés *chronogrammes de base* ou *feuilles*, qui sont ensuite composés avec des opérateurs de composition hiérarchiques.

Une feuille est définie à partir d'un ensemble d'événements, appelés *actions*, certaines provenant de l'environnement, appelées actions *d'entrée*, et d'autres exécutées par le module, appelées actions *de sortie*. De plus, un ensemble de relations temporelles contraignent l'ordre et les temps d'occurrence des actions. On distingue deux catégories de contraintes temporelles: des contraintes *descriptives*, aussi appelées *suppositions* (en anglais *assume constraints*), et des contraintes *réactives*, ou des *engagements*, (en anglais *commit constraints*). Les contraintes temporelles descriptives définissent conjointement l'espace temporel qui correspond aux suppositions que le module fait sur le comportement temporel de son environnement. Les contraintes réactives définissent le comportement interne du module, ses réactions face aux stimuli extérieurs: le module s'engage à respecter toutes les contraintes réactives lorsque son environnement satisfait aux contraintes descriptives.

Différentes variantes de chronogrammes de base sont couramment utilisées dans la conception de systèmes numériques: bus, CPU, mémoires etc. Ces techniques sont, par contre, essentiellement informelles et les propriétés sémantiques de telles spécifications sont souvent ambigües.

Pour formaliser les chronogrammes hiérarchiques nous avons défini une *algèbre de processus temporiseé* ACTC. Une algèbre de processus est définie à partir d'un langage de spécification avec une syntaxe précise, dont les éléments sont appelés *termes*. Par rapport aux autres méthodes formelles, les algèbres de processus présentent l'avantage de combiner une représentation *équationnelle* avec une représentation *opérationnelle*. Une spécification *équationnelle* permet de déduire des égalités de termes d'une manière purement syntaxique à partir d'un ensemble de règles de déduction et d'un ensemble d'égalités de base, appelées *axiomes*. Une représentation *opérationnelle* permet d'associer à tout terme du langage de spécification un *modèle opérationnel* qui est

un *système de transitions étiquetées* ou STE (aussi appelé *machine à transitions* ou *machine à états*). Les systèmes de transitions sont définis sur un ensemble d'états et par une relation de transitions d'états. La notation $T \stackrel{a(v)}{\rightarrow} T'$ est utilisée pour désigner le fait qu'un STE transite d'un état T vers un état T' en exécutant une action a au temps v. Par abus de langage nous identifions souvent un terme avec le STE qui lui est associé. En modélisant les termes par des STE on peut formellement définir la sémantique du langage et spécifier des *relations d'équivalence opérationnelles* de termes. Typiquement, ces équivalences sont basées sur la relation d'*égalité de traces* ou sur la relation de *bisimulation forte* [62].

Nous avons essayé d'utiliser les algèbres de processus déjà existantes, telles que ATP [66, 83], les extensions temps-réel de ACP [13, 56, 14, 15], les variantes avec temps de CCS, TeCCS de [64], et différentes extensions temporisées de LOTOS [23, 38], mais nous nous sommes heurtés à des limitations sémantiques de ces algèbres. La plus importante parmi ces limitations est l'impossibilité d'exprimer le dualisme des actions d'entrée et de sortie et des contraintes temporelles du type supposition et/ou engagement. Nous avons, donc, développé un langage de spécification algébrique $\mathcal{LA}$, qui répond aux besoins spécifiques de la conception d'interfaces matérielles. Nous avons formellement défini la sémantique du langage par un ensemble de *règles de sémantique opérationnelle structurée* à la manière de Plotkin [74]. Par la suite nous avons défini une relation d'équivalence opérationnelle de termes $\approx$ sur le langage $\mathcal{LA}$, basée sur la notion de *bisimulation forte* de Milner [62], et défini un système partiel d'axiomes pour un sous-ensemble de langage $\mathcal{LA}$. Les caractéristiques distinctives du langage de spécification algébrique $\mathcal{LA}$ sont les suivantes:

- la temporisation explicite: on associe à toute action a une variable temporelle t_a qui désigne le temps d'occurrence de l'action, c'est à dire le temps où l'action a été exécutée. Les temps d'occurrence des actions peuvent alors être contraints par diverses relations temporelles définies sur les variables temporelles respectives;

- les contraintes temporelles linéaires de types conjonctif, min, et max;
- le support du paradigme suppositions/engagements dans la sémantique des contraintes temporelles;
- le support du dualisme entrée/sortie et du principe de *causalité* (défini dans [32]) dans la sémantique des divers opérateurs de composition;
- la définition d'opérateurs temporisés d'arité arbitraire pour la composition parallèle avec communication et le choix retardé;
- la définition d'un opérateur temporisé d'exception et d'un opérateur récursif temporisé.

L'expressivité du langage de spécification algébrique $\mathcal{LA}$ est telle qu'elle permet d'envisager le développement d'un outil de traduction automatique d'une spécification d'interface matérielle donnée par un HAAD vers un terme de l'algèbre et de vérifier formellement des propriétés importantes de cette spécification.

Les principaux résultats originaux présentés dans ce document sont les suivants:

- la preuve de la *congruence* de la relation d'équivalence sémantique de termes $\approx \subseteq \mathcal{LA} \times \mathcal{LA}$; une relation d'équivalence est une congruence si elle est préservée par la composition de termes. Formellement si $T, S \in \mathcal{LA}$ sont deux termes composés avec un même opérateur de composition f à partir de sous-termes $T_i, S_i \in \mathcal{LA}$ respectivement équivalents, c'est à dire

$$T = f[T_1, \ldots, T_m], \quad S = f[S_1, \ldots, S_m], \text{ et } \forall i \in \{1, \ldots, n\}\ T_i \approx S_i,$$

  alors $T \approx S$. Il existe des résultats théoriques [44] qui permettent de garantir que la bisimulation forte est une congruence sur un langage de termes, en autant que la sémantique du langage est donnée par un ensemble de règles de *sémantique opérationnelle structurée*, ou SOS, [74] qui sont dans un format

spécifique appelé format *panf* [44]. Malheureusement, ce format est trop restrictif pour nos besoins. Donc, nous avons été obligés de prouver la congruence à partir des définitions seulement;

- la preuve de la *monotonie temporelle* des séquences d'actions exécutées par un terme (aussi appelées *traces*); lorsqu'on définit les règles de sémantique opérationnelle du langage de termes d'une algèbre temporisée il est important d'assurer l'avancement du temps dans des séquences d'exécutions d'actions. Formellement, pour toute séquence de transitions:

$$T_1 \overset{a_1(v_1)}{\rightarrow} T_2 \overset{a_2(v_2)}{\rightarrow} T_3$$

il faut que la relation $v_1 \leq v_2$ soit satisfaite. Nous prouvons la monotonie temporelle de tout terme $T \in \mathcal{L}$, où $\mathcal{L} \in \mathcal{LA}$ est un sous-langage de termes, dits *accessibles*;

- la définition d'une axiomatisation $\mathbf{A}$ pour le langage $\mathcal{LA}$ et la preuve de la cohérence de $\mathbf{A}$ sur $\mathcal{LA}$. Une axiomatisation est dite *cohérente* si toute égalité de termes $T = S$ déduite à partir des axiomes par des inférences équationnelles, notée $\mathbf{A} \vdash T = S$, correspond à des termes équivalents sémantiquement:

$$\mathbf{A} \vdash T = S \text{ implique } T \approx S.$$

L'axiomatisation $\mathbf{A}$ n'est pas *complète*, c'est à dire que l'implication inverse n'est pas vraie. Mais la relation $\approx$ est quand même décidable sur un sous-langage de termes, noté $\mathcal{L}^r_\beta \subset \mathcal{LA}$.

- la preuve de la décidabilité de la relation $\approx$ sur le sous-ensemble de termes du langage $\mathcal{L}^r_\beta$: nous donnons une procédure de type *vérification de modèle* (de l'anglais *model-checking*) [33] de décision de l'équivalence de deux termes quelconques $T, S \in \mathcal{L}^r_\beta$.

Pour prouver que deux termes quelconques sont bisimilaires nous utilisons les techniques standard de définition de relations de bisimulation *fermées aux transitions* et l'*induction structurelle*. Dans les preuves d'égalités de traces nous appliquons la méthode de *construction de sous-ensembles* (de l'anglais *subset construction*), utilisée dans la déterminisation d'un automate non-déterministe. Plus spécifiquement, nous définirons les *systèmes de transitions étiquetées par parties*, ou $\wp$–STE, de la manière suivante: à un STE M quelconque avec un ensemble d'états $\mathbf{S}$ on associe un unique $\wp$–STE, noté $\wp(M)$, dont l'ensemble d'états est $\wp(\mathbf{S})$, c'est à dire l'ensemble sous-ensembles $\mathbf{S}$. Tout $\wp$–STE est *déterministe* et dans le théorème 4.8 nous prouvons que deux états quelconques $s_1, s_2 \in \mathbf{S}$ sont équivalents dans M du point de vue de traces si et seulement si les ensembles $\{s_1\}, \{s_2\} \subset \mathbf{S}$ sont fortement bisimilaires dans $\wp(M)$. Ainsi, pour prouver que s_1 et s_2 ont les mêmes traces il suffit d'appliquer les techniques standard de preuve de bisimulation pour les ensembles $\{s_1\}$ et $\{s_2\}$.

Ce document est organisé comme suit.

- Nous commençons par un survol succinct des méthodes formelles utilisées pour spécifier et vérifier des systèmes temps-réel en géneral, dans le chapitre 2, et des interfaces matérielles en particulier, dans le chapitre 3.

- Dans le chapitre 4 nous donnons les définitions formelles de spécifications équationnelles, des systèmes de transitions étiquetées, et des différentes relations d'équivalence de systèmes de transition rencontrées dans la littérature. Ensuite, nous présentons la définition des algèbres de processus. Finalement, nous introduisons le concept de système de transitions par parties et prouvons des résultats concernant la corrélation qui existe entre les différentes relations d'équivalence de systèmes de transitions simples et par parties.

- Dans le chapitre 5 nous définissons le langage de termes $\mathcal{LA}$ pour la spécification de chronogrammes hiérarchiques.

- Dans le chapitre 6 nous donnons l'ensemble de règles de sémantique opérationnelle

du langage $\mathcal{LA}$.

- Dans le chapitre 7 nous définissons le sous-langage de termes *accessibles* $\mathcal{L} \subset \mathcal{LA}$ et ses propriétés sémantiques et syntaxiques. L'algèbre de processus ACTC est définie sur le sous-langage $\mathcal{L}$. Ceci ne constitue pas, par contre, une limitation sémantique puisque le sous-langage $\mathcal{L}$ est suffisant pour spécifier tout chronogramme hiérarchique.

- Dans le chapitre 8 nous définissons la relation d'équivalence de termes $\approx$. La relation $\approx$ est une bisimulation forte avec deux restrictions supplémentaires, ajoutées pour assurer la congruence de $\approx$ sur $\mathcal{LA}$.

- Dans le chapitre 9 nous donnons la définition formelle de l'algèbre de processus ACTC; ensuite nous définissons l'axiomatisation **A** et prouvons la cohérence de **A**.

- Dans le chapitre 10 nous prouvons la décidabilité de $\approx$ sur le sous-ensemble de termes $\mathcal{L}^r_\beta \subset \mathcal{L}$.

- Finalement, dans le chapitre 11 nous concluons notre travail et nous décrivons des extensions et applications possibles de nos résultats.

- L'annexe contient les preuves de la majorité des propositions, lemmes, et théorèmes présentés dans ce document. Lorsqu'une preuve n'est pas nécessaire à la compréhension d'un résultat, elle sera présentée en annexe.

- Pour alléger la lecture de cet ouvrage, nous avons ajouté à la fin du document un index des notations et de la terminologie utilisées.

Chapitre 2

Méthodes formelles

Dans ce chapitre nous commençons par une présentation de la notion de *système temps-réel* et faisons ensuite un survol succinct des différentes méthodes formelles utilisées dans le développement de systèmes temps-réel. Nous classifions les méthodes formelles selon leur langage de spécification, leur modèle, et la technique d'analyse qu'elles utilisent. Pour des survols plus complets nous référons à [45, 47].

2.1 Systèmes temps-réel

Un système temps-réel est, typiquement, une composition de composants séquentiels communicants (*processus*), dont le comportement doit satisfaire aux contraintes temporelles strictes. Ces systèmes sont de plus en plus utilisés dans la vie quotidienne et ils font souvent partie des applications critiques, dont la fiabilité est cruciale.

Un système temps-réel est correct lorsqu'il est à la fois fonctionnellement et temporellement correct, c'est à dire lorsque l'ordonnancement d'actions est cohérent et lorsque les contraintes temporelles, auquelles l'occurrence des actions du système sont soumises, sont satisfaites. Pour une discussion plus approfondie du dualisme fonctionnalité-temps nous référons à [36, 43]. Pour la vérification de propriétés fonctionnelles, aussi appelées *qualitatives*, la notion de temps se résume à une notion d'ordonnance-

ment des actions. Dans la vérification temporelle, dite aussi *quantitative*, le temps sert surtout à déterminer les distances limites entre les actions. On doit aussi décider d'un modèle précis du temps, tels que le temps discret ou continu. Chacun de ces modèles a ses avantages et ses inconvénients: le temps discret est plus facile à manipuler, mais le temps dense est plus expressif. D'un point de vue algorithmique, par contre, les deux modèles sont de complexité comparable [4, 5], sous certaines conditions qui permettent la partition du domaine temporel en zones équivalentes.

Le développement d'applications temps-réel est souvent caractérisé par un manque de rigueur, due à l'absence de méthodes formelles. La sémantique des spécifications, les modèles choisis et les techniques de vérification utilisées ne sont pas formellement définis et portent souvent à confusion. La simulation est la méthode de vérification la plus courante. Un modèle d'un système temps-réel répond de diverses façons aux stimuli auquels il est soumis. La simulation consiste à observer le comportement du modèle lorsque soumis à des stimuli différents. Évidemment, cette approche ne peut pas garantir la validation d'un système, à moins de le soumettre exhaustivement à tous les stimuli possibles. La simulation exhaustive se heurte en pratique à la complexité des systèmes à vérifier qui rend le nombre total de stimuli possibles trop grand.

Pour éviter les problèmes que pose la simulation, voire la non-fiabilité absolue des résultats de cette forme de vérification, des méthodes plus formelles ont été développées.

Les *méthodes formelles* utilisent des *langages de spécification formels*, ainsi que des *techniques de modélisation formelles* pour décrire le comportement d'un système et des *techniques d'analyse formelles* pour vérifier si le comportement du système, ou d'une spécification du système, a des propriétés critiques.

Il est essentiel de détecter les problèmes de conception le plus tôt possible pour minimiser les coûts élevés associés aux échecs et modifications tardives. La spécification et la vérification de propriétés essentielles des spécifications sont donc des phases cruciales lors de la conception de systèmes. C'est typiquement lors de ces étapes que

des méthodes formelles sont appliquées.

2.2 Langages de spécification formels

Selon le formalisme de spécification on peut distinguer des méthodes *opérationnelles* ou bien *descriptives*. Avec un style opérationnel on décrit les règles de fonctionnement d'un système. Ces méthodes sont en général basées sur le modèle *système de transitions étiquetées*, ou *STE*, parfois aussi appelés *machines à états*, défini par les concepts d'*état* et de *transition*. Les méthodes descriptives définissent un système par ses propriétés; elles spécifient précisément ce qu'un système doit faire, et non pas comment cela se réalise. Ces méthodes sont typiquement basées sur des logiques temporelles. Les algèbres de processus combinent les deux approches, opérationnelle et descriptive.

Les méthodes de spécification diffèrent aussi par la notation formelle utilisée. Ainsi, il y a des notations *textuelles*, telles que les algèbres de processus ou les méthodes axées sur la logique, ou bien des notations *graphiques*, telles que Modechart [51], Timed Transition Model (TTM), [69, 71] ou les différentes versions temporisées des réseaux de Petri [72, 73, 61, 85, 86, 27].

2.3 Modèles formels temporisés

Nous allons présenter brièvement les modèles basés sur des *logiques temporelles* et les *automates temporisés*. Ensuite nous expliquerons plus en détail les modèles basés sur des *algèbres de processus temporisés*.

2.3.1 Les modèles basés sur des logiques temporelles

Des méthodes formelles descriptives sont typiquement associées à un modèle basé sur la logique et utilisent des techniques d'analyse basées sur des preuves de théorèmes:

le système est décrit par un ensemble d'axiomes écrits dans une logique de choix et ses propriétés, exprimées par des formules logiques, peuvent être déduites à partir de la spécification axiomatique. Nous donnons trois exemples de logiques développées pour des analyses temporelles : RTL (de l'anglais Real Time Logic)[50], TRIO [42], et ASTRAL [35]. La plupart des logiques temps-réel décrivent les systèmes en fonction d'événements, de points dans le temps où quelque chose de significatif se produit. Il existe aussi des logiques qui, au contraire, se concentrent sur des conditions dites *stables*, qui sont satisfaites pendant un intervalle de temps non-nul. Pour plus d'information sur ces logiques, appelées logiques d'intervalle, voir [3, 70].

2.3.2 Les automates temporisés

Les automates constituent le formalisme opérationnel principal basé sur le modèle du système de transitions étiquetées, (STE). Un STE est un triplet $(A, \mathbf{S}, \rightarrow)$ où $A = \{a, b, \ldots\}$ est un vocabulaire de symboles aussi appelés *actions*, $\mathbf{S} = \{s, s', \ldots\}$ un ensemble d'*états*, et une *relation de transition* $\rightarrow \subseteq \mathbf{S} \times A \times \mathbf{S}$. Un triplet $(s, a, s') \in \rightarrow$ est noté par $s \xrightarrow{a} s'$. On dit alors que le STE transite de l'état s vers l'état s' après avoir lu (ou exécuté) l'action a. Un *système de transitions étiquetées initialisé*, noté STEI, est un STE auquel on ajoute un état initial $i \in \mathbf{S}$.

Un automate est un tuplet $\text{At} = (M, \mathbf{I}, \mathbf{F})$ défini par un STE $M = (A, \mathbf{S}, \rightarrow)$, dit *sous-jacent* à At, par un sous-ensemble d'états $\mathbf{I} \subseteq \mathbf{S}$, appelés *états initiaux*, et $\mathbf{F} \subseteq \mathbf{S}$ un ensemble d'*états finaux* ou *acceptants*. On utilise aussi la notation explicite $\text{At}= (A, \mathbf{S}, \rightarrow, \mathbf{I}, \mathbf{F})$, pour désigner un automate. On peut alors étudier toutes les exécutions qui partent d'un état initial et mènent vers des états finaux.

On note par A^* l'ensemble de séquences finies et par A^ω les séquences infinies sur A. Notons par $A^\infty = A^* \cup A^\omega$ et appelons les séquences de A^∞ *mots* sur l'alphabet A. La théorie classique sur les automates se résume aux exécutions finies. Les extensions de ces automates aux exécutions infinies ont donné lieu aux ω-automates, qui se distinguent par leur définition d'un état acceptant. Ainsi, les états finaux peuvent

soit être omis complètement de la définition de l'automate, ou bien ils peuvent être définis comme des états que toute exécution visite un nombre infini de fois, ou encore comme états absorbants (à partir desquels il ne sort aucune transition), etc. Pour un résumé plus approfondi sur la théorie des ω-automates nous citons [79], et [57] pour l'application des ω-automates dans la vérification.

Les *automates temporisés* sont une extension des automates présentés ci-haut, par l'introduction d'un ensemble fini H de variables nommées *horloges*. Les horloges évoluent de manière synchronisée pendant les exécutions de l'automate. Elles peuvent être remises à zéro lors de chaque transition. On associe un ensemble de contraintes temporelles linéaires sur les horloges à chaque transition; ces contraintes définissent les conditions qui doivent être respectées pour que la transition puisse être exécutée. Ainsi, un automate temporisé AT est un 8-uplet $(A, \mathbf{S}, \rightarrow, \mathbf{I}, \mathbf{F}, H, \mu, \gamma)$, où A, $\mathbf{S}$, $\rightarrow$, $\mathbf{I}$, $\mathbf{F}$, et H sont tels que définis plus haut. La fonction μ définit l'ensemble d'horloges qui doivent être réinitialisés à zéro pour chaque transition. Ainsi, lors d'une transition quelconque $s \xrightarrow{a} s'$ les horloges de l'ensemble

$$\mu((s, a, s')) = \{h_1, \ldots, h_n\} \subseteq H$$

sont re-initialisées. La fonction γ associe à chaque transition les contraintes temporelles sur l'ensemble d'horloges qui doivent être respectées. Des nombreux travaux peuvent être cités pour des présentations plus détaillées de la théorie des automates temporisés ainsi que des applications des automates temporisés à des problèmes de vérification (voir la collection d'articles [63], [6, 8, 7], [9, 46]).

L'importance des automates est accentuée par le fait que certains problèmes exprimés à l'aide d'autres formalismes, tels que les logiques temporelles ou les algèbres de processus, peuvent être réduits à des problèmes sur les automates. De nombreux travaux soulignent la corrélation entre les logiques temporelles et les automates, nous citons [81] qui fait aussi un survol de ces résultats. Pour des traductions de spécifications algébriques vers des automates temporisés on peut citer [68, 22, 38, 37, 39].

2.3.3 Algèbres de processus temporisées

Une *algèbre de processus* est en même temps un formalisme descriptif et un formalisme opérationnel. Nous présentons ici la notion d'algèbre de processus d'une manière plutôt informelle et intuitive. Les définitions formelles des concepts introduits ici se trouvent au chapitre 4. Nous faisons ensuite un survol succinct des différentes algèbres de processus temporisées existantes. Pour des survols plus complets nous référons à [67, 56, 15].

Une *algèbre*, en général, $\mathcal{A} =< E, F >$ est définie par un ensemble d'éléments E et un ensemble F de fonctions f fermées dans E, c'est à dire $f : E^n \longrightarrow E$, où $n \geq 0$.

On dénote par *signature* un ensemble de *symboles de fonctions*, noté $\mathcal{F}$, muni d'une fonction d'*arité* ar : $\mathcal{F} \longrightarrow \mathbb{N}$. L'arité ar($\phi$) associe à chaque symbole de fonction $\phi \in \mathcal{F}$ un nombre naturel qui dénote le nombre d'arguments du symbole de fonction ϕ. Une *interprétation* de $\mathcal{F}$ dans F est une fonction $i : \mathcal{F} \longrightarrow F$ qui associe à chaque symbole de fonction $\phi \in \mathcal{F}$ une fonction $f = i(\phi) \in F$. À partir de la signature $\mathcal{F}$ et des éléments de l'ensemble E on peut définir syntaxiquement un *langage de spécification* $\mathcal{T}(\mathcal{F})$. Les éléments de $\mathcal{T}(\mathcal{F})$ sont appelés *termes* et sont notés $T, S, \ldots$. Le langage $\mathcal{T}(\mathcal{F})$ est appelé le *langage de termes clos* de la signature $\mathcal{F}$.

On dit qu'une algèbre $\mathcal{A}$ est *axiomatisée* lorsqu'on associe à $\mathcal{A}$ une *spécification équationnelle*, notée $\mathcal{E} = (\mathbf{A}, \mathbf{R})$. $\mathbf{A}$ est un ensemble d'équations de termes, appelées *axiomes*, et $\mathbf{R}$ un ensemble de *règles d'inférence*. À partir de la spécification équationnelle on peut faire des dérivations formelles, appelées *preuves*, d'égalités de termes. Si l'égalité $S = T$ peut être dérivée à partir des axiomes de $\mathbf{A}$ et en utilisant les règles d'inférence de $\mathbf{R}$, alors on écrit $\mathcal{E} \vdash S = T$. $\mathcal{E}$ induit une *relation d'équivalence équationnelle* de termes $\mathcal{R}_\mathcal{E} \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F})$. $\mathcal{R}_\mathcal{E}$ est la plus petite relation qui contient toutes les paires de termes dont l'égalité peut être prouvée dans $\mathcal{E}$.

Lorsqu'on veut modéliser algébriquement des *processus* via un langage de spéci-

fication de termes $\mathcal{T}(\mathcal{F})$, on commence par donner une *sémantique opérationnelle* aux termes du langage, c'est à dire on associe à chaque terme $T \in \mathcal{T}(\mathcal{F})$ un *graphe de processus*, noté graphe(T). graphe(T) est un STEI dont les états sont les termes de $\mathcal{T}(\mathcal{F})$ accessibles à partir de l'état initial T via une relation de transition d'états $\rightarrow$. La relation $\rightarrow \subseteq \mathcal{T}(\mathcal{F}) \times A \times \mathcal{T}(\mathcal{F})$ est étiquetée par des actions de l'ensemble d'actions $A = \{a, b, \ldots\}$.

On définit ensuite, selon différents critères sémantiques, une *relation d'équivalence opérationnelle* de termes, notée $\mathcal{R}_{\mathcal{O}}$. Deux termes sont considérés comme équivalents s'ils ont le même comportement, c'est à dire lorsque pour chaque pas dans l'évolution d'un terme il existe un pas similaire dans le deuxième terme et vice-versa. Ces relations sont typiquement basées soit sur la relation de *bisimulation* ([62]) ou d'*égalité de traces*. On modélise un processus par une *classe d'équivalence* modulo la relation $\mathcal{R}_{\mathcal{O}}$, c'est à dire par l'ensemble de tous les termes équivalents selon la relation d'équivalence sémantique $\mathcal{R}_{\mathcal{O}}$.

On dispose alors de deux modalités différentes pour spécifier des processus: l'une opérationnelle, par les graphes de processus, et une spécification équationnelle $\mathcal{E} = \langle \mathbf{A}, \mathbf{R} \rangle$. On dit que l'axiomatisation $\mathbf{A}$ de l'algèbre $\mathcal{A}$ est *cohérente* ou *saine* par rapport à la relation d'équivalence opérationnelle si $\mathcal{R}_{\mathcal{E}} \subseteq \mathcal{R}_{\mathcal{O}}$, donc si $(S, T) \in \mathcal{R}_{\mathcal{E}}$ implique $(S, T) \in \mathcal{R}_{\mathcal{O}}$, pour toute paire de termes $(S, T) \in \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F})$. Inversement, si $\mathcal{R}_{\mathcal{O}} \subseteq \mathcal{R}_{\mathcal{E}}$ on dit que l'axiomatisation est *complète* par rapport à la relation $\mathcal{R}_{\mathcal{O}}$.

À l'origine, les algèbres de processus, telles que CCS [62], CSP [49], et ACP [19], avaient pour but de spécifier et d'analyser des systèmes non-temporisés. Le temps est représenté implicitement par l'ordonnancement d'actions ou, dans un sens plus général, par la composition séquentielle de processus. Les propriétés temporelles exprimées sont, donc, de nature qualitative seulement.

Récemment, un nombre d'algèbres ont été développées qui représentent le temps de manière explicite et sont ainsi capables d'exprimer et analyser des propriétés

temporelles quantitatives des systèmes avec temps. Ces algèbres, appelées *temporisées*, sont soit des extensions d'algèbres non-temporisées existantes, ou bien ont été développées spécifiquement pour aborder des problèmes qualitatifs de temps, telles que ATP [66, 83] ou ACTC [22]. Dans la première catégorie nous ne retenons que quelques exemples représentatifs: les extensions temps-réel de ACP [13, 56, 14, 15], les variantes avec temps de CCS, TeCCS de [64], une variante avec temps de CSP, TSP [75], et des extensions temporisées de LOTOS [23, 38].

Les algèbres temporisées se distinguent par les différents modèles de temps sous-jacents et par l'expressivité de leur langage.

La plupart des formalismes de description de systèmes temporisés adoptent implicitement les conventions suivantes concernant la modélisation du temps:

- Le système est une composition de processus communicants. Chaque composant a une variable d'état, appelée *temps du processus*, définie sur un domaine temporel D avec une opération binaire $+$, qui est essentiellement l'addition de nombres non-négatifs.

- Le temps du système évolue de manière synchronisée dans tous les processus, c'est à dire le temps est avancé par une quantité d quelconque si et seulement si tous les composants le font simultanément.

Le premier aspect de modélisation temporelle selon lequel on peut classifier les différentes algèbres temporisées est la condition de transition d'un état à l'autre. Dans certains formalismes, dits *à deux phases*, un système peut évoluer d'un état à l'autre soit en laissant passer le temps, ou bien en exécutant une ou plusieurs actions. La séquence d'exécution est bi-phasée: dans la première phase les processus peuvent exécuter, indépendamment ou en coopération, un nombre fini d'actions. Dans la deuxième tous les processus se synchronisent pour laisser avancer le temps par une valeur finie ou infinie. Les deux phases s'alternent. Ce principe de fonctionnement a été adopté, par exemple, dans les différentes versions de ATP. L'alternative de ce

principe de fonctionnement est d'associer à chaque action une variable temporelle qui représente son temps d'occurrence. Les actions sont alors nommées *actions temporisées* . Cette modélisation est adoptée dans ACP_ρ [13, 56], dans TCS [64], dans [75] pour CSP, et dans l'algèbre ACTC que nous avons développée [22]. Ce modèle permet d'exprimer aisément des contraintes temporelles sur les temps d'occurrence des actions. Les transitions d'un état à l'autre se font seulement par l'exécution d'actions temporisées.

Le *domaine temporel D* peut être *dense*, par exemple $\mathbb{R}^+$ comme dans le cas de ACTC, ACP_ρ ou TSP. Une autre option consiste à segmenter le temps et indexer les segments par l'ensemble de nombres naturels. Le temps avance segment par segment et les actions ne peuvent être exécutées qu'à l'intérieur d'un segment. On dit alors que le domaine temporel est *discret*. Cette discrétisation du temps a été adoptée dans ATP, dans TeCCS et dans les versions discrètes de ACP [15].

Lorsque le temps des processus exprime la distance temporelle entre des actions consécutives, on dit que la temporisation est *relative*. Cette approche est surtout utilisée dans les algèbres avec fonctionnement bi-phasé. Le passage du temps exprime alors l'intervalle de temps qui est passé entre la dernière phase d'exécution d'actions et la suivante. Si, par contre, le temps mesure la distance par rapport à une référence globale, la temporisation est dite *absolue*. Utilisée par exemple, dans ACP_ρ et ACTC, cette temporisation permet d'exprimer des contraintes temporelles d'une manière naturelle par des ensembles d'inéquations linéaires sur les temps d'occurrence des actions.

Une analyse plus détaillée des propriétés des modèles temporels des algèbres de processus est présentée dans [67].

Dans le langage d'une algèbre de processus temporisée il y a des opérateurs temporels, qui expriment les contraintes temporelles quantitatives concernant les temps d'occurrence des actions, et les opérateurs définissant les propriétés qualitatives, ou séquentielles des systèmes, aussi appelés des opérateurs non-temporels.

En ce qui concerne les opérateurs non-temporels, toutes les algèbres ont un ensemble assez standard qui contient la composition séquentielle et parallèle (avec ou sans communication), un opérateur de récurrence, aussi appelé boucle, et un choix non-déterministe. Dans les définitions de ces opérateurs de base dans les différentes algèbres considérées il y a quelques variations qui ne sont pas essentielles à notre discussion. Pour plus de détails à ce sujet voir [67, 56].

L'algèbre ACTC présente deux nouveaux opérateurs qualitatifs temporisés: le *choix retardé* et l'opérateur *d'exception.* Le choix retardé est la composition de plusieurs processus, appelés *branches* ou *alternatives.* Cet opérateur, très utile dans la description d'interfaces matérielles, permet la composition de processus qui ont un comportement initial commun et qui se distinguent plus tard dans leur exécution, soit par les stimuli externes qu'ils attendent de leur environnement, ou bien par les actions internes qu'ils exécutent. Un choix retardé de plusieurs alternatives a alors une phase initiale d'exécution, qui correspond au comportement initial commun de toutes les branches, suivie par une phase qui est la suite du comportement d'une alternative (ou plusieurs alternatives, éventuellement résolues à une seule alternative) choisie en fonction des stimuli de l'environnement ou des actions internes spécifiées.

L'opérateur d'exception décrit un mécanisme de traitement d'exception, avec un comportement régulier, une condition d'exception, représentée par un processus, et un comportement exceptionnel qui est exécuté lorsque la condition d'exception est satisfaite (c'est à dire lorsque le processus d'exception finit son exécution avant le processus régulier). Notons qu'un opérateur de choix retardé binaire non-temporisé a été défini aussi dans [16].

Du point de vue des opérateurs temporels, les algèbres mentionnées ont des degrés de complexité différents. Elles permettent toutes d'exprimer la notion de retard ou décalage, soit par une valeur fixe (TCS, TSP, U-LOTOS [24]), par une valeur possiblement infinie (TeCCS, ATP), ou par une valeur quelconque choisie à l'intérieur d'un intervalle (ACP_ρ, ACTC).

Un opérateur plus général, qui permet de contraindre les temps d'occurrence des actions par un ensemble d'inéquations linéaires est défini dans [56] et l'ACTC. Dans l'algèbre de [56], par contre, le temps d'occurrence d'une action ne peut être contraint que par le temps d'occurrence d'actions qui se sont déjà produites dans le passé. Avec ACTC cette restriction n'existe pas.

Le paradigme des contraintes *descriptives/réactives* (en anglais *assume/commit*) [1], à notre connaissance, n'est supporté que dans ACTC. On dispose de deux opérateurs réactifs: l'opérateur Max exprimant les contraintes de type "au plus tard" et l'opérateur Min pour des contraintes de type "au plus tôt". Ces opérateurs servent à déterminer le comportement *réactif* d'une interface matérielle, c'est à dire les réactions internes d'un processus face aux stimuli externes auquels il est soumis.

2.4 Techniques d'analyse formelle

Il existe deux classes principales de techniques d'analyse formelle: des raisonnements théoriques basés sur un *modèle* ou bien sur des *preuves* .

Avec la première approche on parcourt de manière automatique l'espace d'états du graphe de transitions qu'on associe au système pour vérifier si une propriété spécifiée est satisfaite. Cette approche est appelée *vérification de modèle* (de l'anglais *model-checking*) [34] et a été proposée pour la première fois par Clarke et Emerson [33]. Puisqu'on doit énumérer tous les états possibles, et que l'espace considéré peut être très grand, il est important de rendre cette énumération la plus efficace possible. Pour atténuer ce problème de l'explosion de l'espace d'états on a proposé la *vérification symbolique de modèle* (de l'anglais *symbolic model-checking*) [29, 59]. Avec cette approche on représente les états du modèle par des formules booléennes. Ceci engendre un encodage compact des états qui permet de parcourir des espaces d'états plus vastes. La procédure qui est utilisée pour évaluer des grandes expressions booléennes et qui permet la détection de la redondance dans l'espace d'états encodés par des

expressions booléennes est appelée *BDD* (de l'anglais *Binary Decision Diagrams*). L'application de techniques de type vérification de modèle aux systèmes temporisés est encore une question relativement ouverte: l'introduction du temps produit des espaces d'états énormes. Plusieurs techniques ont été proposées pour aborder ce problème [52, 84, 10, 18, 82, 20, 40].

Les approches d'analyse basées sur des preuves sont utilisées par les méthodes basées sur des logiques temporelles ou des algèbres de processus. Dans le premier cas, l'usager développe une théorie sur le système, basée sur une logique, telle que la logique de premier ordre. Une propriété du système est alors exprimée par une formule de la logique choisie. On utilise des règles de déduction logique pour prouver formellement qu'une spécification implique des propriétés d'intérêt ou bien pour prouver que deux spécifications sont équivalentes. Dans le cas des algèbres de processus, on utilise l'ensemble de lois algébriques définies pour faire des manipulations syntaxiques de termes désignant des processus de l'algèbre et la relation d'équivalence, typiquement une relation de *bisimulation* [62], pour déterminer si deux processus ont le même comportement. Pour l'analyse de propriétés d'évolution de processus, tels que l'absence d'un blocage (en anglais *deadlock*), on utilise le modèle du système de transitions induit par la sémantique opérationnelle et on applique les techniques de vérification de modèle.

Le grand avantage des techniques de vérification de modèle est qu'elles sont automatiques, c'est à dire, elles peuvent être exécutées automatiquement par un outil de vérification et ne demandent donc pas de connaissances supplémentaires de la part du développeur de système. Les techniques basées sur des preuves, par contre, ne sont que partiellement automatiques, dans le meilleur des cas. L'application de ces techniques demande des connaissances mathématiques et spécifiques profondes de l'utilisateur. Par contre, elles permettent une compréhension approfondie du fonctionnement du système lors de l'élaboration de la preuve [77].

Nous avons fait un survol succinct des différentes méthodes formelles utilisés dans

la spécification et la vérification de systèmes temps-réel. Dans le chapitre suivant nous présenterons la problématique de la spécification et de la vérification d'interfaces matérielles.

Chapitre 3

Interfaces matérielles

3.1 Spécification d'interfaces matérielles

Un système informatique temps réel est généralement modélisé par un ensemble de composants, aussi appelés *processus*, qui interagissent via une *interface matérielle* commune et dont le protocole de fonctionnement doit satisfaire à des contraintes temporelles strictes. La conception d'un tel système se fait d'une manière hiérarchique: à partir de modules simples on bâtit des systèmes complexes. Le fonctionnement du système est alors déterminé par l'interaction des composants, qui sont définis par leur protocoles de communication respectifs. Les différents modules avec lesquels un composant interagit forment son *environnement*. Un composant est, donc, caractérisé par le protocole de communication avec son environnement.

Cette modularisation permet de clarifier et de simplifier la conception et la vérification de systèmes informatiques. En plus, elle permet la réutilisation de spécifications de composants, dans des contextes différents: les modules peuvent être composés de différentes manières pour générer des systèmes complexes variés. Ceci implique la nécessité que les composants soient spécifiés et vérifiés indépendamment les uns des autres. Par ailleurs, un composant n'est pas conçu pour fonctionner dans un environnement quelconque: son interface spécifie un ensemble d'hypothèses sur le

comportement de son environnement, qui doivent être satisfaites pour que la communication soit possible. Une interface définit ces suppositions sur les propriétés de l'environnement, ainsi que les réactions du composant face aux différents stimuli extérieurs reçus.

Traditionnellement, les techniques de spécification et de vérification d'interfaces matérielles s'appliquent surtout dans la conception de systèmes numériques asynchrones: bus, CPU, mémoires, etc. Ces systèmes sont couramment représentés sous forme de *chronogramme* (*timing diagram, action diagram*, ou *timing charts*).

Les chronogrammes sont un outil de spécification graphique qui permet de décrire naturellement des interfaces matérielles. Le manque de standard pour cette technique de spécification essentiellement informelle a donné lieu à des nombreux travaux qui proposent différents formalismes pour les chronogrammes [76, 25, 28, 26, 58, 55, 53, 31].

Nos recherches s'appuient sur la méthodologie de spécification des *chronogrammes hiérarchiques* (*Hierarchical Action Diagrams*) proposée par Khordoc et Cerny [55, 53, 31]: l'algèbre temporisée ACTC définit la sémantique et des méthodes de vérification formelles pour ces spécifications. La méthode consiste à décrire le comportement entrée–sortie d'un système à l'aide de chronogrammes de base, appelées *feuilles*. Les feuilles sont ensuite composées à l'aide d'un ensemble d'opérateurs hiérarchiques, tels que la composition séquentielle, la composition parallèle m-aire avec ou sans communication, le choix non-déterministe m-aire, le choix retardé [21, 16], et l'exception [21].

Un *chronogramme feuille* est essentiellement un diagramme qui décrit l'évolution d'*actions* dans le temps. Les actions correspondent à des changements de valeurs sur les divers *ports* d'une interface. Les ports modélisent des signaux physiques qui établissent la communication avec l'environnement. On distingue deux types d'actions: les actions de direction *sortie* (c'est à dire produits par le composant) et les actions de direction *entrée*, (exécutées par l'environnement).

Les *temps d'occurrence* peuvent être contraints par des *contraintes temporelles descriptives* ou *réactives*. Les contraintes descriptives sont aussi appelées *suppositions* (en anglais *assume constraints*) puisqu'elles décrivent les suppositions que le système fait sur le comportement de l'environnement. Les contraintes réactives sont appelés des *engagements* (en l'anglais *commit constraints*): elles spécifient des propriétés temporelles que le système s'engage à respecter.

Soient a et b deux actions quelconques avec leurs temps d'occurrence respectifs t_a et t_b. Une *contrainte temporelle* est de la forme $m \leq t_b - t_a \leq M$, où m et M sont appelées *borne inférieure* et *borne supérieure*, l'action a est la *source* et l'action b est le *puits* de la contrainte. Les bornes sont telles que $m \leq M$ et $M \in \mathbb{R}^+$. En plus, selon la valeur de m, on distingue des contraintes de *précédence*, où $m > 0$, et des contraintes de *concurrence*, avec une borne inférieure $m \leq 0$.

Lorsque plusieurs contraintes temporelles $m_i \leq t_b - t_{a_i} \leq M_i$ ont la même action puits b, où i est dans un ensemble d'indices I, on doit spécifier une forme de composition. On distingue trois types de contraintes temporelles composées:

- *Conj*: contrainte conjonctive

$$\text{Conj}(m_i \leq t_b - t_{a_i} \leq M_i) \stackrel{\text{def}}{=} \max_{i \in I}(m_i + t_{a_i}) \leq t_b \leq \min_{i \in I}(M_i + t_{a_i})$$

- *Latest*: contrainte–Max

$$\text{Latest}(m_i \leq t_b - t_{a_i} \leq M_i) \stackrel{\text{def}}{=} \max_{i \in I}(m_i + t_{a_i}) \leq t_b \leq \max_{i \in I}(M_i + t_{a_i})$$

- *Earliest*: contrainte–Min

$$\text{Earliest}(m_i \leq t_b - t_{a_i} \leq M_i) \stackrel{\text{def}}{=} \min_{i \in I}(m_i + t_{a_i}) \leq t_b \leq \min_{i \in I}(M_i + t_{a_i})$$

- *Span*: contrainte disjonctive

$$\text{Span}(m_i \leq t_b - t_{a_i} \leq M_i) \stackrel{\text{def}}{=} \min_{i \in I}(m_i + t_{a_i}) \leq t_b \leq \max_{i \in I}(M_i + t_{a_i})$$

Les contraintes *descriptives* définissent conjointement l'espace temporel correspondant au comportement temporel de l'environnement, tel que supposé par le composant. Elles peuvent être de type précédence ou concurrence. Leur composition est toujours conjonctive. En fait, les notions de source, cible et composition pour cette catégorie de contraintes sont pratiquement non-significatives: elles forment un système d'inégalités linéaires, dont l'espace de solutions correspond aux comportements acceptables de l'environnement.

Les contraintes *réactives* (en anglais *commit*) spécifient les propriétés qu'un composant s'engage à respecter lorsque son environnement satisfait aux contraintes descriptives. Elles sont toujours de type précédence et expriment une relation de causalité: l'exécution de l'action source a à un moment $t_a = x$ cause une réaction précise de la part du composant: il va produire l'action cible b quelque part à l'intérieur de l'intervalle $[x + m, x + M]$. Implicitement, l'action puits d'une contrainte est une action de sortie. Dans différents travaux on permet différents types de composition des contraintes réactives: conjonctive seulement dans [32], Max ou Min dans [21, 43], et Max, Min ou Span dans [22].

Le paradigme des propriétés descriptives/réactives (assume/commit) est couramment utilisé dans les méthodes formelles de conception et de vérification compositionnelle de systèmes distribués. Ce paradigme permet de mieux distinguer le comportement interne d'un agent de celui de son environnement, d'étudier des relations cause–effet entre les événements internes et externes ainsi que de vérifier des différentes propriétés de systèmes composés telles que la *réalisabilité*, la *causalité* [32], la *compatibilité* [28], et autres. Pour un bon "plaidoyer" en faveur de l'utilisation du paradigme assume/commit nous citons [1]. Ce paradigme est utilisé avec des nombreux formalismes de spécification. Un survol de ces travaux est fait dans [78]. À notre connaissance, ACTC est la seule algèbre temporelle qui supporte ce paradigme. Nous présenterons ces propriétés plus en détail dans la section suivante.

Les *chronogrammes hiérarchiques* sont composés à partir des chronogrammes feuilles

en utilisant des opérateurs de *composition parallèle communicante*, la *composition séquentielle*, le *choix non-déterministe*, le *choix retardé*, la *boucle*, et l'opérateur de *traitement d'exception.*

3.2 Vérification d'interfaces matérielles

Une spécification d'un système quelconque est inutilisable si elle ne peut pas être *réalisée* par une implantation concrète. Il y a des raisons évidentes pour lesquelles une spécification pourrait être non-réalisable: elle pourrait être logiquement inconsistante ou bien avoir d'autres problèmes fonctionnels flagrants. Un danger plus subtil est de spécifier des propriétés d'une partie de l'univers, appelé *environnement*, sur lequel on n'a aucun contrôle lors de l'implantation du système. Cette source de non-réalisabilité affecte surtout les spécifications de systèmes temps réel, réactifs, et concurrents, tels que les interfaces matérielles. Dans ce qui suit nous nous concentrons sur la question de la réalisabilité *temporelle* d'une spécification et ignorons les autres aspects fonctionnels d'une implantation physique. Pour un survol plus détaillé de la vérification d'interfaces matérielles nous référons à [43].

Une condition minimale pour qu'une spécification soit réalisable est la *consistance* [28]: une spécification définie à partir d'un ensemble d'actions dont le temps d'occurrence est déterminé par des contraintes temporelles est dite *temporellement consistante* s'il existe au moins une affectation des temps d'occurrence des actions qui satisfait à toutes les contraintes temporelles de la spécification. Des algorithmes efficaces pour le calcul des séparations temporelles maximales (et minimales) entre les temps d'occurrences d'actions sont présentés dans [60, 18, 12].

La consistance n'est pas suffisante pour assurer la réalisabilité d'une spécification d'un système *non-fermé*, c'est à dire qui contient la description de propriétés de l'environnement, puisqu'elle ne fait pas de distinction entre les contraintes descriptives et les contraintes réactives. Dans [32] on donne l'exemple d'une spécification qui

est consistante mais qui n'est pas réalisable parce qu'elle n'est pas *causale*. Dans [32] la *causalité* est définie comme la propriété d'une spécification dont le temps d'occurrence de toute action de sortie peut être décidé sans tenir compte des actions d'entrée pouvant être exécutées dans le futur par l'environnement. Des méthodes de vérification de la causalité pour certaines classes de chronogrammes sont données dans [32] et dans [43].

Le paradigme des contraintes d'engagement et de supposition est présent aussi dans la définition de la réalisabilité donnée dans [2]: une spécification est alors définie comme *réalisable* si et seulement si elle ne contraint pas le comportement de l'environnement.

Lorsqu'un système est spécifié comme une composition de composants communicants, il ne se pose pas seulement la question de la réalisabilité de chaque composant mais aussi celle de la *compatibilité* réciproque des composants. Dans le cas d'interfaces matérielles cette question s'exprime comme suit: étant donné les spécifications des interfaces d'au moins deux composants communicants, est-ce que toutes implantations des composants, selon leurs spécifications respectives, interagissent correctement?

La résponse donnée dans [28] à cette question est la suivante: la spécification d'un composant est *compatible* avec son environnement si l'espace de solutions de ses contraintes descriptives contient celui déterminé par l'ensemble des contraintes réactives de son environnement et vice-versa. Cerny et Khordoc démontrent dans [32] que cette condition est trop sévère (elle peut donner des fausses réponses négatives). Ils définissent une variation de cette condition que nous notons par *compatibilité*$^{\mathrm{CK}}$, et qui consiste à vérifier si le système composé de contraintes réactives du composant et de son environnement satisfait les systèmes de contraintes descriptives du composant ainsi que le système de contraintes descriptives de l'environnement. Ils montrent ensuite que la compatibilité$^{\mathrm{CK}}$ n'est pas suffisante pour assurer la compatibilité puisqu'elle peut donner des fausses réponses positives. Par contre, ils prouvent que pour certaines classes de chronogrammes, si les specifications des composants sont

causales et qu'elles sont compatibles^{CK}, alors elles sont compatibles.

Dans [21, 31] on propose une autre propriété d'une spécification d'interface donnée comme un terme ACTC: la *réactivité*, ainsi qu'une procédure de type *model-checking* pour la vérifier. Selon la sémantique opérationnelle définie pour ACTC toute inconsistance ou non-causalité d'un terme est signalée par l'exécution d'une action spéciale, appelée *action de blocage*. Un terme est alors défini comme *réactif* si aucune de ses traces ne contient l'action de blocage.

Nous avons présenté des techniques de spécification et de vérification d'interfaces matérielles en général, et des méthodes basées sur des chronogrammes hiérarchiques, en particulier. Dans le chapitre suivant nous donnons la définition formelle de la notion d'algèbre de processus et présentons les concepts théoriques nécessaires pour la suite de notre travail.

Chapitre 4

Algèbres de processus

Dans le dictionnaire "Le Petit Robert" un *processus* est défini comme un "ensemble de phénomènes conçu comme actif et organisé dans le temps". Toute spécification ou modèle formel d'un processus doit exprimer les trois concepts centraux de cette définition: l'ensemble de phénomènes, le dynamisme, et l'organisation. Dans ce chapitre nous présentons les notions et notations nécessaires pour définir une représentation formelle de processus par des algèbres de processus.

Premièrement, nous rappelons le concept de *système de transitions étiquetées* ou STE, présenté dans la section 1 et qui est le modèle de base des processus. Les STE décrivent un système dynamique qui évolue à l'intérieur d'un ensemble d'états selon une relation de transitions donnée. Les transitions sont *étiquetées* par des *actions*, c'est à dire qu'une transition d'un état à un autre est causée par l'occurrence d'un événement spécifique appelé *action*. Nous rappelons ensuite la définition de trois relations d'équivalence de STE, notamment l'*isomorphisme*, la *bisimulation forte* [62], et l'*équivalence de traces*.

Nous définissons par la suite le concept de *spécification équationnelle*, notée $\mathcal{E}$, qui permet de spécifier des processus par des expressions d'un langage de spécification avec une syntaxe précise et qui sont appelées *termes*. Une spécification équationnelle associe au langage de spécification un ensemble d'égalités de termes, appelées *ax-*

iomes, et un ensemble de règles d'inférences syntaxiques qui permettent de faire des déductions formelles d'égalités de termes quelconques. Ces déductions appelées *preuves* sont faites d'une manière purement syntaxique.

Ensuite nous présentons une méthode simple, appelée *sémantique opérationnelle structurée*, [74] notation SOS, d'associer à chaque terme d'une spécification équationnelle un STE. Les états du STE sont les termes du langage de spécification équationnelle. La relation de transition est définie par un ensemble de règles d'inférence sur la syntaxe du langage, en utilisant l'induction structurelle.

Les *algèbres de processus* donnent une *interprétation* des spécifications équationnelles et opérationnelles comme processus. Plus précisément, étant donné un langage de spécification quelconque avec une sémantique opérationnelle structurée on peut interpréter les termes du langage comme processus.

L'approche suivie en pratique lorsqu'on définit une algèbre de processus, consiste à définir d'abord un langage de spécification comme langage de termes d'une signature spécifique $\mathcal{F}$. Selon différents critères sémantiques on établit ensuite une relation d'équivalence sémantique de termes du langage. Cette relation d'équivalence partitionne l'ensemble de termes du langage dans des *classes d'équivalence sémantique* disjointes. Deux termes d'une même classe d'équivalence peuvent, donc, être syntaxiquement différents mais sont sémantiquement indistinguables. Deux termes équivalents spécifient le même processus et on modélise un processus par une classe d'équivalence sémantique. On essaye ensuite de définir une spécification équationnelle $\mathcal{E}$ telle que la relation d'équivalence syntaxique déterminée par $\mathcal{E}$ soit identique à la relation d'équivalence sémantique de termes.

Finalement, nous introduisons deux généralisations de STE, les STE *temporisés* et les *STE par parties*, notés $\wp$–STE.

4.1 Systèmes de transitions

Un *système de transitions*, noté ST et aussi appelé *machine à transitions*, est défini comme une paire $M = (\mathbf{S}, \rightarrow)$, où

- $\mathbf{S} = \{s, s', t, \ldots\}$ est un ensemble d'*états*;
- $\rightarrow \subseteq \mathbf{S} \times \mathbf{S}$ est une *relation de transition*; si $(s, s') \in \rightarrow$ on dit que M transite de l'état s vers l'état s' et on note $s \rightarrow s'$.

Le ST évolue dans le temps à l'intérieur de son espace d'états, en suivant les règles de transition définies par la relation $\rightarrow$.

Si on veut modéliser le fait que les transitions d'un état à un autre sont causées par des phénomènes spécifiques, appelés *actions*, alors on définit un *ensemble d'actions* A, appelé *alphabet*, et on étiquette les transitions par des actions. Rappelons ici les définitions de STE présentées dans la section 1.

La relation de transitions est définie comme suit:

- $\rightarrow \subseteq \mathbf{S} \times A \times \mathbf{S}$ est la relation de transition *étiquetée*; une transition $(s_1, a, s_2) \in \rightarrow$ est notée par $s_1 \xrightarrow{a} s_2$. Si $s_1 \xrightarrow{a} s_2$ on dit que M exécute l'action a et transite de l'état s_1 vers l'état s_2.

La machine ainsi définie est appelée un *système de transitions étiquetées*, notation STE.

De plus, si le système commence son évolution à partir d'un certain état $i \in \mathbf{S}$, appelé *état initial*, alors le STE est dit *initialisé* et est noté par STEI. Un STEI $M^i = (\mathbf{S}, A, \rightarrow, i)$ peut être défini comme $M^i = (M, i)$, où $M = (\mathbf{S}, A, \rightarrow)$ est appelé le STE (ou la machine) *sous-jacent* à M^i. Ainsi, étant donné un STE M quelconque on peut associer à chaque état $s \in \mathbf{S}$ un STEI $M^s = (M, s)$. Lorsque la définition de la machine sous-jacente est sous-entendue, on peut alors identifier l'état s à M^s. Donc, par abus de langage, lorsque $s \xrightarrow{a} s'$ est une transition d'un STE M on dit que

s exécute a et transite vers s', pour dire que le STEI M^s exécute a et transite de s vers s'.

Alternativement, on peut définir les transitions par une *fonction de transition* $\rho : \mathbf{S} \times A \longrightarrow \wp(\mathbf{S})$, telle que

$$\rho(s, a) = \{s' \mid s \xrightarrow{a} s'\}.$$

Rappelons que A^* désigne l'ensemble de séquences finies, et A^ω les séquences infinies sur A et que les séquences de $A^\infty = A^* \cup A^\omega$ sont appelées *mots* sur l'alphabet A .

Une *trace* d'un STE M à partir d'un état s_0 est un mot $\mu = a_0 a_1 ...$ de A^∞ tel qu'il existe une séquence d'états $r = s_0 s_1 ... \in \mathbf{S}^\infty$ telle que $s_i \xrightarrow{a_i} s_{i+1}$ pour tout i. La *longueur* d'une trace finie μ est le nombre d'actions de la trace et est notée par $|\mu|$ et

$$|\mu| = |a_0 \dots a_n| = n + 1.$$

μ est aussi appelé une *trace de* s_0. L'ensemble de traces de s_0 est noté par $\mathbb{T}(s_0)$. On appelle r l'exécution de la trace μ à partir de l'état s_0 (en anglais *run*). L'ensemble des exécutions à partir d'un état s est dénoté par $\mathbb{E}(s)$. Formellement,

$$\mathbb{T}(s) \stackrel{\text{def}}{=} \{a_0 a_1 ... \in A^\infty \mid \exists\, s_0 s_1 ... \in \mathbf{S}^\infty : (\forall i \geq 0)(s_i \xrightarrow{a_{i+1}} s_{i+1}) \wedge (s \xrightarrow{a_0} s_0)\}$$

$$\mathbb{E}(s) \stackrel{\text{def}}{=} \{s_0 s_1 ... \in \mathbf{S}^\infty \mid \exists\, a_0 a_1 ... \in A^\infty : (\forall i \geq 0)(s_i \xrightarrow{a_{i+1}} s_{i+1}) \wedge s \xrightarrow{a_0} s_0\}$$

La relation de transition $\rightarrow$ peut être étendue par transitivité pour permettre des transitions étiquetées par des traces finies $\mu = a_0 a_1 ... a_n \in A^*$. On dit que l'exécution de μ d'un état s_0 mène vers un état s_{n+1} et on note

$$s_0 \xrightarrow{\mu} s_{n+1}$$

si et seulement s'il existe une exécution $r = s_0 s_1 ... s_{n+1}$, telle que $s_i \xrightarrow{a_i} s_{i+1}$ pour tout i, $0 \leq i \leq n$.

Nous introduisons les notations suivantes:

- $s_1 \overset{a}{\not\rightarrow} s_2$ si et seulement si $(s_1, a, s_2) \notin \rightarrow$;
- $s_1 \overset{a}{\not\rightarrow}$ si et seulement si pour tout $s_2 \in A$ $(s_1, a, s_2) \notin \rightarrow$;
- $s_1 \not\rightarrow$ si et seulement si $s_1 \overset{a}{\not\rightarrow}$ pour toute action $a \in A$;
- $s_1 \overset{*}{\rightarrow} s_2$ si et seulement s'il existe une trace $\mu \in \mathbb{T}(s_1)$ telle que $s_1 \overset{\mu}{\rightarrow} s_2$.

Les *successeurs* immédiats d'un état s sont tous les états s' vers lesquels s peut transiter. L'ensemble de successeurs immédiats de s est noté par succ(s) et est défini formellement par:

$$\text{succ}(s) = \{s' \in \mathbf{S} \mid s \rightarrow s'\}.$$

On peut surcharger la notation succ comme suit,

$$\text{succ}(s, a) = \{s' \in \mathbf{S} \mid s \overset{a}{\rightarrow} s'\}.$$

Avec cette notation on arrive à:

$$\text{succ}(s) = \bigcup_{a \in A} \text{succ}(s, a).$$

L'ensemble des *descendants* d'un état $s \in \mathbf{S}$, noté ds(s) est défini comme:

$$\text{ds}(s) = \{s' \in \mathbf{S} \mid s \overset{*}{\rightarrow} s'\}.$$

Les actions *exécutables* d'un état s sont toutes les actions que M peut exécuter à partir de s. L'ensemble d'actions exécutables de s est noté par exec(s) et est défini comme:

$$\text{exec}(s) = \{a \in A \mid (\exists s' \in \mathbf{S})(s \overset{a}{\rightarrow} s')\}.$$

Un STEI $M^i = (A, \mathbf{S}, \rightarrow, i)$ est *connexe* si tous ses états sont accessibles à partir de l'état initial i via la relation de transition $\rightarrow$, c'est à dire

$$(\forall s \in \mathbf{S})(i \overset{*}{\rightarrow} s).$$

Un STEI connexe est appelé un *graphe de processus*. La partie accessible d'un STEI quelconque M^i est appelée *graphe de processus* de M^i, notée graphe(M^i), et elle est définie comme le STEI $G = (\mathbf{S}', A, \rightarrow')$ tel que

$$\begin{aligned} \mathbf{S}' &= \{s \in \mathbf{S} \mid i \xrightarrow{*} s\} \\ \rightarrow' &= \rightarrow \cap (\mathbf{S}' \times A \times \mathbf{S}'). \end{aligned}$$

Évidemment, si tous les états d'un STEI sont atteignables alors le STEI et son graphe de processus correspondant sont identiques.

Un STE est *déterministe* lorsque pour tout état s et toute action a l'image $\mathcal{S} = \rho(s, a)$ de la fonction de transition ρ contient tout au plus un élément. Dans le cas contraire le STE est appelé *non-déterministe*. Dans un STE non-déterministe il existe au moins une action a et un état s tels que l'ensemble $\mathcal{S} = \rho(s, a)$ contient au moins deux états différents, $\mathcal{S} = \{s_1, s_2, \ldots\}$. Dans ce cas on dit que l'exécution de a à partir de l'état s mène vers les états s_1, s_2, etc. Un STEI est *déterministe* si et seulement si le STE sous-jacent est déterministe.

4.2 Relations d'équivalence de STEI

Pour modéliser un processus on doit définir des critères sémantiques d'identification des processus. La relation d'équivalence induite par un ou plusieurs critères sémantiques est appelée *équivalence sémantique*. Une classification des équivalences sémantiques selon le critère "faire plus de distinction entre les processus" est donnée dans [80]. Nous allons présenter trois relations d'équivalence sémantiques de STEI, notamment la bisimulation, l'équivalence de traces, et l'isomorphisme. Une relation d'équivalence sémantique induit une partition de l'ensemble de STEI en *classes d'équivalence* telles que chaque classe correspond à un processus unique.

Considérons l'exemple suivant, inspiré de [17]. Supposons que nous sommes dans un édifice à plusieurs étages et voulons monter du rez-de-chaussée au 5ème étage en prenant un ascenseur. Ce processus peut être modélisé par les actions suivantes:

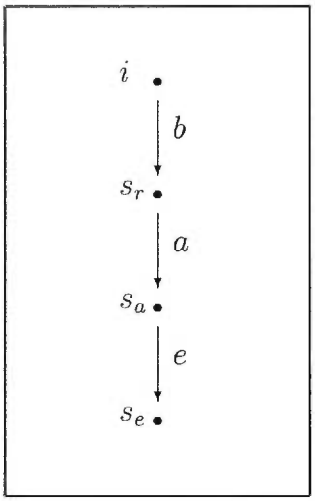

Figure 4.1: Le graphe de processus M_1

- b = appuyer sur le bouton pour appeler l'ascenseur;
- a = monter dans l'ascenseur;
- e = sortir de l'ascenseur.

et les états:

- i = état initial = être au rez-de-chaussée,
- s_r = être au rez-de-chaussée et attendre l'ascenseur,
- s_a = être dans l'ascenseur,
- s_e = être à l'étage.

Nous pouvons alors construire le graphe de processus $M_1 = (\mathbf{S}, A, \rightarrow, i)$, où $\mathbf{S} = \{i, s_r, s_a, s_e\}$, $A = \{b, a, e\}$ et la relation de transition $\rightarrow$ est telle qu'illustrée dans la figure 4.1.

Supposons maintenant que nous avons le choix entre deux ascenseurs différents. Pour modéliser cette nouvelle situation nous avons besoin des actions et des états suivants:

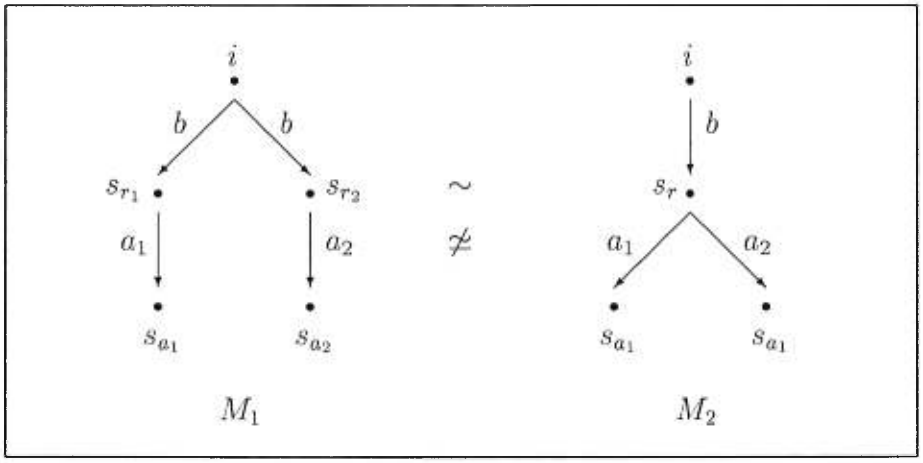

Figure 4.2: L'équivalence de traces $\sim$

- a_1 = monter dans l'ascenseur 1;
- a_2 = monter dans l'ascenseur 2;
- s_{a_1} = être dans l'ascenseur 1;
- s_{a_2} = être dans l'ascenseur 2.

Nous pouvons ignorer, pour l'instant, l'action e. Avant de pouvoir redéfinir les modèles correspondant à ce nouveau processus nous devons répondre à la question suivante: "Quand est-ce que nous choisissons l'ascenseur, avant ou après avoir appuyer sur le bouton?" Si nous choisissons avant, par exemple s'il y a un bouton différent pour chaque ascenseur, alors le STEI correspondant est M_1 de la figure 4.2. Si, au contraire, nous disposons d'un seul bouton et les deux ascenseurs arrivent simultanément au rez-de-chaussée, nous faisons le choix après avoir appuyé sur le bouton. Le STEI correspondant est présenté par M_2 dans la figure 4.2. Du point de vue d'un observateur extérieur qui ne peut observer que les séquences d'actions exécutées, M_1 et M_2 sont identiques. On dit que M_1 et M_2 sont *équivalents du point de vue des traces* et on écrit $M_1 \sim M_2$. Par contre, du point de vue de l'utilisateur de l'ascenseur, M_1 et M_2 ne sont pas du tout identiques. Il suffit de s'imaginer qu'un

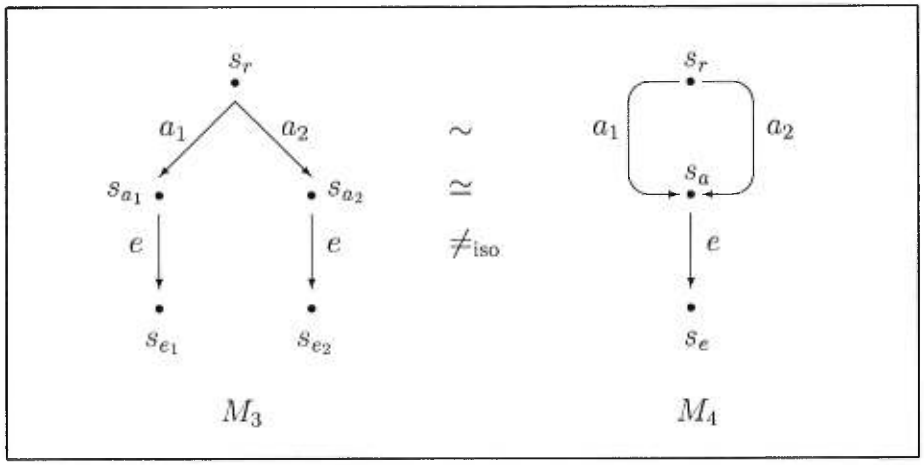

Figure 4.3: La bisimulation forte $\simeq$

des ascenseurs est défectueux et son usage est très risqué tandis que l'autre est tout à fait sécuritaire. Si l'on ne peut distinguer entre l'ascenseur sécure et celui défectueux que par un avertissement affiché à l'intérieur de l'ascenseur et si en plus on est obligé de prendre l'ascenseur que l'on a choisi, alors il est évident que tout usager sensé préférerait le scénario décrit par M_2 plutôt que celui de M_1.

Ignorons maintenant l'action b et considérons les STEI M_3 et M_4 de la figure 4.3. Les ensembles d'actions des deux STEI sont identiques et leurs espaces d'états sont respectivement,

- $\mathbf{S}_3 = \{s_r, s_{a_1}, s_{a_2}, s_{e_1}, s_{e_2}\}$,
- $\mathbf{S}_4 = \{s_r, s_a, s_e\}$,
- $A_3 = A_4 = \{a_1, a_2, e\}$,

où s_{e_i} = "être arrivé à l'étage par la porte de l'ascenseur i", avec $i = 1, 2$. Du point de vue de l'utilisateur de l'ascenseur, dont le seul but est de se rendre du rez-de-chaussée à l'étage, il n'y a aucune distinction entre les deux STEI. On dit que M_1 et M_2 sont *fortement bisimilaires* et on écrit $M_1 \simeq M_2$. M_1 et M_2 ne sont pas identiques, par contre, puisque leurs espaces d'états sont différents. Imaginons que nous avons donné

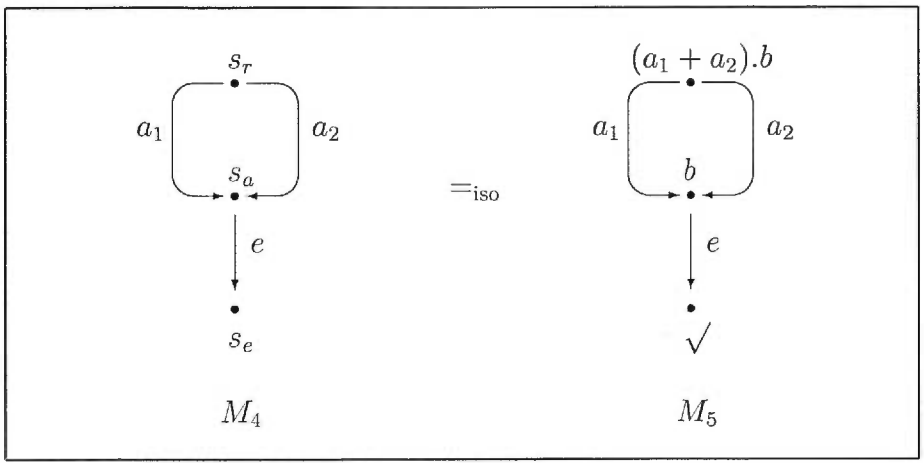

Figure 4.4: L'isomorphisme $=_{\text{iso}}$

un rendez-vous avec un ami "devant la porte de l'ascenseur". Il serait alors utile de pouvoir distinguer entre les deux portes, ce qui est possible avec M_3 mais pas avec M_4.

Finalement considérons les STEI M_4 et M_5 de la figure 4.4. M_4 et M_5 sont identiques, modulo un renommage des états:

- $s_r = (a_1 + a_2).b$,
- $s_a = b$,
- $s_e = \surd$.

On dit alors que M_4 et M_5 sont *isomorphes* et on écrit $M_4 =_{\text{iso}} M_5$. Notons que dans M_5 les états sont identifiés par des expressions syntaxiques formées à partir des symboles d'actions, d'un ensemble de symboles d'opérations (+ qui signifie le choix non-déterministe et . qui est la composition séquentielle) et d'un symbole spécial $\surd$ qui désigne l'état de terminaison d'un processus.

Donc, pour modéliser un processus il ne suffit pas de lui associer un STEI. On doit aussi définir une *relation d'équivalence sémantique* ou *opérationnelle*, notée $\mathcal{R}_{\mathcal{O}}$, de

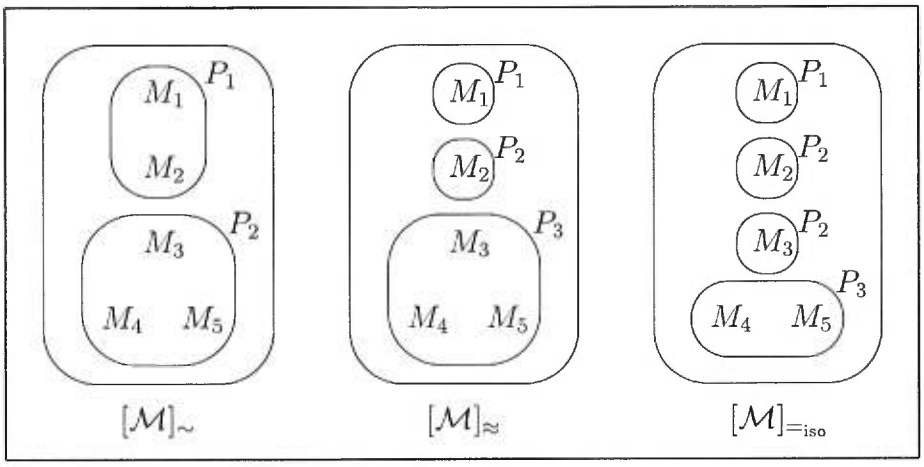

Figure 4.5: Classes d'équivalence sémantique

STEI telle que des STEI sont équivalents si et seulement si ils désignent le même processus. On peut alors construire des *classes d'équivalence modulo* $\mathcal{R}_{\mathcal{O}}$ sur l'ensemble des STEI et désigner un processus réel P_i unique pour chaque classe d'équivalence.

Soit $\mathcal{M} = \{M_1, \ldots, M_5\}$ l'ensemble de STEI de l'exemple précédent. Alors les classes d'équivalence sur $\mathcal{M}$ correspondant à la relation d'équivalence opérationnelle $\mathcal{R}_{\mathcal{O}}$ sont notées par $[\mathcal{M}]_{\mathcal{R}_{\mathcal{O}}}$ ou bien par $[\mathcal{M}]/\mathcal{R}_{\mathcal{O}}$. La correspondance entre les classes de $[\mathcal{M}]_{\sim}$, $[\mathcal{M}]_{\simeq}$, $[\mathcal{M}]_{=\text{iso}}$ et des processus P_i est illustrée à la figure 4.5.

Donnons maintenant les définitions formelles des relations d'équivalence de STEI mentionnées plus haut. Rappelons que nous identifions un STEI avec son état initial. Donc, par abus de langage, les équivalences de STEI peuvent aussi être considérées comme des relations d'états.

Définition 4.1 *Soient s_1 et s_2 deux états d'une machine à états. On dit que s_1 et s_2 sont équivalents du point de vue traces et on note $s_1 \sim s_2$ si et seulement si*

$$s_1 \sim s_2 \Leftrightarrow \mathbb{T}(s_1) = \mathbb{T}(s_2).$$

On dénote l'équivalence de traces par $\sim \subseteq \mathbf{S} \times \mathbf{S}$.

Une relation de *bisimulation forte* [62] est définie comme suit :

Définition 4.2 *Une relation $\mathcal{R} \subseteq \mathbf{S} \times \mathbf{S}$ est une bisimulation forte si et seulement si toute paire d'états (s_1, s_2) de $\mathcal{R}$ satisfait aux deux conditions suivantes:*

1. $(\forall a \in A, s_1' \in \mathbf{S})(s_1 \xrightarrow{a} s_1' \Rightarrow (\exists s_2' \in \mathbf{S})(s_2 \xrightarrow{a} s_2' \wedge (s_1', s_2') \in \mathcal{R}))$,
2. $(\forall a \in A, s_2' \in \mathbf{S})(s_2 \xrightarrow{a} s_2' \Rightarrow (\exists s_1' \in \mathbf{S})(s_1 \xrightarrow{a} s_1' \wedge (s_1', s_2') \in \mathcal{R}))$.

Autrement dit, $\mathcal{R}$ est *fermée à gauche et à droite aux transitions.*

Milner a prouvé [62] que la plus grande bisimulation forte, définie comme l'union de toutes les bisimulations fortes, existe et qu'elle est une relation d'équivalence sur l'ensemble d'états. On note cette relation par $\simeq$ et, par abus de langage, on l'appelle *bisimulation.* Formellement $\simeq$ est définie comme suit:

$$\simeq = \bigcup_{\mathcal{R} \subseteq \mathbf{S} \times \mathbf{S}} \mathcal{R}, \text{ telle que } \mathcal{R} \text{ est une bisimulation forte.}$$

Finalement, la relation d'*isomorphisme* est définie comme suit:

Définition 4.3 *Soient M_1 et M_2 deux STEI et $G_1 = \text{graphe}(M_1)$ et $G_2 = \text{graphe}(M_2)$ les graphes de processus correspondants. G_1 et G_2 sont isomorphes si et seulement si il existe une relation bijective $\mathcal{R}$ entre les états de G_1 et les états de G_2 telle que*

- $(i_1, i_2) \in \mathcal{R}$,
- *si $(s_1, s_2) \in \mathcal{R}$ et $(s_1', s_2') \in \mathcal{R}$, alors $s_1 \xrightarrow{a} s_1'$ si et seulement si $s_2 \xrightarrow{a} s_2'$, pour toute action a.*

Il est trivialement vrai que que l'équivalence de traces est plus faible que la bisimulation, qui, à son tour, est plus faible que l'isomorphisme. Cette propriété est formulée dans le théorème suivant:

Théorème 4.1 *Si s_1 et s_2 sont des états d'un ST quelconque alors*

$$s_1 =_{\text{iso}} s_2 \Rightarrow s_1 \simeq s_2 \Rightarrow s_1 \sim s_2$$

D'autre part, les implications inverses ne sont vraies que dans des conditions particulières. Le théorème suivant est un résultat connu [48].

Théorème 4.2 *Dans tout système à transitions déterministe l'égalité de traces et la bisimulation forte sont équivalentes. Formellement, si s_1 et s_2 sont deux états d'un STE déterministe, alors*

$$s_1 \simeq s_2 \Leftrightarrow s_1 \sim s_2.$$

4.3 Spécifications équationnelles

La modélisation d'un processus réel par un STEI modulo une relation d'équivalence opérationnelle, quoique très suggestive, a l'inconvénient d'être difficile à décrire et à analyser: on doit énumérer explicitement les ensembles d'états et les transitions de la relation $\rightarrow$. Or, les ensembles d'états sont typiquement très grands et rendent cette représentation extrêmement lourde.

L'alternative consiste à définir une *spécification équationnelle* $\mathcal{E}$ qui modélise des processus par des expressions syntaxiques, appelées *termes*, d'un langage de spécification avec une syntaxe précise. Avec une spécification équationnelle on peut déduire des égalités de termes à partir d'un ensemble d'équations $\mathbf{A}$, appelées *axiomes*, et d'un ensemble de *règles d'inférence* R_E d'une manière purement syntaxique.

Avant de formaliser ces concepts, considérons l'exemple suivant: soient M_3 et M_4 les STEI de la figure 4.3. M_3 et M_4 pourraient être identifiés aux expressions suivantes:

- $M_3 \stackrel{\text{def}}{=} a_1.e + a_2.e$,
- $M_4 \stackrel{\text{def}}{=} (a_1 + a_2).e$.

Si on définit l'axiome

$$x.z + y.z = (x + y).z,$$

où x, y, et z sont des variables désignant des termes, ainsi qu'une règle d'inférence permettant de substituer x par a_1, y par a_2, et z par e, alors on peut déduire que M_3 et M_4 sont égaux dans $\mathcal{E}$, notation $\mathcal{E} \models M_3 = M_4$.

Nous commençons par la définition des concepts de *signature* et de *terme.*

Une fonction *à n arguments* ou *n–aire* est une fonction $f : E^n \longrightarrow D$, où E et D sont des ensembles quelconques et n un nombre naturel. L'*arité* d'une fonction est définie comme le nombre n d'arguments de la fonction. Si $n = 0$, la fonction définit une *constante* de D. Si $n = 2$ alors f est dite *binaire.* Une fonction qui est définie sur un nombre arbitraire d'arguments est appelée *fonction d'arité arbitraire.*

Une *signature* $\mathcal{F}$ est un ensemble dont les éléments sont appelés *symboles de fonctions* ou *d'opérateurs,* muni d'une application $ar : \mathcal{F} \longrightarrow \mathbb{N}$, appelée fonction d'arité. La fonction ar associe à chaque symbole de fonction $f \in \mathcal{F}$ une arité $\mathrm{ar}(f)$. Si les éléments de $\mathcal{F}$ sont groupés en sous-classes selon leur arité,

$$\mathcal{F}_n = \{f \in \mathcal{F} \mid \mathrm{ar}(f) = n\},$$

alors la signature est uniquement déterminée par la famille de classes $\langle \mathcal{F}_n, n \in \mathbb{N} \rangle$. Par abus de langage nous désignons une signature $(\mathcal{F}, \mathrm{ar})$ uniquement par $\mathcal{F}$.

Soit $\mathcal{X}$ un ensemble d'éléments appelés *variables*, tel que $\mathcal{X} \cap \mathcal{F} = \emptyset$. L'ensemble $\mathcal{T}(\mathcal{F}, \mathcal{X})$, nommé *langage de termes* de $\mathcal{F}$ est défini comme suit:

- $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$;
- $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$;
- si $f \in \mathcal{F}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

Les variables de $\mathcal{X}$ sont notées par x, y, x_1, etc. et les termes de $\mathcal{T}(\mathcal{F}, \mathcal{X})$ par s, t, r, etc.

Remarquons ici que dans la littérature on rencontre parfois une définition légèrement différente de la notion de langage de termes: la signature $\mathcal{F}$ contient alors uniquement des symboles de fonctions f tels que $\mathrm{ar}(f) \geq 1$. L'ensemble $\mathcal{F}_0$ est remplacé par un

ensemble A dit *ensemble d'actions.* Le langage de termes est alors défini comme un triplet $\mathcal{T}(\mathcal{F}, \mathcal{X}, A)$ tel que

- $A \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X}, A)$;
- $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X}, A)$;
- si $f \in \mathcal{F}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X}, A)$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X}, A)$.

Soit Var : $\mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow \mathcal{X}$ l'application qui associe à chaque terme t l'ensemble de variables apparaissant dans t. On dit qu'un terme est *clos* s'il ne contient aucune variable. L'ensemble de termes clos d'une signature $\mathcal{F}$ est noté par $\mathcal{T}(\mathcal{F})$ et défini comme suit:

- $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F})$;
- si $f \in \mathcal{F}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F})$.

Une $\mathcal{T}(\mathcal{F}, \mathcal{X})$*–substitution* est une fonction $\varphi : \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ qui associe à chaque variable $x \in \mathcal{X}$ un terme $\varphi(x) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Une *extension homomorphique* de φ sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$, notée $\varphi^\sharp$, est une application $\varphi^\sharp : \mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ telle que

$$\begin{aligned}
\varphi^\sharp(t) &= t, \ \forall t \in \mathcal{T}(\mathcal{F}); \\
\varphi^\sharp(x) &= \varphi(x), \ \forall x \in \mathcal{X}; \\
\varphi^\sharp(f(t_1, \ldots, t_n)) &= f(\varphi^\sharp(t_1), \ldots, \varphi^\sharp(t_n)), \ \forall f \in \mathcal{F}, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X}), i = 1, \ldots, n.
\end{aligned}$$

Par abus de notation, on désigne $\varphi^\sharp$ par φ. Le terme $\varphi(t)$ est obtenu en remplaçant chaque occurrence dans t de toute variable $x \in \mathrm{Var}(t)$ par $\varphi(x)$. Remarquons que si la substitution φ est telle qu'elle associe à chaque variable $x \in \mathcal{X}$ un terme clos $\varphi(x) \in \mathcal{T}(\mathcal{F})$, alors l'extension homomorphique de φ aussi retourne uniquement des termes clos. Autrement dit, $\varphi^\sharp(t) \in \mathcal{T}(\mathcal{F})$, pour tout $t \in \mathcal{T}(\mathcal{X}, \mathcal{F})$.

Définissons maintenant la notion de *sous-terme.* Si $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ alors on dit que s est un *sous-terme* de t et on écrit $t[s]$ si et seulement si

- $s = x \in \mathcal{X}$ et $x \in \text{Var}(t)$ ou,
- $s = g(s_1, \ldots, s_m)$ et $t = f(t_1, \ldots, t_n)$ et soit
  - $f = g, m = n$, et $t_i = s_i (1 \leq i \leq n)$, ou bien
  - $\exists i : 1 \leq i \leq n$ et $t_i[s]$.

Si $t[x]$, où $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et $x \in \mathcal{X}$ et si on remplace chaque occurrence de x dans t par un terme $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ on obtient un terme $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, noté $t[s/x]$. On dit que la variable x a été *liée* au terme s, ou bien qu'on lui a *assigné* le terme s.

Une *spécification équationnelle* $\mathcal{E} = (\mathcal{F}, \mathbf{A})$ est définie par une signature $\mathcal{F}$ qui, combinée à un ensemble de variables $\mathcal{X}$, détermine un langage de termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$, et un ensemble $\mathcal{A}$ d'*équations de termes* ou *axiomes* de la forme $s = t$, où $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

On associe à $\mathcal{E}$ l'ensemble de *règles d'inférence* R_E de la logique équationnelle. R_E contient les règles suivantes:

1. les propriétés d'*équivalence* de $=$:
   pour tous $t, s, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   - *réflexivité*: $t = t$
   - *symétrie*: $t = s$ implique $s = t$
   - *transitivité*: $t = s$ et $s = r$ implique $t = r$
2. la règle de *substitution*:
   pour tous $t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et toute substitution $\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow T(\mathcal{F}, \mathcal{X})$,
   $$t_1 = t_2 \text{ implique } \varphi(t_1) = \varphi(t_2);$$
3. la règle de *contexte*:
   pour tout $n \geq 1$ et tous $t_1, \ldots, t_n, s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   $$(\forall i : 1 \leq i \leq n)(t_i = s_i), \text{ implique } f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n).$$

On dit qu'une égalité de termes $s = t$ peut être *dérivée* dans $\mathcal{E}$ et on écrit $\mathcal{E} \vdash s = t$ si et seulement si

- $s = t \in \mathbf{A}$ ou bien
- l'égalité $s = t$ peut être déduite à partir des axiomes de $\mathbf{A}$ en utilisant les règles d'inférence de R_E.

Un système de spécification équationnelle ϵ induit une relation d'*équivalence équationnelle* ou *syntaxique* de termes, notée $\mathcal{R}_\epsilon$, telle que

$$\mathcal{R}_\epsilon = \{(s,t) \in \mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X}) \mid \epsilon \vdash s = t\}.$$

4.4 Sémantique opérationnelle structurée

Une spécification équationnelle a l'avantage d'être compacte, facile à analyser et représenter. Mais ce n'est qu'une représentation *syntaxique*, c'est à dire formée par des séquences de symboles abstraites, sans aucune interprétation réelle. Pour pouvoir utiliser une telle spécification on doit d'abord lui donner une *sémantique réelle*, autrement dit définir une manière d'associer à chaque objet syntaxique un processus.

Il existe une méthode simple pour donner une *sémantique opérationnelle* à une spécification équationnelle, c'est à dire une application qui associe un STE à chaque terme du langage de spécification équationnelle. La méthode s'appelle *sémantique opérationnelle structurée* ou *SOS* et a été développée par Plotkin [74] pour donner une sémantique opérationnelle aux langages de programmation. L'idée principale est de définir un système de transitions dont les états sont des éléments du langage de spécification et les transitions entre les états sont définies par un ensemble de règles d'inférence sur la syntaxe du langage, en utilisant un *système de déduction de termes* ou SDT.

Un SDT est une structure $\mathcal{O} = (\mathcal{F}, \mathcal{D})$ avec une signature $\mathcal{F}$ et un ensemble de *règles de déduction* $\mathcal{D}$. On appelle les règles de déduction aussi des *règles de*

sémantique opérationnelle, et qui sont notées par notation SOp.

L'ensemble $\mathcal{D} = \mathcal{D}(\mathcal{O}_p, \mathcal{O}_r)$ est paramétrisé par $\mathcal{O}_p \subseteq 2^{\mathcal{T}(\mathcal{F},\mathcal{X})}$ et $\mathcal{O}_r \subseteq 2^{\mathcal{T}(\mathcal{F},\mathcal{X})\times\mathcal{T}(\mathcal{F},\mathcal{X})}$, qui représentent respectivement, un ensemble de symboles de *prédicats* et de *relations* de termes du langage $\mathcal{T}(\mathcal{F},\mathcal{X})$.

Soient $P \in \mathcal{O}_p$, $\mathcal{R} \in \mathcal{O}_r$, et $s,t,u \in \mathcal{T}(\mathcal{F},\mathcal{X})$. On appelle les expressions Ps et $t\mathcal{R}u$ des *formules*. Une *règle de déduction* $d \in \mathcal{D}$ a la forme suivante:

$$\frac{H}{C}$$

où H est un ensemble de formules appelées *hypothèses* de d et C est une formule, dite *conclusion* de d.

Souvent, s'il n'y pas de confusion possible, on dénote un axiome simplement par sa conclusion. Les notions de "substitution", "Var ", et "clos" s'étendent aux formules et règles de déduction comme attendu.

Soit $\mathcal{O} = (\mathcal{F},\mathcal{D})$ un SDT. Une *preuve* d'une formule ψ à partir de $\mathcal{O}$ est un arbre à branchement ascendant, dont les noeuds sont étiquetés par des formules tel que:

- la racine est étiquetée par ψ, et
- si χ est l'étiquette d'un noeud q et $\{\chi_i \mid i \in I\}$ est l'ensemble des étiquettes des noeuds $\{q_i \mid i \in I\}$ directement au dessus de q alors il existe une règle de déduction

  $$\frac{\{\psi_i \mid i \in I\}}{\psi}$$

  et une substitution $\varphi : \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F},\mathcal{X})$ telle que $\varphi(\psi) = \chi$ et $\varphi(\psi_i) = \chi_i$ pour $i \in I$.

Si une preuve de ψ existe dans $\mathcal{O}$, alors on dit que ψ est *prouvable* à partir de $\mathcal{O}$, notation $\mathcal{O} \vdash \psi$. Une preuve est *close* si elle contient uniquement des formules closes.

Un système de déduction de termes induit de façon naturelle un système de transitions.

Soit $\mathcal{O} = (\mathcal{F}, D)$ un SDT avec $D = D(\mathcal{O}_p, \mathcal{O}_r)$. Le *système de transition induit* par $\mathcal{O}$, noté $\mathrm{ST}(\mathcal{O}) = (\mathbf{S}, \rightarrow)$ est défini comme suit:

- l'ensemble d'états est $\mathbf{S} = \mathcal{T}(\mathcal{F})$,
- la relation de transition $\rightarrow$ est telle que

$$\rightarrow = \{(s,t) \in \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}) \mid (\exists \mathcal{R} \in \mathcal{O}_r)\ (\mathcal{O} \vdash s\mathcal{R}t)\}.$$

Pour définir un système de transitions étiquetées par un ensemble d'actions A à partir de $\mathcal{O}$, l'ensemble $\mathcal{O}_r$ doit contenir une relation $\mathcal{R}_a$ pour chacune des actions $a \in A$:

$$\mathcal{O}_r = \{\mathcal{R}_a \mid a \in A\}.$$

Notons ce STE par STE($\mathcal{O}$). La relation de transition étiquetée est alors définie comme:

$$\rightarrow = \{(s,a,t) \in \mathcal{T}(\mathcal{F}) \times A \times \mathcal{T}(\mathcal{F}) \mid \mathcal{O} \vdash s\mathcal{R}_a t\}.$$

Donc, pour donner une sémantique opérationnelle aux termes clos de $\mathcal{T}(\mathcal{F})$, il suffit de définir un SDT $\mathcal{O} = (\mathcal{F}, \mathcal{D})$, qui associe le système de transitions étiquetées STE($\mathcal{O}$) au langage de termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Ensuite, tel que présenté dans la section 4.1, on définit pour chaque terme clos $t \in \mathcal{T}(\mathcal{F})$ le STEI, noté graphe($\mathcal{O}, t$) ou simplement graphe(t), qui est la partie atteignable du système de transitions STE($\mathcal{O}$) initialisé par t.

Dans la section 4.2 nous avons présenté la notion de relation d'équivalence sémantique de STEI, notée $\mathcal{R}_\mathcal{O}$. Puisqu'à chaque terme clos $t \in \mathcal{T}(\mathcal{F})$ correspond un STEI graphe(t), nous pouvons étendre la relation sémantique à l'ensemble de termes clos $\mathcal{T}(\mathcal{F})$ comme suit:

$$(t,s) \in \mathcal{R}_\mathcal{O} \Leftrightarrow (\text{graphe}(\mathcal{O},t), \text{graphe}(\mathcal{O},s)) \in \mathcal{R}_\mathcal{O}.$$

On appele $\mathcal{R}_\mathcal{O}$ la *relation d'équivalence sémantique* (ou *opérationnelle*) de termes *induite* par la sémantique opérationnelle de $\mathcal{O}$.

Nous avons maintenant introduit tous les concepts nécessaires pour pouvoir définir formellement la notion d'*algèbre de processus* dans la section suivante.

4.5 Algèbres de processus – Définition

Une *algèbre* $\mathcal{A} = (E, F)$ est définie par un ensemble d'éléments E et un ensemble F de fonctions fermées sur E, c'est à dire tel que toute fonction $f \in F$ est définie comme $f : E^n \longrightarrow E$, où $n \in \mathbb{N}$ est l'arité de f. E est aussi appelé le *domaine* de $\mathcal{A}$.

Soit $\mathcal{F}$ une *signature*, c'est à dire un ensemble de *symboles* de fonctions munie d'une fonction d'arité ar : $\mathcal{F} \longrightarrow \mathbb{N}$ (ici $\mathbb{N}$ désigne l'ensemble de nombres naturels supérieurs ou égaux à zéro). Étant donné un ensemble quelconque d'éléments E, une *interprétation* dans E d'un symbole de fonction $f \in \mathcal{F}$ est une fonction $i(f)$: $E^{\mathrm{ar}(f)} \longrightarrow E$ de $E^{ar(f)}$ vers E. Une interprétation de $\mathcal{F}$ est une application i qui associe à chaque symbole $f \in \mathcal{F}$ une interprétation dans E.

Pour une signature $\mathcal{F}$ et un ensemble d'éléments E une *$\mathcal{F}$–algèbre* $\mathcal{A}$ est définie par une interprétation i de $\mathcal{F}$ dans E. Elle est dénotée $\mathcal{A} = (E, i(\mathcal{F}))$ ou comme $\mathcal{F}_{\mathcal{A}}$. L'interprétation du symbole de fonction f dans E est notée $f_{\mathcal{A}}$.

Soit $\mathcal{T}(\mathcal{F})$ le langage de termes clos tel que défini dans la section 4.3. Une interprétation i de $\mathcal{F}$ dans un ensemble quelconque E associe à tout terme $t \in \mathcal{T}(\mathcal{F})$ un élément unique de $i(t) \in E$, tel que:

- si $t = f \in \mathcal{F}_0$, c'est à dire si t est un symbole de fonction d'arité zéro, alors $i(t) = f_{\mathcal{A}} \in E$ est une *constante* du domaine E,
- si $t = f(t_1, \ldots, t_n)$, alors $i(t) = f_{\mathcal{A}}(i(t_1), \ldots, i(t_n))$, avec $n \geq 1$.

On dit que $\mathcal{A}$ donne une *sémantique dans E* aux termes de $\mathcal{T}(\mathcal{F})$.

Si $\mathcal{A} = (E, i(\mathcal{F}))$ est une $\mathcal{F}$–algèbre, alors une égalité $t_1 = t_2$ de termes clos $t_1, t_2 \in \mathcal{T}(\mathcal{F})$ est *vraie dans $\mathcal{A}$* et on note

$$\mathcal{A} \models t_1 = t_2$$

si et seulement si $i(t_1)$ et $i(t_2)$ désignent le même élément de E. Si les termes $t_{1,2}$ contiennent des variables, alors celles-ci doivent être quantifiées universellement, c'est

à dire

$$\mathcal{A} \models t_1 = t_2 \overset{\text{notation}}{\Leftrightarrow} (\forall x \in \text{Var}(t_1))(\forall y \in \text{Var}(t_2))(\mathcal{A} \models t_1 = t_2).$$

L'algèbre $\mathcal{A}$ définit, donc, une relation d'*équivalence sémantique* sur le langage de termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$, notée $\mathcal{R}_{\mathcal{A}}$:

$$\mathcal{R}_{\mathcal{A}} = \{(t, s) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \mathcal{A} \models t = s\}.$$

Soit $\approx \subseteq E \times E$ une relation d'équivalence quelconque sur le domaine E d'une algèbre $\mathcal{A}(E, i(\mathcal{F}))$. Rappelons que nous notons par $[a]_{\approx} \subseteq E$ la classe d'équivalence modulo $\approx$ de l'élément $a \in E$:

$$[a]_{\approx} = \{a' \in E \mid a' \approx a\}.$$

On dit que $\approx$ est une *congruence* par rapport à $\mathcal{F}$ si et seulement si pour toute fonction $f_{\mathcal{A}} \in i(\mathcal{F})$ d'arité $n \geq 0$

$$(\forall i : 0 \leq i \leq n)(a_i \approx a_i') \;\rightarrow\; f_{\mathcal{A}}(a_1, \ldots, a_n) \approx f_{\mathcal{A}}(a_1', \ldots, a_n'),$$

où $a_i, a_i' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ pour tout $i : 1 \leq i \leq n$.

Le résultat suivant est évident: si $\approx$ est une congruence pour toutes les fonctions de $i(\mathcal{F})$, alors $\mathcal{A}_{\approx} \overset{\text{def}}{=} (E_{\approx}, i_{\approx}(\mathcal{F}))$ est aussi une $\mathcal{F}$–algèbre de domaine $E_{\approx}$, où l'interprétation $i_{\approx}$ de $\mathcal{F}$ est définie comme suit:

$$i_{\approx}(f) \overset{\text{not.}}{=} f_{\mathcal{A}_{\approx}}([a_1]_{\approx}, \ldots, [a_n]_{\approx}) \overset{\text{def}}{=} [f_{\mathcal{A}}(a_1, \ldots, a_n)]_{\approx},$$

pour tout $f \in \mathcal{F}$. Alors, la relation d'équivalence sémantique $\mathcal{R}_{\mathcal{A}_{\approx}}$ définie par $\mathcal{A}_{\approx}$ sur le domaine $E_{\approx}$ est telle que

$$[a_1]_{\approx} \; \mathcal{R}_{\mathcal{A}_{\approx}} \; [a_2]_{\approx} \overset{\text{def.}}{\Leftrightarrow} \mathcal{A}_{\approx} \models [a_1]_{\approx} = [a_2]_{\approx} \Leftrightarrow a_1 \approx a_2.$$

La condition que $\approx$ soit une congruence est nécessaire pour la raison suivante: supposons que $a_i \approx b_i$, alors a_i et b_i appartiennent à la même classe d'équivalence:

$$[a_i]_{\approx} = [b_i]_{\approx}.$$

Donc, pour qu'une fonction quelconque $f_{\mathcal{A}_\approx}$ soit bien définie, il faut que

$$[f_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_n)]_\approx = [f_{\mathcal{A}}(a_1, \ldots, b_i, \ldots, a_n)]_\approx,$$

pour tous $a_1, \ldots, a_n \in E$. Donc, il faut que

$$f_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_n) \approx f_{\mathcal{A}}(a_1, \ldots, b_i, \ldots, a_n).$$

Or, ceci revient à dire que $\approx$ est une congruence pour les fonctions de $\mathcal{F}_{\mathcal{A}}$.

On dit que l'algèbre $\mathcal{A}$ est *axiomatisée* s'il existe une spécification équationnelle $\mathcal{E} = (\mathcal{F}, \mathbf{A})$ pour le langage de termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$ défini par la signature $\mathcal{F}$. Soit $\mathcal{R}_{\mathcal{E}}$ est la relation d'équivalence syntaxique induite par $\mathcal{E}$ sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Si $\mathcal{R}_{\mathcal{E}} \subseteq \mathcal{R}_{\mathcal{A}}$, donc si

$$(\forall t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X}))(\mathcal{E} \vdash t = s \text{ implique } \mathcal{A} \models t = s),$$

alors on dit que $\mathbf{A}$ est une *axiomatisation cohérente* de l'algèbre $\mathcal{A}$, ou bien que $\mathbf{A}$ est cohérente par rapport à $\mathcal{R}_{\mathcal{A}}$, et on écrit

$$\mathcal{A} \models \mathbf{A}.$$

Si $\mathcal{R}_{\mathcal{A}} \subseteq \mathcal{R}_{\mathcal{E}}$, donc si

$$(\forall t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X}))(\mathcal{A} \models t = s \text{ implique } \mathcal{E} \vdash t = s),$$

alors on dit que $\mathbf{A}$ est une *axiomatisation complète* de l'algèbre $\mathcal{A}$, ou bien que $\mathbf{A}$ est complète par rapport à $\mathcal{R}_{\mathcal{A}}$, et on écrit

$$\mathbf{A} \vdash \mathcal{A}.$$

Supposons maintenant que nous voulons formaliser un langage algébrique $\mathcal{T}(\mathcal{F})$ d'une signature $\mathcal{F}$ pour la modélisation de processus. Pour ce faire nous donnons une sémantique opérationnelle aux termes de $\mathcal{T}(\mathcal{F})$ en définissant un système de déduction de termes $\mathcal{O} = (\mathcal{F}, D)$ pour ce langage. Ainsi nous associons un graphe de processus graphe$(\mathcal{O}, t)$ à chaque terme clos t d'un langage algébrique $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Notons par $\mathcal{G}(\mathcal{O}, \mathcal{F})$ l'ensemble de graphes de processus associés aux termes de $\mathcal{T}(\mathcal{F})$. Selon des différents critères sémantiques on peut définir une relation d'équivalence sémantique $\approx$ sur $\mathcal{G}(\mathcal{O}, \mathcal{F})$. Les processus sont alors modélisés par des classes d'équivalences modulo $\approx$.

Puisque nous identifions souvent un terme $t \in \mathcal{T}(\mathcal{F})$ au STEI correspondant M^t, nous surchageons la notation $\approx$ pour désigner également la relation d'équivalence sémantique induite ainsi sur l'ensemble de termes clos de $\mathcal{T}(\mathcal{F})$:

$$t \approx t' \overset{\text{not.}}{\Leftrightarrow} \text{graphe}(\mathcal{O}, t) \approx \text{graphe}(\mathcal{O}, t').$$

Nous pouvons alors définir une $\mathcal{F}$–algèbre $\mathcal{A} = (\mathcal{G}(\mathcal{O}, \mathcal{F}), i(\mathcal{F}))$ telle que l'interprétation i associe à chaque terme clos $t \in \mathcal{T}(\mathcal{F})$ le graphe de processus graphe$(\mathcal{O}, t)$, selon la sémantique opérationnelle de $\mathcal{O}$. De plus, si $\approx$ est une congruence par rapport à $\mathcal{F}$, alors on peut définir l'algèbre

$$\mathcal{A}_{\approx}([\mathcal{G}(\mathcal{O}, \mathcal{F})]_{\approx}, i(\mathcal{F})).$$

Chaque terme clos $t \in \mathcal{T}(\mathcal{F})$ est interprété dans $\mathcal{A}_{\approx}$ comme un *modèle de processus* qui est la classe d'équivalence $[\text{graphe}(\mathcal{O}, t)]_{\approx}$ telle que, pour tout terme clos $t \in \mathcal{T}(\mathcal{F})$,

$$i(t) = [\text{graphe}(\mathcal{O}, t)]_{\approx}.$$

L'algèbre $\mathcal{A}_{\approx}$ ainsi définie est appelée une *algèbre de processus*, notée $\mathcal{AP}$.

À toutes fins pratiques, une algèbre de processus peut être caractérisée par les trois composantes suivantes:

1. un langage de termes, noté $\mathcal{T}(\mathcal{F})$, d'une signature $\mathcal{F}$;
2. l'ensemble de règles de déduction, noté SOp, d'un système de déduction de termes $\mathcal{O} = (\mathcal{F}, \text{SOp})$, aussi appelées *règles de sémantique opérationnelle*;
3. une relation d'équivalence opérationnelle de termes, notée $\approx$, définie en fonction de critères sémantiques spécifiques et induite par $\mathcal{O}$.

Rappelons qu'on associe aussi fréquemment à une algèbre de processus un ensemble d'actions

$$A = \{a, b, \ldots\}$$

qui peuvent être formellement représentées par des symboles de fonctions d'arité zéro.

On dit que l'algèbre de processus $\mathcal{A}_{\approx}$ est *axiomatisée* lorsqu'une spécification équationnelle $\mathcal{E} = (\mathcal{F}, \mathbf{A})$ est définie pour $\mathcal{T}(\mathcal{F})$. Pour vérifier si l'axiomatisation $\mathbf{A}$ de la spécification équationnelle $\mathcal{E}$ est cohérente et complète par rapport à $\mathcal{A}_{\approx}$ il suffit alors de vérifer que $\mathcal{R}_{\mathcal{A}_{\approx}} = \mathcal{R}_{\mathcal{E}}$. Ceci revient à vérifier que pour tous $t, t' \in \mathcal{T}(\mathcal{F})$

$$t \approx t' \Leftrightarrow \mathbf{A} \models t = t'.$$

Dans la section suivante nous présentons deux généralisations de la notion de système de transitions étiquetées. Puisque les STE sont le modèle de base des algèbres de processus, les généralisations présentées ici s'appliquent naturellement aussi aux algèbres de processus.

4.6 Généralisations

Nous commençons par la définition de systèmes à transitions *temporisés*. Ensuite nous introduisons le concept de système de transition *par parties*, noté $\wp$–STE.

4.6.1 Temporisations

Nous présentons ici les notations et les définitions associées aux machines et aux algèbres de processus temporisés. Les implications sur les modèles et la sémantique de ces formalismes ont été décrites dans le Chapitre 2.

On définit un ST *temporisé* comme une extension de la notion de système de transitions étiquetées. L'extension consiste à étiqueter chaque transition non seulement par une action, mais aussi par un instant temporel. Un STE temporisé, noté MT, est un quadruplet $\mathrm{MT} = (\mathbf{S}, A, \rightarrow, D)$, où

- A et $\mathbf{S}$ sont comme dans la définition d'un STE;
- D est un domaine temporel quelconque(par exemple, $\mathbb{R}^+$ ou $\mathbb{N}$); et
- $\rightarrow \subseteq \mathbf{S} \times A \times D \times \mathbf{S}$ est la relation de transition étiquetée par des actions temporisées; on écrit $s_1 \stackrel{a(v)}{\rightarrow} s_2$ si $(s_1, a, v, s_2) \in \rightarrow$ et on dit que la machine MT exécute l'action a au temps v et transite de l'état s_1 vers l'état s_2.

Toutes les notions basées sur des machines non-temporisées s'appliquent naturellement aux machines temporisées: il suffit de considérer $A \times D$ comme ensemble d'actions – elles sont alors appelées *actions temporisées* .

Dans la section 4.1 nous avons défini la fonction exec pour des machines non-temporisées. Dans le cas des machines temporisées, l'ensemble exec(s) contient les actions temporisées qui peuvent être exécutées à partir d'un état s:

$$\text{exec}(s) = \{a(v) \mid (\exists s' \in S(s \stackrel{a(v)}{\rightarrow} s')\}.$$

Lorsqu'aucune confusion ne sera crée, nous surchargerons cette notation comme suit

$$\text{exec}(s) = \{a \mid (\exists s' \in S)(\exists v \in D)(s \stackrel{a(v)}{\rightarrow} s')\}.$$

Pour définir des *automates*, des *processus*, et des *algèbres temporisées* il suffit de leur associer un STE temporisé. Ceci revient à ajouter le domaine temporel à leur définition et à étendre la relation de transition pour permettre des transitions étiquetées par des actions temporisées.

Notons qu'il existe une temporisation plus complexe des automates [6, 8, 7]: outre le domaine temporel D, on ajoute à la définition d'un automate non-temporisé un ensemble d'*horloges* H, des variables qui prennent leurs valeurs dans D et qui évoluent de manière synchronisée, une fonction μ qui associe à chaque transition de $\rightarrow$ un l'ensemble d'horloges qui doivent être ré-initialisés, et une fonction γ qui associe à chaque transition des contraintes temporelles sur l'ensemble d'horloges qui doivent être respectées pour que la transition puisse avoir lieu.

4.6.2 Les systèmes de transitions par parties

Soit $M = (\mathbf{S}, A, \rightarrow)$ un STE quelconque. Via la relation de transition $\rightarrow \subseteq \mathbf{S} \times A \times \mathbf{S}$ on peut analyser l'évolution de la machine, état par état, lors de l'exécution d'une trace d'actions. Si un STE est non-déterministe, différentes exécutions peuvent correspondre à une seule trace. Lorsque seules les traces d'une machine nous intéressent, sans distinction entre les différentes exécutions qui peuvent leur correspondre, il peut être utile d'étudier les transitions d'un *ensemble d'états* vers un autre ensemble d'états, appelées *transitions par parties* , notée $\longmapsto$. Les sous-ensembles d'états sont notés par $\mathcal{S}, \mathcal{S}', \mathcal{T}, \mathcal{T}'$, etc. Rappelons que l'ensemble de parties de $\mathbf{S}$ est noté par $\wp(\mathbf{S})$ ou bien par $2^{\mathbf{S}}$.

Intuitivement, lorsqu'on exécute une action a à partir d'un ensemble d'états $\mathcal{S}_1 \subseteq \mathbf{S}$ on transite vers un ensemble d'etats $\mathcal{S}_2 \subseteq \mathbf{S}$ qui est l'union de toutes les destinations possibles de transitions à partir d'états $s_1 \in \mathcal{S}_1$.

Soit $M = (\mathbf{S}, A, \rightarrow)$ un STE. Le STE par parties associé à M et noté $\wp(M) = (\wp(\mathbf{S}), A, \longmapsto)$, est paramétré par:

- A, qui est le même vocabulaire que celui de M,
- $\wp(\mathbf{S})$, qui est l'ensemble d'états de $\wp(M)$;
- $\longmapsto \subseteq \wp(\mathbf{S}) \times A \times \wp(\mathbf{S})$, qui est la relation de transition par parties; pour tout $\mathcal{S}_1 \subseteq \mathbf{S}$ et toute action $a \in \mathbf{A}$ on définit la transition par parties étiquetée par a de $\mathcal{S}_1$ comme suit:

$$\mathcal{S}_1 \overset{a}{\longmapsto} \mathcal{S}_2 \overset{\text{def.}}{\Leftrightarrow} \mathcal{S}_2 = \bigcup_{s_1 \in \mathcal{S}_1} \{s_2 \in \mathbf{S} \mid s_1 \overset{a}{\rightarrow} s_2\}.$$

Selon cette définition, s'il n'existe aucun état de $\mathcal{S}_1$ à partir duquel on puisse exécuter l'action a, alors $\mathcal{S}_2 = \emptyset$. On dit alors que la transition est *triviale*. Par conséquent, la relation $\longmapsto$ est complète: pour tout $\mathcal{S}_1 \subseteq \mathbf{S}$ et toute action $a \in \mathbf{A}$ il existe un sous-ensemble $\mathcal{S}_2 \subseteq \mathbf{S}$, tel que $\mathcal{S}_1 \overset{a}{\longmapsto} \mathcal{S}_2$. Plus encore, le sous-ensemble $\mathcal{S}_2$ est unique.

On remarque que $\mathcal{S}_2$ peut être exprimé comme suit:

$$\mathcal{S}_2 = \bigcup_{s_1 \in \mathcal{S}_1} \rho(s_1, a) \tag{4.1}$$

$$\stackrel{\text{not.}}{=} \rho(\mathcal{S}_1, a). \tag{4.2}$$

La *fonction de transition par parties* correspondant à ρ est notée par

$$\wp(\rho)\colon 2^{\mathbf{S}} \times A \longrightarrow 2^{2^{\mathbf{S}}}.$$

Le théorème suivant est un résultat important, dont la preuve suit directement des remarques et définitions précédentes.

Théorème 4.3 *Soit $M = (\mathbf{S}, A, \rightarrow)$ un STE quelconque et $\wp(M) = (\wp(\mathbf{S}), A, \longmapsto)$ le STE par parties correspondant. Alors $\wp(M)$ est un STE déterministe.*

Preuve: Nous avons déjà remarqué que pour tout $\mathcal{S}_1 \subseteq \mathbf{S}$ et toute action $a \in \mathbf{A}$ il existe un sous-ensemble unique $\mathcal{S}_2 \subseteq \mathbf{S}$ tel que $\mathcal{S}_1 \stackrel{a}{\longmapsto} \mathcal{S}_2$. Alors, selon la définition de la fonction de transition correspondante, on a que

$$\wp(\rho)(\mathcal{S}_1, a) = \{\mathcal{S}_2\}.$$

Donc, l'image de $\wp(\rho)$ appliquée à $(\mathcal{S}_1, a)$ est l'ensemble qui contient comme unique élément le sous-ensemble $\mathcal{S}_2$. Alors, par définition, $\wp$(M) est déterministe. ∎

Étant donné qu'un STE par parties est un STE, toutes les définitions et notations concernant les traces et exécutions des STE s'appliquent aux $\wp$–STE. Ainsi, on note par $\mathbb{T}(\mathcal{S}) \in A^\infty$ les traces et par $\mathbb{E}(\mathcal{S})$ les exécutions (des séquences de sous-ensembles d'états) d'une machine étendue à partir d'un sous-ensemble d'états $\mathcal{S}$. Pour ce faire nous devons quand même faire une distinction minime: la relation $\longmapsto$ est complète, par opposition à $\rightarrow$ qui ne l'est pas. Par conséquence, lorsqu'on définit les traces d'un ensemble d'états, on ne doit pas considérer les transitions triviales:

$$\mathbb{T}(\mathcal{S}) \stackrel{\text{def}}{=} \{\mu \in A^\infty \mid (\exists \mathcal{S}' \subseteq \mathbf{S})(\mathcal{S}' \neq \emptyset \ \wedge \ \mathcal{S} \stackrel{\mu}{\longmapsto} \mathcal{S}')\}.$$

Nous pouvons maintenant énoncer des résultats qui expriment les liens qui existent entre les traces d'un M et du $\wp$–STE correspondant $\wp(M)$.

Lemme 4.4 *Soit $M = (\mathbf{S}, A, \rightarrow)$ un STE quelconque et $\wp(M) = (\wp(\mathbf{S}), A, \longmapsto)$ le $\wp$–STE correspondant. Alors, pour tout sous-ensemble d'états $\mathcal{S} \subseteq \mathbf{S}$ on a que*

$$\mathbb{T}(\mathcal{S}) = \bigcup_{s \in \mathcal{S}} \mathbb{T}(s).$$

Preuve:

$$\begin{aligned}
\mathbb{T}(\mathcal{S}) &\stackrel{\text{def}}{=} \{\mu \in A^\infty \mid (\exists \mathcal{S}' \subseteq \mathbf{S})(\mathcal{S}' \neq \emptyset \ \wedge \ \mathcal{S} \stackrel{\mu}{\longmapsto} \mathcal{S}')\} \\
&= \{\mu \in A^\infty \mid (\exists s \in \mathcal{S})(\exists s' \in \mathbf{S})(\ s \stackrel{\mu}{\longmapsto} s')\} \\
&= \{\mu \in A^\infty \mid (\exists s' \in \mathcal{S})(\mu \in \mathbb{T}(s))\} \\
&= \{\mu \in A^\infty \mid \mu \in \bigcup_{s \in \mathcal{S}} \mathbb{T}(s)\} \\
&= \bigcup_{s \in \mathcal{S}} \mathbb{T}(s). \ \blacksquare
\end{aligned}$$

Corollaire 4.5 *Soit $M = (\mathbf{S}, A, \rightarrow)$ un STE quelconque et $\wp(M) = (\wp(\mathbf{S}), A, \longmapsto)$ le $\wp$–STE associé. Alors M et $\wp(M)$ ont les mêmes traces, c'est à dire, pour tout état $s \in \mathbf{S}$*

$$\mathbb{T}(s) = \mathbb{T}(\{s\}).$$

Lorsqu'aucune confusion ne sera créée, nous allons surcharger la notation $\rightarrow$ pour désigner en même temps les transitions simples et les transitions par parties $\longmapsto$.

D'une manière analogue on associe à un STEI $M^i = (M, i)$ un STEI par parties $\wp(M^i) = (\wp(\mathrm{M}), \{i\})$, où $\{i\}$ est l'ensemble qui ne contient que l'état i et $\wp(\mathrm{M})$ est le $\wp$–STE associée à M.

Les définitions et les notations des relations d'équivalence de traces $\sim$, de bisimulation forte $\simeq$ et d'isomorphisme $=_{\text{iso}}$ pour les $\wp$–STEI sont identiques aux définitions correspondantes pour les STEI simples. Lorsqu'aucune confusion ne sera possible nous surchargeons les notation $\sim$, *simeq* et $=_{\text{iso}}$ pour dsigner à la fois des relations d'équivalence de termes ou des relations d'éqiovalence de sous-ensembles de termes, lorsqu'elles s'appliquent aux STEI ou, respectivement au $\wp$-STEI.

Proposition 4.6 *Les unions d'ensembles bisimilaires d'états sont bisimilaires. Formellement, soient* $\mathcal{T}_i, \mathcal{S}_i \in 2^{\mathbf{S}}$, *avec* $i \in I$. *Alors,*

$$(\forall i \in I)(\mathcal{T}_i \simeq \mathcal{S}_i) \text{ implique } \bigcup_{i\in I} \mathcal{T}_i \simeq \bigcup_{i\in I} \mathcal{S}_i.$$

Preuve: L'idée principale de la preuve est de définir une relation $\mathcal{R} \subseteq 2^{\mathbf{S}} \times 2^{\mathbf{S}}$ qui contient toutes les paires de sous-ensembles $(\mathcal{S}, \mathcal{T}) \in \wp(\mathbf{S}) \times \wp(\mathbf{S})$ qui peuvent être écrites comme des unions d'ensembles bisimilaires.

Puisque $\simeq$ est définie comme l'union de toutes les bisimulations fortes, alors il suffit de prouver que $\mathcal{R}$ est une bisimulation forte pour déduire que $\mathcal{S} \simeq \mathcal{T}$.

Soit $\mathcal{R}$ la relation suivante:

$$\mathcal{R} = \{(\mathcal{T}, \mathcal{S}) \mid (\exists \mathcal{T}_i, S_i \in 2^{\mathbf{S}}, i \in I)(\mathcal{T} = \bigcup_{i\in I} \mathcal{T}_i \ \wedge \ \mathcal{S} = \bigcup_{i\in I} \mathcal{S}_i \ \wedge \ (\forall i \in I)(\mathcal{T}_i \simeq \mathcal{S}_i))\},$$

où I est un ensemble d'indices quelconque. Nous allons montrer que $\mathcal{R}$ est une bisimulation forte. Soit $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$ une paire quelconque de $\mathcal{R}$. Alors, $\mathcal{T}$ et $\mathcal{S}$ sont tels que

$$\begin{aligned} \mathcal{T} &= \bigcup_{i\in I} \mathcal{T}_i, \\ \mathcal{S} &= \bigcup_{i\in I} \mathcal{S}_i, \end{aligned}$$

et $\mathcal{T}_i \simeq \mathcal{S}_i$, pour tout $i \in I$.

Notons par

$$\begin{aligned} J(a, \mathcal{T}) &\stackrel{\text{not.}}{=} \{j \in I \mid a \in \text{exec}(\mathcal{T}_j)\} \\ J(a, \mathcal{S}) &\stackrel{\text{not.}}{=} \{j \in I \mid a \in \text{exec}(\mathcal{S}_j)\}. \end{aligned}$$

Puisque pour tout $i \in I$ on a que $\mathcal{T}_i \simeq \mathcal{S}_i$, on déduit que

$$J(a, \mathcal{T}) = J(a, \mathcal{S}) \stackrel{\text{not.}}{=} J(a).$$

De plus, pour toute transition $\mathcal{T}_i \stackrel{a}{\longmapsto} \mathcal{T}_i'$ il existe une transition $\mathcal{S}_i \stackrel{a}{\longmapsto} \mathcal{S}_i'$ telle que $\mathcal{T}_i' \simeq \mathcal{S}_i'$. Donc, pour toute transition $\mathcal{T} \stackrel{a}{\longmapsto} \mathcal{T}'$ il existe une transition $\mathcal{S} \stackrel{a}{\longmapsto} \mathcal{S}'$ telle

que

$$\mathcal{T}' = \bigcup_{j \in J(a)} \mathcal{T}'_j \text{ et}$$
$$\mathcal{S}' = \bigcup_{j \in J(a)} \mathcal{S}'_j$$

et telle que

$$(\forall j \in J(a))(\mathcal{T}'_j \simeq \mathcal{S}'_j).$$

Puisque $\mathcal{R}$ est symétrique, on peut alors déduire que $(\mathcal{T}', \mathcal{S}') \in \mathcal{R}$, ce qu'il fallait prouver. ∎

Nous énonçons quelques propositions, nécessaires à la preuve d'un résultat plus important, que nous appellerons *théorème fondamentale d'équivalence.*

Lemme 4.7 *Soit* $M = (\mathbf{S}, A, \rightarrow)$ *une machine à états quelconque et* $\wp(M) = (\wp(\mathbf{S}), A, \longmapsto)$ *le* $\wp$*–STE associé. Alors pour tous* $\mathcal{S}, \mathcal{T} \in \text{''}(\mathbf{S})$ *on a que*

$$\mathcal{S} \sim \mathcal{T} \Leftrightarrow \mathcal{S} \simeq \mathcal{T}.$$

Preuve: Remarquons que, selon la convention de notation de la page 56, les notations $\sim$ et $\simeq$ dénotent ici les relations d'égalité de traces et de bisimulation forte dans $\wp(M)$. Selon le théorème 4.3, le $\wp$–STE associé à M est déterministe. Alors, il suit directement du théorème 4.2 que $\mathcal{S}$ et $\mathcal{T}$ sont similaires si et seulement si ils sont bisimilaires. ∎

Nous pouvons maintenant prouver le résultat suivant, que nous avons appelé le *théorème fondamental de l'équivalence*:

Théorème 4.8 [Théorème fondamental de l'équivalence]: *Soit* $M = (\mathbf{S}, A, \rightarrow)$ *un STE et* $s, t \in \mathbf{S}$ *deux états quelconques. Alors*

$$s \sim t \Leftrightarrow \{s\} \simeq \{t\}.$$

Preuve:

$$\begin{aligned} s \sim t &\overset{\text{def.}}{\Leftrightarrow} \mathbb{T}(s) = \mathbb{T}(t) \\ &\overset{\text{Lemme 4.4}}{\Leftrightarrow} \mathbb{T}(\{s\}) = \mathbb{T}(\{t\}) \\ &\overset{\text{def.}}{\Leftrightarrow} \{s\} \sim \{t\} \\ &\overset{\text{Lemme 4.7}}{\Leftrightarrow} \{s\} \simeq \{t\} \quad \blacksquare \end{aligned}$$

Corollaire 4.9 *Soit* $M = (\mathbf{S}, A, \rightarrow)$ *une machine à états et* s *et* t *deux états quelconques de* $\mathbf{S}$. *Alors*

$$s \simeq t \Rightarrow \{s\} \simeq \{t\}.$$

Notons que l'implication inverse n'est vraie que si M est déterministe. Sinon, par le théorème 4.8 on peut suelment déduire que $\{s\} \simeq \{t\} \Rightarrow s \sim t$.

Ceci termine ce chapitre théorique, dans lequel nous avons donné les définitions des concepts liés à la notion d'algèbre de processus. Dans la suite de ce document nous définirons l'algèbre de processus temporisée ACTC. Nous commençons, dans le prochain chapitre, par la définition du langage de termes pour la spécification de chronogrammes, appelé $\mathcal{LA}$.

Chapitre 5

Le langage $\mathcal{LA}$

Le but de notre travail est de définir une algèbre de processus ACTC qui formalise la méthodologie de spécification et de vérification d'interfaces matérielles avec des *chronogrammes hiérarchiques* proposée par Cerny et Khordoc [55, 53, 31]. La première étape de la définition de l'algèbre de processus ACTC est la définition d'un *langage de termes* pour la spécification de chronogrammes, noté $\mathcal{LA}$. Les éléments du langage, appelés *termes*, sont des spécifications de chronogrammes. Le langage $\mathcal{LA}$ est défini à partir d'un ensemble d'opérateurs de composition qui forment la signature Op et de plusieurs ensembles d'opérandes qui sont composées selon une syntaxe précise.

Dans la section 5.1 nous introduisons les notions et les notations nécessaires à la définition syntaxique du langage de termes $\mathcal{LA}$.

Ensuite, dans la section 5.2, nous complétons la définition syntaxique du langage $\mathcal{LA}$ par l'introduction d'un nombre de restrictions supplémentaires sur la syntaxe des termes de $\mathcal{LA}$.

5.1 Définitions et notations

Le langage $\mathcal{LA}$ est un ensemble de *termes*, notés $T, T', S, \ldots$. Chaque terme $T \in \mathcal{LA}$ est la spécification d'un chronogramme. Dans un sens plus large, un terme désigne un processus qui exécute des actions temporisées et évolue à l'intérieur d'un espace d'états, qui est un sous-ensemble de $\mathcal{LA}$.

$\mathcal{LA}$ est défini à partir des éléments suivants:

- Op = la *signature* d'opérateurs de composition

$$\text{Op} = \{\ .\ , \trianglelefteq, \lhd,\ \|, \text{Max}, \text{Min}, :, \text{Seq}, +, \oplus, [\ ], \text{Loop}, \text{Ex}\};$$

- A = l'ensemble fini d'*actions*, notées par $a, b, \ldots$. L'ensemble A contient le sous-ensemble d'actions d'*entrée* $I = \{i_1, i_2, ...\}$, qui sont des actions de l'environnement, le sous-ensemble d'actions de *sortie* $O = \{o_1, o_2, ...\}$, qui sont les actions internes du système, le sous-ensemble d'actions de *communication* $C = \{c_1, c_2, ...\}$, qui servent à établir la communication entre des termes exécutés en parallèle, et l'action spéciale de *blocage* (en anglais *deadlock*), δ:

$$A = I \cup O \cup C \cup \{\delta\}.$$

  Le sous-ensemble d'actions $A_\beta = A \setminus C$ est nommé l'ensemble d'*actions simples*.

- T^A = l'ensemble de *variables temporelles* ; à toute action $a \in A \setminus \{\delta\}$ on associe une variable temporelle notée t_a, qui dénote le *temps d'occurrence* de l'action a et qui prend des valeurs dans D; une paire $(a, t_a) \in A \times T^A$ est appelée une *action temporisée* et est notée par $a(t_a)$;

- D = le *domaine temporel* = l'ensemble de nombres réels supérieurs ou égaux à V_0, où $V_0 \in \mathbb{R}^{\geq 0}$ est appelé l'*origine* du domaine temporel D; les constantes temporelles de D sont notées par $v, v_1, ...$ Plutôt que d'être des valeurs *absolues*,

les constantes temporelles ont une valeur *relative* à l'origine V_0. Ainsi, par convention de notation, une constante v dénote la valeur absolue $v + V_0$. Cette représentation relative facilite l'utilisation de certains outils de vérification symbolique des systèmes temporisés, tels que ceux basés sur des DDD (Difference Decision Diagrams) [65].

- Θ = l'ensemble de *contraintes temporelles descriptives*. Une contrainte temporelles $\theta \in \Theta$ décrit un système d'inégalités linéaires dont les inconnues sont des variables temporelles $t_a, t_b, ... \in T^A$.

Rappelons que nous avons défini dans la section 4.3 une *signature* $\mathcal{F} = \{f_1, \ldots, f_n\}$ comme un ensemble de symboles de fonctions muni d'une fonction d'*arité* ar : $\mathcal{F} \longrightarrow \mathbb{N}$ qui associe à chaque symbole $f \in \mathcal{F}$ un nombre naturel, appelé arité, et qui représente le nombre d'opérandes sur lesquels la fonction f est définie. On dénote par $\mathcal{F}_k$ l'ensemble de symboles de fonction de Op qui sont d'arité k. Nous avions alors défini la notion de langage de termes clos, noté $\mathcal{T}(\mathcal{F}, A)$ associé à une signature $\mathcal{F}$ et un ensemble fini A de symboles, appelées actions, comme suit:

- $A \subseteq \mathcal{T}(\mathcal{F}, A)$;
- si $f \in \mathcal{F}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, A)$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, A)$.

Selon cette définition, tous les opérandes de tout opérateur $f \in \mathcal{F}$ sont des termes et tout terme peut être une opérande de tout opérateur f.

La définition du langage de termes $\mathcal{LA}$ est plus complexe. Premièrement, les actions de $\mathcal{LA}$ sont temporisées

$$\{a(t_a) \mid a \in A, t_a \in T^A\} \subseteq \mathcal{LA}.$$

De plus, outre les termes de $\mathcal{LA}$, les opérandes des opérateurs de compositions de Op peuvent être:

- des constantes temporelles $v \in D$ (par exemple pour les opérateurs de décalage $\trianglelefteq$ et $\triangleleft$),

- des contraintes temporelles $\theta \in \Theta$ (voir l'opérateur :);
- des actions et ensembles d'actions (voir les opérateurs Min, Max, et [ ]);
- des intervalles temporels $[m, M]$ avec $m, M \in D$ et $m \leq M$ (voir les opérateurs Min, Max).

Dans les sous-sections suivantes nous définissons d'abord la syntaxe de $\mathcal{LA}$ ensuite les différents composants du langage $\mathcal{LA}$.

5.1.1 Syntaxe de $\mathcal{LA}$

Le langage $\mathcal{LA}$ est l'union du sous-langage de *termes feuilles* $\mathcal{LA}_\beta$, défini par la grammaire du tableau 5.1, et du sous-langage des *termes hiérarchiques* $\mathcal{LA}_\mathcal{H}$, défini dans le tabeau 5.2. Ces sous-langages sont construits à partir des sous-signatures suivantes: la signature des *opérateurs de base* $\mathrm{Op}_\beta \subseteq \mathrm{Op}$

$$\mathrm{Op}_\beta = \{\ .\ , \trianglelefteq, \lhd,\ \|,\ \mathrm{Max},\ \mathrm{Min}, \theta :\}.$$

et la signature des *opérateurs hiérarchiques*

$$\mathrm{Op}_\mathcal{H} = \{\mathrm{Seq}, +, \oplus, [\ ], \mathrm{Loop}, \mathrm{Ex}\}.$$

On note par Op_β^+ l'ensemble des opérateurs de base auxquels on rajoute l'opérateur de choix non-déterministe $+$:

$$\mathrm{Op}_\beta^+ = \mathrm{Op}_\beta \cup \{+\}.$$

$\mathcal{LA}_\beta$ contient deux termes particuliers: $\uparrow$ et $\downarrow$, appelés respectivement *terme d'acceptation* et de *refus*. $\uparrow$ désigne un terme qui a fini son exécution avec succès et $\downarrow$ un terme qui a fini son exécution avec échec, c'est à dire qui est arrivé dans une situation de *blocage*. Dans la section suivante nous donnerons une explication intuitive de la notion de blocage, qui sera ensuite définie formellement dans la section 6. Lorsque nous n'avons pas besoin de distinguer entre $\uparrow$ et $\downarrow$ nous notons par $\surd$ un

$$
\begin{array}{lcl}
T & = & S \mid \uparrow \mid \downarrow \\
S & = & P \mid \theta : S \mid \mathrm{Max}(o, H, [m, M], S) \mid \mathrm{Min}(o, H, [m, M], S) \mid \\
 & & \trianglelefteq_v(S) \mid \lhd_v(S) \mid \|(S, S, ...) \\
P & = & a(t_a) \mid a(t_a).P
\end{array}
$$

Tableau 5.1: Définition du langage de termes feuilles $\mathcal{LA}_\beta$

$$
\begin{array}{lcl}
T & = & S \mid \mathrm{Seq}(T, T) \mid \trianglelefteq_v(T) \mid \lhd_v(T) \\
 & & +(T, \ldots, T) \mid [\mathcal{C}, \epsilon](T, \ldots, T) \\
 & & \mid \oplus(T, \ldots, T) \mid \mathrm{Ex}(T, T, T) \mid \mathrm{Loop}(T)
\end{array}
$$

Tableau 5.2: Définition du langage $\mathcal{LA}$

des termes $\uparrow$ ou $\downarrow$ quelconque. Les termes élémentaires qui sont à la base de toute composition de termes de $\mathcal{LA}$ sont les *actions temporisées* $(a, t_a) \in A \setminus \{\delta\} \times T^A$, notées $a(t_a)$. Les termes $P = a_1(t_{a_1}) \ldots a_n(t_{a_n})$ sont appelés *ports* et sont définis comme une séquence d'actions temporisées.

Des ports peuvent être composés en parallèle avec l'opérateur $\|$.

Des contraintes sur les temps d'occurrence des actions t_{a_i} peuvent alors être définies par les opérateurs θ :, Min, et Max. On note par $\mathrm{ROp}(\cdot)$ ou $\mathrm{ROp}(o, H, [m, M], \cdot)$ un opérateur réactif quelconque, ROp = Min ou ROp = Max. L'action $o \in O$ d'un terme

$$T = \mathrm{ROp}(o, H, [m, M], S) \in \mathcal{R},$$

est appelée l'action *puits*, $H \subseteq I \cup O$ est appelé l'ensemble d'actions *sources*, et $[m, M]$ est tel que $m, M \in \mathbb{R}^{\geq 0}$ et $m \leq M$ et est appelé l'*intervalle réactif* de T.

Finalement, les exécutions des actions d'un terme T peuvent être décalées par les opérateurs de *décalage* $\trianglelefteq_v$ et/ou $\lhd_v$, avec $v \geq V_0$.

Un terme feuille T peut donc être obtenu à partir des actions de $A_\beta \setminus \{\delta\}$ par les compositions décrites plus haut, ou bien T peut être le terme d'acceptation $\uparrow$ ou de

refus $\downarrow$.

Le sous-langage de *termes hiérarchiques* $\mathcal{LA}_{\mathcal{H}}$ est défini comme

$$\mathcal{LA}_{\mathcal{H}} \stackrel{\text{def}}{=} \mathcal{LA} \setminus \mathcal{LA}_{\beta}.$$

Donc, $\mathcal{LA}_{\mathcal{H}}$ est l'ensemble de termes qui contiennent au moins un opérateur hiérarchique.

$\mathcal{LA}_{\mathcal{H}}$ contient des termes composés séquentiellement avec l'opérateur Seq, des termes qui décrivent des comportements alternatifs non-déterministes (composés avec l'opérateur $+$), ou déterministes (composés avec l'opérateur de choix retardé $\oplus$), de comportements parallèles communicants (l'opérateur [ ]), des termes à comportement répétitif (composés avec l'opérateur Loop), et des termes composés avec l'opérateur d'exception Ex. Dans le cas d'une composition parallèle avec communication $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$, $\mathcal{C} \subseteq C$ désigne un ensemble d'*actions de communication* et ϵ est une *fonction de communication* qui associe à chaque action $c \in \mathcal{C}$ un ensemble $\epsilon(c)$ d'actions des opérandes T_i. Les actions de $\epsilon(c)$ sont appelées *instances* de c. Toute action de communication c doit être exécutée par le terme T simultanément avec toutes les instances $a_i \in \epsilon(c)$ des sous-termes T_i de T. Si $c \in \mathcal{C}$ est une action de communication de T alors on dit que les sous-termes T_i qui contiennent une instance $a_i \in \epsilon(c)$ de c communiquent via c.

On remarque que les opérateurs $\trianglelefteq$ et $\triangleleft$ peuvent être considérés en même temps comme opérateurs de base et comme opérateurs hiérarchiques. Tous les termes hiérarchiques peuvent être décalés avec les opérateurs $\trianglelefteq$ et $\triangleleft$.

5.1.2 La signature Op

La signature Op contient l'opérateur de *préfixage d'actions* ".", les opérateurs de *décalage* $\trianglelefteq$ et $\triangleleft$, l'opérateur de *composition parallèle non-communicante* $\|$, les *contraintes réactives* Max et Min, les *contraintes descriptives* ":", la *composition séquentielle* Seq, le *choix non-déterministe* $+$, le *choix retardé* $\oplus$, la *composition parallèle*

communicante [ ], l'opérateur *boucle* Loop, et l'opérateur de *traitement d'exception* Ex.

Notons par $T = f(\cdot) \in \mathcal{LA}$ un terme quelconque de $\mathcal{LA}$ composé avec l'opérateur $f \in \mathrm{Op}$ et dont les opérandes ne sont pas spécifiés. Les opérandes de f qui sont des termes de $\mathcal{LA}$ sont appelés *sous-termes* de T. On note par

$$T = f[S_1, \ldots, S_n]$$

un terme composé avec un opérateur $f \in \mathrm{Op}$ et dont les sous-termes sont les termes $S_1, \ldots, S_n$. Remarquons que si f n'a pas d'autres opérandes que les sous-termes $S_1, \ldots, S_n$, alors ce n'est pas nécessaire de faire une distinction entre les notations $f(\cdot)$ et $f[\cdot]$.

On définit alors l'arité d'un opérateur quelconque $f \in \mathrm{Op}$ par le nombre d'opérandes de f qui sont des termes de $\mathcal{LA}$. Remarquons que dans $\mathcal{LA}$ tous les opérateurs sont d'arité supérieure à 0. Nous avons:

$$\begin{aligned}
\mathrm{Op}_1 &= \{:, \trianglelefteq, \triangleleft, \mathrm{Max}, \mathrm{Min}, \mathrm{Loop}\},\\
\mathrm{Op}_2 &= \{., \mathrm{Seq}\},\\
\mathrm{Op}_3 &= \{\mathrm{Ex}\},\\
\mathrm{Op}_{\geq 2} &= \{\|, +, \oplus, [\,]\},
\end{aligned}$$

où nous notons par $f \in \mathrm{Op}_{\geq 2}$ un opérateur tel que $\mathrm{ar}(f) \geq 2$. Ces opérateurs seront appelés opérateurs d'arité *arbitraire*.

Nous disons qu'un sous-terme S d'un terme $T \in \mathcal{LA}$ est *inactif* si et seulement si

1. $T = a(t_a).S$, avec $a \in A_\beta$ et $t_a \in T^A$;
2. $T = \mathrm{Seq}(T_1, S)$;
3. $T = \mathrm{Ex}(T_n, T_c, S)$.

Par négation, un sous-terme est *actif* s'il n'est pas dans une des situations énumérées ci-haut.

$\mathrm{Act}(\uparrow)$	$=$	$\emptyset$
$\mathrm{Act}(\downarrow)$	$=$	$\emptyset$
$\mathrm{Act}(a(t_a))$	$=$	$\{a\}$
$\mathrm{Act}(a(t_a).T)$	$=$	$\{a\} \cup \mathrm{Act}(T)$
$\mathrm{Act}(\|(T_1, \ldots, T_m))$	$=$	$\bigcup_{i=1}^{m} \mathrm{Act}(T_i)$
$\mathrm{Act}(\theta : T)$	$=$	$\mathrm{Act}(T)$
$\mathrm{Act}(\mathrm{ROp}(o, H, [m, M], T))$	$=$	$\mathrm{Act}(T)$

Tableau 5.3: La fonction Act appliquée aux termes de $\mathcal{LA}_\beta$

$\mathrm{Act}(\mathrm{Seq}(T_1, T_2)$	$=$	$\mathrm{Act}(T_1) \cup \mathrm{Act}(T_2)$
$\mathrm{Act}([\mathcal{C}\epsilon](T_1, \ldots, T_m))$	$=$	$\mathcal{C} \cup \bigcup_{i=1}^{m} \mathrm{Act}(T_i) \setminus \bigcup_{c \in \mathcal{C}} \epsilon(c)$
$\mathrm{Act}(\mathrm{Loop}(T))$	$=$	$\mathrm{Act}(T)$
$\mathrm{Act}(\sum_{i=1}^{m} T_i)$	$=$	$\bigcup_{i=1}^{m} \mathrm{Act}(T_i)$
$\mathrm{Act}(\bigoplus_{i=1}^{m} T_i)$	$=$	$\bigcup_{i=1}^{m} \mathrm{Act}(T_i)$
$\mathrm{Act}(\mathrm{Ex}(T_n, T_c, T_e))$	$=$	$\mathrm{Act}(T_n) \cup \mathrm{Act}(T_c) \cup \mathrm{Act}(T_e)$

Tableau 5.4: La fonction Act appliquée aux termes de $\mathcal{L}_\mathcal{H}$

5.1.3 Les actions d'un terme

Dans cette section nous formalisons le concept d'*ensemble d'actions d'un terme.* Par définition, tous les termes de $\mathcal{LA}$ sont composés à partir d'un ensemble d'actions temporisées. Nous définissons une fonction Act qui, pour chaque terme $T \in \mathcal{LA}$, retourne un ensemble d'actions $\mathrm{Act}(T) \subseteq A \setminus \{\delta\}$ appelées actions *visibles* du terme T

$$\mathrm{Act} : \mathcal{L} \longrightarrow \wp(A \setminus \{\delta\}),$$

où $\wp(A \setminus \{\delta\})$ dénote l'ensemble de sous-ensembles de $A \setminus \{\delta\}$.

La fonction Act est définie récursivement dans le tableau 5.3 pour des termes feuilles et dans le tableau 5.4 pour des termes hiérarchiques. La plupart de ces définitions sont simples et intuitives. Nous considérons que seule la définition de Act dans le cas d'un terme $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$ nécessite des explications supplémentaires. On a alors que

$$\mathrm{Act}(T) = \mathcal{C} \cup \bigcup_{i=1}^{m} \mathrm{Act}(T_i) \setminus \bigcup_{c \in \mathcal{C}}^{n} \epsilon(c).$$

Autrement dit, les instances $a_i \in \epsilon(c)$ de toute action de communication $c \in \mathcal{C}$ ne sont pas visibles dans T, elles sont remplacées par l'action de communication c qui leur correspond.

5.1.4 Les contraintes temporelles

L'ensemble T^A contient un nombre fini de variables temporelles

$$T^A = \{t_{a_1}, \ldots, t_{a_n}\},$$

où chaque variable t_{a_i} correspond à une action unique $a_i \in A \setminus \{\delta\}$ et prend des valeurs dans le domaine temporel D. On dit que T^A est *interprété* dans D. n est appelée la *dimension* de $\mathcal{LA}$.

En supposant un ordonnancement fixe des variables $t_{a_i} \in T^A$, on peut définir un espace temporel à n dimensions, noté D^A tel que

$$D^A \subseteq D^n,$$

et la i-ième dimension de D^A correspond à la variable temporelle t_{a_i}, pour tout $i, 1 \leq i \leq n$.

Un *vecteur temporel*

$$\vec{v} \stackrel{\text{def}}{=} \langle v_1, \ldots, v_n \rangle \in D^A,$$

est un $n-$*tuplet* de valeurs $v_i \in D$, telles que v_i correspond à la variable temporelle t_{a_i}, pour tout $i, 1 \leq i \leq n$. On utilise aussi les notations suivantes pour désigner le

vecteur $\vec{v}$:

$$\vec{v} \stackrel{\text{def}}{=} \langle v_1, \ldots, v_n \rangle \tag{5.1}$$

$$\stackrel{\text{not.}}{=} \langle t_{a_1} = v_1, \ldots, t_{a_n} = v_n \rangle \tag{5.2}$$

$$\stackrel{\text{not.}}{=} \langle t_{a_i} = v_i \mid a_i \in A \rangle. \tag{5.3}$$

Soit

$$A' = \{a_i, a_i, \ldots, a_k\} \subseteq A \setminus \{\delta\}$$

un sous-ensemble quelconque de A, qui préserve l'ordonnancement des actions dans A, respectivement dans T^A. Une *projection* d'un vecteur

$$\vec{v} = \langle v_1, \ldots, v_n \rangle \in D^A$$

sur A', notée $\text{Prj}(\vec{v}, A')$, est définie comme

$$\text{Prj}(\vec{v}, A') \stackrel{\text{def}}{=} \langle t_{a_i} = v_i \mid a_i \in A' \rangle \tag{5.4}$$

$$\stackrel{\text{not.}}{=} \langle \cdot, v_i, \cdot, v_j, \cdot, v_k, \cdot \rangle, \tag{5.5}$$

où chaque variable temporelle t_a telle que $a \notin A'$ est remplacée par la notation "$\cdot$". On dit alors que t_a est *non-fixée* dans $\vec{v}$ et que le vecteur $\vec{v}$ est *incomplet.* On remarque qu'un vecteur est complet si et seulement si toutes les variables de T^A sont fixées, autrement dit si et seulement si $\vec{v} \in D^A$. On note l'ensemble de tous les vecteurs, complets et incomplets, par D^{A^*} .

Un vecteur incomplet, où une variable $t_{a_i} \in T^A$ est non-fixée, décrit un sous-espace temporel noté par

$$\langle v_{a_1}, \ldots, v_{a_{i-1}}, \cdot, v_{a_{i+1}}, \ldots, v_{a_n} \rangle \stackrel{\text{not.}}{=} [\vec{v}] \subseteq D^A$$

et défini comme suit:

$$[\vec{v}] = \{\langle v_{a_1}, \ldots, v_{a_{i-1}}, v, v_{a_{i+1}}, \ldots, v_{a_n} \rangle \mid v \in D\}.$$

La fonction Act : $\mathcal{LA} \longrightarrow 2^A$ qui retourne l'ensemble des actions visibles d'un terme a été définie dans la section précédente. Puisqu'aucune confusion n'est possible,

nous surchargeons la notation Act pour désigner la fonction Act : $D^{A^*} \longrightarrow 2^A$ qui retourne pour chaque vecteur $\vec{v}$ l'ensemble d'actions fixées dans $\vec{v}$. Donc, pour tout vecteur complet $\vec{v} \in D^A$ et tout sous-ensemble d'actions $A' \subseteq A$ on a:

$$\text{Act}(\vec{v}) = A \text{ et} \tag{5.6}$$

$$\text{Act}(\text{Prj}(\vec{v}, A')) = A'. \tag{5.7}$$

Si $V = \{\vec{v_1}, \vec{v_2}, \ldots\}$ est un ensemble arbitraire de vecteurs complets

$$\vec{v_i} = \langle v_i^1, \ldots, v_i^n \rangle \in D^A$$

alors pour tout $x \in D$ on note par $V^{\prec x} \subseteq D^A$ le sous-ensemble de vecteurs de V défini comme suit:

$$V^{\prec x} = \{\vec{v_i} \in V \mid \bigwedge_{j=1}^{n} v_i^j \prec x\},$$

où $\prec$ peut être une des relations $\geq$, $>$, $\leq$, $<$, $=$.

Soit $V = \{\vec{v_1}, \vec{v_2}, \ldots, \vec{v_n}\} \subseteq D^{A^*}$ un ensemble arbitraire de vecteurs

$$\vec{v_i} = \langle t_{a_i^j} = v_i^j \mid a_i^j \in A_i \rangle \in D^{A^*},$$

où nous notons l'ensemble d'actions fixées dans un vecteur quelconque $\vec{v_i} \in V$ par A_i:

$$\text{Act}(\vec{v_i}) = A_i \subset A.$$

Pour tout $x \in D$ on note par $V^{\geq x} \subset D^{A^*}$ l'ensemble de vecteurs $\vec{v_i} \in V$ tels que les valeurs temporelles assignées aux actions fixées dans $\vec{v_i}$ sont supérieures ou égales à x. Formellement,

$$V^{\prec x} = \{\vec{v_i} = \langle v_i^j = t_{a_i^j} \mid a_i^j \in A_i \rangle \mid \bigwedge_{j=1}^{n_i} v_i^j \prec x\}, \tag{5.8}$$

où $\prec$ peut être une des relations $\geq$, $>$, $\leq$, $<$, $=$. On remarque que

$$\text{Act}(V) = \text{Act}(V^{\prec x}),$$

autrement dit les actions non-fixées dans les vecteurs $\vec{v_i}$ de V demeurent non-fixées dans $V^{\prec x}$.

Exemple 5.1 *Soit* $\mathrm{V} = \{\vec{v_1}, \vec{v_2}\} \subseteq D^{A^*}$ *un ensemble de deux vecteurs incomplets, où*

$$\begin{aligned} \vec{v_1} &= \langle t_{a_1} = 3, t_{a_2} = 10 \rangle, \\ \vec{v_2} &= \langle t_{a_3} = 8, t_{a_2} = 15 \rangle. \end{aligned}$$

Alors

$$V^{\geq 2} = V,\ V^{\geq 5} = \{v_2\},\ \text{et}\ V^{>15} = \emptyset.$$

Soit $\vec{v_3}$ *le vecteur suivant*

$$\vec{v_3} = \langle t_{a_1} = 3, t_{a_2} = 10, t_{a_4} = 0 \rangle \in [\vec{v_1}]$$

avec $a_4 \notin \{a_1, a_2, a_3\}$. *Alors, puisque l'action* a_4 *n'est fixée ni dans* V, *ni dans* $V^{\geq 2}$, $\vec{v_3}$ *est tel que*

$$\vec{v_3} \in V^{\geq 2},$$

malgré le fait que $t_{a_4} = 0 < 2$.

Une *contrainte temporelle* $\theta \in \Theta$ est un système d'inéquations linéaires, dont les inconnues sont des variables temporelles de T^A.

L'ensemble de *contraintes temporelles* $\theta \in \Theta$ est défini par la grammaire suivante

$$\theta = \text{Vrai} \mid \text{Faux} \mid t_a - t_b \prec v \mid t_a \prec v \mid \theta \wedge \theta. \tag{5.9}$$

où $t_a, t_b \in T^A$, $v \in D \cup \{\infty\}$ et $\prec$ est une des relations suivantes:

$$\prec \in \{<, \leq, >, \geq\}.$$

La notation ∞ désigne un valeur telle que, pour tout $v \in D$,

$$v < \infty \text{ et } v > -\infty$$

et telle que

$$v + \infty = \infty \text{ et } v - \infty = -\infty.$$

Une contrainte temporelle $\theta \in \Theta$ est une *conjonction* , notée $\wedge$, d'inégalités linéaires dont les inconnues sont les variables temporelles de T^A. Une contrainte θ décrit, donc, un espace temporel convexe multi-dimensionnel tel qu'à chaque variable temporelle t_a correspond une dimension. La *contrainte universelle*, notée par Vrai, décrit l'entier espace temporel et la *contrainte vide* ou Faux, l'espace vide.

On définit l'ensemble de contraintes temporelles *étendu*, noté Θ^+, comme suit:

$$\theta = \text{Vrai} \mid \text{Faux} \mid t_a - t_b \prec v \mid t_a \prec v \mid \theta \wedge \theta \mid \theta \vee \theta \tag{5.10}$$

oú t_a, t_b, v et $\prec$ sont comme dans la définition de Θ. Donc, Θ^+ contient des conjonctions et disjonctions d'inégalités linéaires sur T^A.

Nous notons par $\text{Var}(\theta)$ les variables temporelles d'une contrainte $\theta \in \Theta$

$$\text{Var} : \Theta \longrightarrow \wp(T^A).$$

Nous surchargeons la notation Act pour désigner la fonction

$$\text{Act} : \Theta \longrightarrow \wp(A \setminus \{\delta\}),$$

telle que $\text{Act}(\theta)$ retourne l'ensemble d'actions dont les variables temporelles sont concernées par la contraintes θ. Donc, si $\text{Act}(\theta) = \{a_1, \ldots, a_n\} \subseteq A$, avec $n \geq 1$, alors $\text{Var}(\theta) = \{t_{a_1}, \ldots, t_{a_n}\} \subseteq T^A$. On définit

$$\begin{aligned} \text{Act}(\text{Vrai}) &\stackrel{\text{def}}{=} A \quad \text{et} \quad \text{Var}(\text{Vrai}) = T^A \text{ et;} \\ \text{Act}(\text{Faux}) &\stackrel{\text{def}}{=} \emptyset \quad \text{et} \quad \text{Var}(\text{Faux}) = \emptyset. \end{aligned}$$

La contrainte θ dans laquelle la variable t est remplacée par la constante v est notée $\theta[v/t]$, ou bien par $\theta_{v/t}$.

Soit $\theta \in \Theta$ une contrainte quelconque tel que

$$\text{Act}(\theta) = A' = \{a_1, \ldots, a_n\} \subseteq A$$

est un sous-ensemble quelconque d'actions. Soit

$$\vec{v} = \langle t_{a_i} = v_i \mid a_i \in A' \rangle \in D^{A^*}$$

un vecteur quelconque tel que

$$\text{Act}(\theta) = \text{Act}(\vec{v})$$

L'*évaluation* de $\theta \in \Theta$ par $\vec{v} \in D^{A^*}$, notée

$$\theta[\vec{v}] \text{ ou bien } \theta_{\vec{v}},$$

est la valeur booléenne *vrai* ou *faux* obtenue en remplaçant chaque variable $t_{a_i} \in \text{Var}(\theta)$ par la valeur temporelle correspondante $v_i \in \vec{v}$ et en interprétant les opérations $\wedge$ et $\vee$ par les opérations booléennes de conjonctions et disjonctions, respectivement. Si

$$\theta[\vec{v}] = vrai$$

on dit que le vecteur $\vec{v}$ *valide* la contrainte θ. L'ensemble de vecteurs qui valident la formule θ est noté par $\text{Sol}(\theta) \subseteq D^{A^*}$:

$$\text{Sol}(\theta) \stackrel{\text{def}}{=} \{\vec{v} \in D^{A^*} \mid \text{Act}(v) = \text{Act}(\theta) \wedge \theta[\vec{v}] = vrai\}. \tag{5.11}$$

Par définition, on dit que deux contraintes $\theta_1, \theta_2 \in \Theta$ sont *égales*, notation $\theta_1 = \theta_2$ si et seulement si elles ont le même espace de solutions. :

$$\theta_1 = \theta_2 \stackrel{\text{def.}}{=} \text{Sol}(\theta_1) = \text{Sol}(\theta_2).$$

Les *contraintes vides* , c'est à dire qui n'admettent aucune solution de leur système d'inégalités, sont notées par Faux . Les *contraintes universelles*, c'est à dire dont les solutions décrivent l'entier domaine temporelle, sont notées par Vrai. Donc

$$\text{Sol(Faux)} = \emptyset \text{ et } \text{Sol(Vrai)} = D^A.$$

Soit $\theta \in \Theta$ une contrainte telle que $\text{Act}(\theta) = A_i \subseteq A$. Alors selon l'équation 5.8, $(\text{Sol}(\theta))^{\geq v} \subseteq D^{A^*}$ est l'ensemble des vecteurs qui satisfont la contrainte θ et tels que les variables temporelles de toutes les actions de A_i prennent des valeurs supérieures ou égales à v.

Exemple 5.2 *Soit $\theta \in \Theta$ la contrainte suivante:*

$$t_a = 10 \wedge t_b \geq 2.$$

Alors $\mathrm{Act}(\theta) = \{a, b\} \subset A$ *et*

$$\mathrm{Sol}(\theta) = \{\langle t_a = 10, t_b = v\rangle \mid v \geq 2\}.$$

Donc, un vecteur $\vec{v} \in D^{A^*}$ est dans $\mathrm{Sol}(\theta)$ *si et seulement si*

$$\mathrm{Prj}(\vec{v}, \{a\}) = \langle t_a = 10\rangle \text{ et } \mathrm{Prj}(\vec{v}, \{b\}) = \langle t_b = v\rangle \text{ avec } v \geq 2.$$

Alors les ensembles de vecteurs $\mathrm{Sol}^{\geq 5}(\theta), \mathrm{Sol}^{\geq 15}(\theta) \subseteq D^{A^*}$ *sont déterminés comme suit:*

$$\begin{aligned} \mathrm{Sol}^{\geq 5}(\theta) &= \{\langle t_a = 10, t_b = v\rangle \mid v \geq 5\}, \\ \mathrm{Sol}^{\geq 15}(\theta) &= \emptyset. \end{aligned}$$

Remarquons que les variables temporelles non-fixées dans θ, tel que $t_c \in D^A \setminus \mathrm{Var}(\theta)$ peuvent prendre toute valeur temporelle dans $\mathrm{Sol}^{\geq 5}(\theta)$*, même inférieure à 5. D'une autre part,*$\mathrm{Sol}^{\geq 15}(\theta)$ *ne contient aucun vecteur puisque $t_a = 10$ pour tout vecteur $\vec{v} \in \mathrm{Sol}(\theta)$ et que $t_a \in \mathrm{Var}(\theta)$ est fixée.*

Nous énonçons les propriétés suivantes, dont la preuve est triviale:

Proposition 5.1 *1.* $\mathrm{Sol}(t_a = v) = \{\langle t_a = v\rangle\}$*;*

2. $\mathrm{Sol}(\theta_1 \wedge \theta_2) = \mathrm{Sol}(\theta_1) \cap \mathrm{Sol}(\theta_2)$*;*

3. $\mathrm{Sol}(\theta_1 \vee \theta_2) = \mathrm{Sol}(\theta_1) \cup \mathrm{Sol}(\theta_2)$.

4. $\mathrm{Sol}(\theta[v/t_a]) = \mathrm{Prj}(\theta \wedge (t_a = v), \mathrm{Act}(\theta) \setminus \{a\})$*;*

5. si $\mathrm{Act}(\theta_1) \cap \mathrm{Act}(\theta_2) = \emptyset$ *alors* $\mathrm{Prj}(\mathrm{Sol}(\theta_1 \wedge \theta_2), \mathrm{Act}(\theta_i)) = \mathrm{Sol}(\theta_i)$*, pour tout* $i = 1, 2$.

Les opérations $\wedge$ et $\vee$ sont *commutatives*:

$$\theta_1 \vee \theta_2 = \theta_2 \vee \theta_1 \tag{5.12}$$

$$\theta_1 \wedge \theta_2 = \theta_2 \wedge \theta_1, \tag{5.13}$$

associatives:

$$(\theta_1 \vee \theta_2) \vee \theta_3 = \theta_1 \vee (\theta_2 \vee \theta_3) \tag{5.14}$$

$$(\theta_1 \wedge \theta_2) \wedge \theta_3 = \theta_1 \wedge (\theta_2 \wedge \theta_3), \tag{5.15}$$

distributives:

$$\theta_1 \vee (\theta_2 \wedge \theta_3) = (\theta_1 \vee \theta_2) \wedge (\theta_1 \vee \theta_3) \tag{5.16}$$

$$\theta_1 \wedge (\theta_2 \vee \theta_3) = (\theta_1 \wedge (\theta_2) \vee (\theta_1 \wedge \theta_3), \tag{5.17}$$

idempotentes:

$$\theta \vee \theta = \theta \tag{5.18}$$

$$\theta \wedge \theta = \theta \tag{5.19}$$

et ont des éléments neutres uniques:

$$\theta \vee \text{Faux} = \theta \tag{5.20}$$

$$\theta \wedge \text{Vrai} = \theta, \tag{5.21}$$

pour tous $\theta, \theta_1, \theta_2, \theta_3 \in \Theta$.

La *négation* $\neg$ d'une contrainte est alors définie comme suit:

$$\neg(t_a - t_b \leq v) = t_a - t_b > v \tag{5.22}$$

$$\neg(t_a - t_b \geq v) = t_a - t_b < v \tag{5.23}$$

$$\neg(t_a \leq v) = t_a > v \tag{5.24}$$

$$\neg(t_a \geq v) = t_a < v \tag{5.25}$$

$$\neg(\theta_1 \wedge \theta_2) = \neg\theta_1 \vee \neg\theta_2 \tag{5.26}$$

$$\neg(\theta_1 \vee \theta_2) = \neg\theta_1 \wedge \neg\theta_2. \tag{5.27}$$

Pour simplifier, on utilise souvent la notation

$$t_a = v \stackrel{\text{not.}}{=} t_a \leq v \wedge t_a \geq v \tag{5.28}$$

$$t_a - t_b = v \stackrel{\text{not.}}{=} t_a - t_b \leq v \wedge t_a - t_b \geq v. \tag{5.29}$$

Toute formule $\theta \in \Theta^+$ peut alors être écrite en *forme normale disjonctive*, notée $\text{fnd}(\theta)$, où

$$\text{fnd}(\theta) = \bigvee_{i=1}^{n} \theta_i,$$

où $n \geq 1$ et $\theta_i \in \Theta$ pour tout i, $1 \leq i \leq n$. Remarquons que si $\theta \in \Theta$, alors $\text{fnd}(\theta) = \theta$.

Pour tous $\theta_1, \theta_2 \in \Theta^+$ l'égalité suivante est triviale:

$$\theta_1 = \theta_2 \Leftrightarrow \theta_1 \wedge \neg\theta_2 = \text{Faux et } \neg\theta_1 \wedge \theta_2 = \text{Faux}.$$

Soit $\theta \in \Theta^+$ une contrainte quelconque. Alors pour vérifier si $\theta = \text{Faux}$ on suit le raisonnement suivant:

1. si $\theta \in \Theta$, donc si $\text{Sol}(\theta)$ décrit un sous-espace convexe, on peut résoudre le système d'inégalités linéaires et vérifer s'il admet ou non des solutions;

2. si $\theta \in \Theta^+$ on doit d'abord transformer θ dans sa forme normale disjonctive

   $$\text{fnd}(\theta) = \bigvee_{i=1}^{n} \theta_i,$$

   où $\theta_i \in \Theta$ sont les composantes convexes de $\text{fnd}(\theta)$. Alors,

   $$\theta = \text{Faux} \Leftrightarrow (\forall i : 1 \leq i \leq n)(\theta_i = \text{Faux}).$$

   Donc, pour chaque composante convexe θ_i on doit vérifier si $\theta_i = \text{Faux}$.

La procédure de décision de l'égalité de deux formules $\theta_1, \theta_2 \in \Theta^+$ est la suivante:

1. transformer $\theta_1 \wedge \neg\theta_2$ et $\neg\theta_1 \wedge \theta_2$ dans leur forme normale disjonctive;

2. vérifier si $\text{fnd}(\theta_1 \wedge \neg\theta_2) = \text{Faux}$ et $\text{fnd}(\neg\theta_1 \wedge \theta_2) = \text{Faux}$.

5.2 Restrictions syntaxiques

Rappelons la définition d'un langage de termes clos $\mathcal{T}(\mathrm{Op})$ associé à une signature quelconque Op que nous avons présenté à la section 4.3

- $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F})$;
- si $f \in \mathcal{F}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F})$.

Ici $\mathcal{F}_0$ désigne l'ensemble d'opérateurs d'arité 0, qui dans la définition de $\mathcal{LA}$ correspondent aux actions temporisées $a(t_a) \in A \times T^A$. On remarque que la syntaxe de $\mathcal{LA}$ telle que définie dans les tableaux 5.1 et 5.2 est plus restrictive quant à la façon de composer des termes. Par exemple, une composition

$$a(t_a).(b(t_b)\|c(t_c))$$

n'est pas permise dans $\mathcal{LA}$ puisque selon la grammaire 5.1 pour tout terme $T \in \mathcal{LA}$

$$T = a(t_a).S$$

le sous-terme S doit être un *port*, c'est à dire une composition

$$S = a_1(t_{a_1}) \ldots a_n(t_{a_n}).$$

De plus, nous imposons certaines restrictions supplémentaires sur les ensembles d'actions des termes composés. Considérons l'exemple suivant:

$$\begin{aligned} T &= T_1 \| T_2 \\ T_1 &= a_1(t_{a_1}).a_2(t_{a_2}).a_3(t_{a_3}) \\ T_2 &= b_1(t_{b_1}).a_2(t_{a_2}).b_2(t_{b_2}), \end{aligned}$$

T est la composition parallèle *non-communicante* des sous-termes T_1 et T_2. Or, T_1 et T_2 ont l'action a_2 en commun. Quelle serait alors la sémantique d'une telle composition? Est-ce que juste un des sous-termes peut exécuter l'action a_2 et l'autre se

retrouve ensuite dans une situation de blocage? Est-ce que T_1 et T_2 doivent exécuter l'action a_2 simultanément? Ces situations sont typiquement liées à la *communication* des termes et sont abordées par l'opérateur de composition parallèle communicante [ ]. Donc, pour éviter de telles situations confuses nous limitons la composition parallèle non-communicante $\|$ aux sous-termes avec des ensembles d'actions disjoints.

Formalisons ici les restrictions syntaxiques supplémentaires qui sont imposées dans la composition des termes de $\mathcal{LA}$:

1. Si $T = \|(T_1, \ldots, T_m) \in \mathcal{LA}_\beta$ alors

$$\mathrm{Act}(T_i) \cap \mathrm{Act}(T_j) = \emptyset,$$

   pour tout $1 \leq i \leq j \leq n$. Autrement dit seulement des termes avec des ensembles d'actions disjoints peuvent être composés en parallèle avec l'opérateur $\|$.

2. Si $T = \theta : S \in \mathcal{LA}_\beta$, alors

$$\mathrm{Act}(\theta) \subseteq \mathrm{Act}(S) \text{ et } \mathrm{Act}(\theta) \subseteq I.$$

   Donc, seulement les temps d'occurrences d'actions de direction entrée peuvent être contraints par des contraintes θ. De plus, θ ne concerne que les actions du sous-terme S.

3. Si $T = \mathrm{ROp}(o, H, [m, M], S) \in \mathcal{LA}_\beta$, alors

$$o \in O \setminus H \text{ et } H \cup \{o\} \subseteq \mathrm{Act}(S).$$

   Donc, l'action puits $o \in O$ ainsi que toutes les actions sources de H doivent être des actions visibles de S; de plus o ne peut pas être une action source.

4. Si $T = [\mathcal{C}\epsilon](T_1, \ldots, T_m)$ alors

$$\mathrm{Act}(T_i) \cap \mathrm{Act}(T_j) = \emptyset,$$

pour tout $i \neq j$; autrement dit les ensembles d'actions des sous-termes sont disjoints; de plus

$$\mathcal{C} \cap \bigcup_{i=1}^{m} \mathrm{Act}(T_i) = \emptyset.$$

Donc, les actions de communication ne comptent pas parmi les actions de sous-termes de T.

5. Si $T = \mathrm{Ex}(T_n, T_c, T_e)$, alors $\mathrm{Act}(T_c) \subseteq I$. On dit que la condition d'exception T_c d'un terme est un *observateur* puisqu'il ne contient que des actions de direction entrée.

Nous avons maintenant défini le langage $\mathcal{LA}$ de spécification algébrique de chronogrammes. Dans le chapitre suivant nous donnons une sémantique opérationnelle aux termes de $\mathcal{LA}$, autrement dit nous présentons une méthode formelle pour associer à chaque terme $T \in \mathcal{LA}$ un graphe de processus graphe(T), qui est un système à transitions étiquetées par des actions temporisées.

Chapitre 6

Sémantique opérationnelle

Dans ce chapitre nous définissons l'ensemble SOp des règles de sémantique opérationnelle du langage de spécification algébrique $\mathcal{LA}$ et appliquons la méthode SOS présentée dans la section 4.4 pour associer à chaque terme $T \in \mathcal{LA}$ un *graphe de processus*, noté graphe(T).

Le chapitre commence par une présentation informelle et intuitive de la sémantique de chacun des opérateurs de composition de Op (section 6.1). Dans la section 6.2.1, nous introduisons les notations et concepts de base nécessaires à la définition de la sémantique opérationnelle de $\mathcal{LA}$.

Dans les sections 6.2.2 et 6.2.3 nous allons définir la sémantique opérationnelle du langage $\mathcal{LA}$. Finalement, dans la section 6.3 nous montrons que la relation de transition $\rightarrow$ définie par les règles de SOp est fermée dans $\mathcal{LA}$.

6.1 Sémantique intuitive de $\mathcal{LA}$

Les règles de sémantique opérationnelle SOp de $\mathcal{LA}$ définissent la relation de transition d'états

$$\rightarrow \subseteq \mathcal{LA} \times A \times D \times \mathcal{LA}$$

qui est étiquetée par des *actions temporisées* $(a, v) \in A \times D$, notées $a(v)$. Un tuple $(T, a, v, T') \in \rightarrow$ est noté par

$$T \stackrel{a(v)}{\rightarrow} T'.$$

La relation $\rightarrow$ est définie comme l'ensemble de tuples $(T, a, v, T') \in \mathcal{LA} \times A \times D \times \mathcal{LA}$ tel que la formule $T \stackrel{a(v)}{\rightarrow} T'$ peut être déduite par des inférences syntaxiques à partir des règles de SOp. Dans la section 6.3 nous prouverons que cette relation est bien définie, c'est-à-dire que pour tout terme $T \in \mathcal{LA}$ et pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ déduite à partir des règles de SOp, le terme T', aussi appelé *successeur* de T, est un terme de $\mathcal{LA}$.

Par abus de langage nous dirons alors qu'un terme T exécute une action $a(v)$, et transite vers le terme T' lorsque le processus spécifié par T exécute l'action a au temps v et évolue vers le processus spécifié par le terme T'.

Lorsqu'un terme T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$ on dit que le terme T' est *démarré* au temps v. Nous définissons dans la section 6.2.1.2 la fonction α

$$\alpha : \mathcal{LA} \longrightarrow D$$

qui retourne, pour chaque terme $T \in \mathcal{LA} \setminus \{\uparrow, \downarrow\}$ son temps de démarrage $\alpha(T) \in D$. La valeur $\alpha(T)$ doit être déterminée uniquement à partir de la syntaxe de T. Dans le chapitre suivant nous prouverons que la fonction α est bien définie, c'est-à-dire que pout tout $T \in \mathcal{LA}$ et toute transition $T \stackrel{a(v)}{\rightarrow} T'$ déduite à partir des règles SOp, on a que:

$$T \stackrel{a(v)}{\rightarrow} T' \;\Rightarrow\; \alpha(T') = v, \tag{6.1}$$

$$T \stackrel{a(v)}{\rightarrow} T' \;\Rightarrow\; v \geq \alpha(T). \tag{6.2}$$

Si un terme T ne peut exécuter aucune de ses actions, alors il se trouve dans une situation de *blocage* (en anglais *deadlock*). T va alors exécuter l'action δ et transiter vers l'état de refus $\downarrow$. Le blocage est donc signalé explicitement. Ceci permet, d'une part, de détecter une situation de blocage le plus tôt possible, c'est-à-dire dès que les problèmes qui causent le blocage sont détectés. D'autre part, l'exécution de δ permet de distinguer un terme qui bloque d'un terme qui finit son exécution avec succès juste en regardant les traces d'actions du terme. La sémantique du langage $\mathcal{LA}$ est telle que toute situation de blocage est signalée, c'est-à-dire que tout terme différent de $\uparrow$ ou $\downarrow$ peut exécuter au moins une action, à la limite l'action de blocage.

Un terme dont aucune trace ne mène vers l'état de refus est appelé *réactif*. Le sous-ensemble de termes réactifs de $\mathcal{LA}$ est noté par $\mathcal{LA}^r$ et est défini comme suit:

$$\mathcal{LA}^r \stackrel{\text{def}}{=} \{T \in \mathcal{LA} \mid T \stackrel{*}{\not\rightarrow}\downarrow\}.$$

Quand un terme T exécute une action d'entrée $i \in I$ on dit aussi que T *lit* l'action i. Lorsque T exécute une action de sortie $o \in O$, on dit que T *écrit* l'action o. Un terme dont toutes les actions sont de direction entrée est appelé un *observateur*. Un terme qui ne contient aucune action de direction entrée décrit un *système fermé* et est appelé un terme *fermé*.

6.1.1 Sémantique intuitive de $\mathcal{LA}_\beta$

Donnons maintenant la sémantique intuitive détaillée des termes de $\mathcal{LA}_\beta$.

Préfixage: Un terme $T = a(t_a).S$ peut exécuter l'action a (et seulement celle-ci) à un temps quelconque $t_a = v \geq V_0$ et transite ensuite vers le terme S, décalé au temps v, c'est-à-dire tel que toute transition du successeur de T se produira au temps v ou plus tard.

Composition parallèle sans communication: Un terme $\|(S_1, \ldots, S_m)$ exécute simultanément et sans synchronisation les actions des sous-termes $S_1, \ldots S_m$. Lorsque appliqué à deux opérandes seulement, on utilise surtout la notation infixe $S_1 \| S_2$.

Décalage: $\lhd_v$ est l'opérateur de décalage strict et $\trianglelefteq_v$ le décalage non-strict, où $v \in D$ est une valeur temporelle quelconque. Un terme $T = \lhd_v(S)$, respectivement $T = \trianglelefteq_v(S)$, peut exécuter toutes les actions temporisés $a(v')$ et seulement celles que S peut exécuter et qui sont telles que $v' > v$, respectivement $v' \geq v$. Si aucune telle transition n'est possible, le terme T est dans une situation de blocage et exécute l'action δ au temps v.

Les contraintes réactives Max et Min: Les contraintes réactives déterminent le temps d'occurrence d'une action de sortie $o \in O$, appelée action *puits*, en fonction du temps d'occurrence d'un ensemble d'actions H, nommées actions *sources*, et d'un intervalle temporel $[m, M]$, dit intervalle *réactif*. $\text{Max}(o, H, [m, M], S)$ définit un processus T qui se comporte comme son sous-terme S à une restriction près: T ne peut exécuter l'action puits $o \in O$ que dans un intervalle de temps $[m, M]$ après l'occurrence de la *dernière* action de l'ensemble d'actions sources H. La sémantique d'un terme $T = \text{Min}(o, H, [m, M], S)$ est similaire: T ne peut exécuter l'action puits $o \in O$ que dans un intervalle de temps $[m, M]$ après l'occurrence de la *première* action de l'ensemble d'actions source H.

Les contraintes Min et Max sont appelées *réactives* parce qu'elles déterminent la réaction d'un système qui s'engage à exécuter ses actions de sortie dans certains délais temporels calculés en fonctions des temps d'occurrence des actions d'entrée de l'environnement (et possiblement de d'autres actions de sortie). Les opérateurs Max et Min ne contraignent donc pas les temps d'occurrence des actions de l'environnement, elles décrivent le comportement interne du système. Si, par contre, S ne peut exécuter aucune action sauf l'action puits, alors T se trouve dans une situation de blocage et doit exécuter l'action de blocage $\delta(\alpha(T))$, où $\alpha(T)$ est le temps de démarrage du terme T. Donc, dès que le terme T est démarré on peut détecter une situation de blocage et exécuter l'action δ au temps $\alpha(T)$.

Contrainte descriptive θ: Un terme $T = \theta : S$ décrit le comportement du terme S, dont les temps d'occurrence des actions sont liés par l'ensemble de con-

traintes θ. Si ces contraintes sont telles qu'elles contredisent l'information temporelle de S, le terme $\theta : S$ se trouve dans une situation de blocage et exécute l'action δ au temps de démarrage $\alpha(T)$ du terme T.

6.1.2 Sémantique intuitive de $\mathcal{LA}_{\mathcal{H}}$

Présentons la sémantique intuitive des opérateurs de composition hiérarchique de termes.

La composition séquentielle: $T = \text{Seq}(T_1, T_2)$ dénote un processus qui se comporte d'abord comme T_1 et ensuite comme T_2. Ainsi, le sous-terme T_2 est démarré dès que T_1 finit son exécution.

L'opérateur de récurrence: Un terme $T = \text{Loop}(S)$ est la composition séquentielle d'un nombre quelconque $n \geq 1$ de fois de S avec lui-même. Le nombre n est choisi de manière non-déterministe. Dans la littérature on note $\text{Loop}(S)$ souvent par S^+. On dit que T exécute une séquence d'un nombre arbitraire d'*instances* du processus S.

Le choix non-déterministe: Un terme $T = +(T_1, \ldots, T_m)$ décrit le choix non-déterministe faible entre les opérandes $T_1, \ldots, T_m$, aussi appelés *branches*. Le choix est faible dans le sens que le passage du temps n'élimine pas la possibilité de choisir n'importe quel des opérandes. Par exemple, un terme $(t_a = 1) : a(t_a).T_1 + (t_b = 3) : b(t_b).T_2$ peut aussi bien exécuter $a(1)$ et transiter vers $\lhd_1(T_1)$, qu'exécuter $b(3)$ et transiter vers $\lhd_3(T_2)$.

Nous utilisons souvent les notations suivantes:

$$+(T_1, \ldots, T_m) \overset{\text{not.}}{=} \sum_{i=1}^{m} T_i, \text{ ou}$$
$$+(T_1, \ldots, T_m) \overset{\text{not.}}{=} \sum_{i \in I} T_i, \text{ où } I = \{1, \ldots, m\}.$$

Le choix retardé: Un terme $T = \oplus(T_1, \ldots, T_m)$ est la composition des processus T_i, aussi appelés *branches* ou *alternatives*, qui ont un comportement initial commun et qui se distinguent plus tard dans leur exécution par les actions d'entrée

qu'ils peuvent lire. Le terme T a alors une phase initiale d'exécution, appelée *phase de sélection*, (qui peut être vide) pendant laquelle uniquement des actions d'entrée, peuvent être exécutées. On dit alors que l'exécution de la phase de sélection n'a pas d'*effets secondaires*. Lorsque T exécute une action d'entrée $i(v)$ pendant cette phase, il sélectionne uniquement les branches qui peuvent également exécuter l'action $i(v)$ et il poursuit la phase de sélection avec les successeurs de ces branches seulement. La phase de sélection se poursuit jusqu'à ce qu'une branche unique soit sélectionnée. Dès que la phase de sélection est finie, la restriction de ne pas produire des effets secondaires est levée: la branche sélectionnée peut alors exécuter des actions de direction sortie. Si, par contre, une branche exécute des actions de sortie pendant la phase de sélection, le terme T exécute l'action δ et signale ainsi un blocage.

Les notations suivantes sont souvent utilisées:

$$\oplus(T_1, \ldots, T_m) \stackrel{\text{not.}}{=} \bigoplus_{i=1}^{m} T_i, \text{ ou}$$

$$\oplus(T_1, \ldots, T_m) \stackrel{\text{not.}}{=} \bigoplus_{i \in I} T_i, \text{ où } I = \{1, \ldots, m\}.$$

L'opérateur d'exception: $\mathrm{Ex}(T_n, T_c, T_e)$ est un terme qui, à priori, se comporte comme le sous-terme T_n, nommé *comportement normal*. Lorsque le processus T_c , dit *condition d'exception*, finit son exécution avant T_n, on interrompt l'exécution de T_n et on démarre le processus T_e, nommé *exception*. T_c est un observateur, autrement dit il ne contient que des actions d'entrée.

La composition parallèle avec communication: L'opérateur [ ] définit une composition parallèle avec communication de type "lecture multiple/écriture exclusive" d'un nombre $m \geq 2$ quelconque de termes. Le terme

$$T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$$

est la composition parallèle des sous-termes $T_1, \ldots, T_m$ qui communiquent via les actions de l'ensemble d'*actions de communication*

$$\mathcal{C} = \{c_1, \ldots, c_n\} \subseteq C.$$

À chaque action $c \in \mathcal{C}$ on associe un ensemble d'actions

$$\epsilon(c) \subseteq \bigcup_{i=1}^{m} \mathrm{Act}(T_i)$$

appelées *instances de c*, telles que tout sous-terme T_i a tout au plus une instance de c parmi ses actions:

$$(\forall c \in \mathcal{C}, T_i \text{ sous} - \text{terme de } T)(|\ \epsilon(c) \cap \mathrm{Act}(T_i)| \leq 1).$$

Les sous-termes T_i tels que

$$|\ \epsilon(c) \cap \mathrm{Act}(T_i)| = 1$$

communiquent via l'action c. Si un sous-terme T_i communique via une action de communication T_i on utilise la notation

$$a_i = \epsilon(c, T_i) \in \mathrm{Act}(T_i)$$

pour désigner l'instance de c dans T_i.

En exécutant une action de communication c le terme T force l'exécution synchronisée de toutes les instances $a_i \in \epsilon(c)$ par les sous-termes qui communiquent via c. Les actions de $\epsilon(c)$ sont aussi dites *internes* parce qu'elles ne sont pas *visibles* dans T, c'est-à-dire qu'aucune trace de T ne peut les contenir.

6.2 Sémantique opérationnelle de $\mathcal{LA}$

Nous commençons par un rappel des définitions de SDT. Ensuite, nous définirons les notations et notions utilisées dans la définition de la sémantique opérationnelle de $\mathcal{LA}$. Finalement, nous définirons un ensemble de règles de sémantique opérationnelle, noté SOp, pour le langage $\mathcal{LA}$.

6.2.1 Définitions et notations

Avant de donner les règles de la sémantique opérationnelle de ACTC nous devons introduire un nombre de prédicats et de fonctions définis sur le langage $\mathcal{LA}$.

Commençons par un rappel succinct des définitions présentées dans la section 4.4.

6.2.1.1 SDT – rappel de définitions

La méthode SOS consiste à définir un *système de déduction de termes*, notation SDT, sur le langage $\mathcal{LA}$. La définition complète d'un SDT a été donnée dans la section 4.4. Rappelons ici les définitions les plus importantes de cette section.

Un SDT clos $\mathcal{O} = (\mathcal{T}(\mathcal{F}), \mathcal{D})$ est défini sur le langage de termes clos associé à une signature $\mathcal{F}$ par un ensemble de *règles de déduction* (aussi appelées *règles de sémantique opérationnelle*) $\mathcal{D}$.

L'ensemble $\mathcal{D} = (\mathcal{O}_p, \mathcal{O}_r)$ est paramétrisé par $\mathcal{O}_p \subseteq 2^{\mathcal{T}(\mathcal{F})}$ et $\mathcal{O}_r \subseteq 2^{\mathcal{T}(\mathcal{F})\times\mathcal{T}(\mathcal{F})}$, qui représentent respectivement un ensemble de symboles de *prédicats* et de *relations* de termes du langage de terme $\mathcal{T}(\mathcal{F})$.

Soient $P \in \mathcal{O}_p$ un symbole de prédicat et $\mathcal{R} \in \mathcal{O}_r$ un symbole de relation et soient $s, t, u \in \mathcal{T}(\mathcal{F})$ des termes quelconques. On appelle les expressions Ps et $t\mathcal{R}u$ des *formules*. Une *règle de déduction* $d \in \mathcal{D}$ a la forme suivante:

$$\frac{H}{C}$$

où H est un ensemble de formules appelées *hypothèses* de d et C est une formule, dite *conclusion* de d. Informellement, une formule ψ est prouvable dans $\mathcal{O}$ et on écrit

$$\mathcal{O} \vdash \psi$$

si ψ peut être déduite à partir des règles de $\mathcal{D}$ par induction structurelle.

Soit $\mathcal{O} = (T(\mathcal{F}), \mathrm{SOp})$ un SDT avec $\mathrm{SOp} = (\mathcal{O}_p, \mathcal{O}_r)$. Le *système de transition induit* par $\mathcal{O}$, noté $\mathrm{ST}(\mathcal{O}) = (\mathbf{S}, \rightarrow)$ est défini comme suit:

- l'ensemble d'états est $\mathbf{S} = \mathcal{T}(\mathcal{F})$,
- la relation de transition $\rightarrow$ est telle que

$$\rightarrow = \{(s,t) \in \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}) \mid (\exists \mathcal{R} \in \mathcal{O}_r)\ (\mathcal{O} \vdash s\mathcal{R}t)\}.$$

Pour définir à partir de $\mathcal{O}$ un système de transitions étiqueté par un ensemble d'actions temporisées $A \times D$, où D est un domaine temporel quelconque, l'ensemble $\mathcal{O}_r$ doit contenir une relation $\mathcal{R}_{a,v}$ pour chacune des actions $a \in A$:

$$\mathcal{O}_r = \{\mathcal{R}_{a,v} \mid a \in A, v \in D\}.$$

Notons ce STE par STE($\mathcal{O}$). La relation de transition étiquetée est alors définie comme:

$$\rightarrow = \{(s, a, v, t) \in \mathcal{T}(\mathcal{F}) \times A \times D \times \mathcal{T}(\mathcal{F}) \mid \mathcal{O} \vdash s\mathcal{R}_a t\}.$$

Donc, pour donner une sémantique opérationnelle aux termes clos de $\mathcal{T}(\mathcal{F})$, il suffit de définir un SDT $\mathcal{O} = (\mathcal{T}(\mathcal{F}), \mathcal{D})$, qui associe le système à transitions étiquetées STE($\mathcal{O}$) aux langage de termes $\mathcal{T}(\mathcal{F})$. Ensuite, tel que présenté dans la section 4.1, on définit pour chaque terme clos $t \in \mathcal{T}(\mathcal{F})$ le STEI, noté graphe(t), qui est la partie atteignable du système à transitions STE($\mathcal{O}$) initialisé par t.

6.2.1.2 Fonctions, prédicats et relations dans SOp

Dans cette section nous définissons les prédicats et fonctions par lesquels l'ensemble de règles de sémantique opérationnelle SOp du langage $\mathcal{LA}$ est paramétrisé.

La fonction α

Rappelons que dans la section 6.1 nous avons présenté d'une manière intuitive la notion de démarrage d'un terme quelconque $T \in \mathcal{LA}$. Donnons maintenant la définition formelle de la fonction de démarrage α.

Nous notons par $\mathcal{LA}_0 \subset \mathcal{LA}$ le sous-ensemble de termes qui ne contiennent aucun opérateur de décalage:

$$\mathcal{LA}_0 \stackrel{\text{def}}{=} \mathcal{LA} \cap \mathcal{T}(\text{Op} \setminus \{\trianglelefteq, \triangleleft\}).$$

$\mathcal{LA}_0$ est appelé le sous-langage de *termes initiaux*. Nous prouverons dans le chapitre suivant que selon les règles de SOp il n'existe aucune transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $T' \in \mathcal{LA}_0$ – d'où la notion de terme initial.

Définition 6.1 *La fonction de démarrage*

$$\alpha : \mathcal{LA} \setminus \{\uparrow, \downarrow\} \longrightarrow D$$

est définie récursivement pour tout terme $T \in \mathcal{LA}$ *différent de* $\uparrow$ *et* $\downarrow$ *comme suit:*

1. *si* $T \in \mathcal{LA}_0$ *alors* $\alpha(T) = V_0$;
2. *si* $T = \lhd_v(S)$ *alors* $\alpha(T) = v$;
3. *si* $T = \trianglelefteq_v(S)$ *alors* $\alpha(T) = v$;
4. *si* $T = f[T_1, \ldots, T_n]$ *avec* $f \in \mathrm{Op} \setminus \{\trianglelefteq, \lhd\}$ *et* $n \geq 1$ *alors* $\alpha(T) = \min_{\{T_i \text{ actif}\}} \alpha(T_i)$.

Rappelons que, par définition, un sous-terme S d'un terme quelconque $T \in \mathcal{LA}$ est dit inactif si et seulement si

1. $T = a(t_a).S$, avec $a \in A_\beta$ et $t_a \in T^A$ ou
2. $T = \mathrm{Seq}(T_1, S)$ ou encore
3. $T = \mathrm{Ex}(T_n, T_c, S)$.

Par négation, un sous-terme est *actif* s'il n'est pas dans une des situations énumérées ci-haut.

La fonction Γ

Tout terme de $T \in \mathcal{LA}$ contient de l'information temporelle, qui est soit *réactive* (spécifiée par les opérateurs réactifs Max et Min), ou bien *descriptive*. L'information descriptive peut être implicite, telle que l'ordonnancement des temps d'occurrence d'actions composées séquentiellement, ou spécifiée explicitement par les contraintes descriptives θ. L'information temporelle θ décrit les suppositions que le terme fait sur le fonctionnement de son environnement et l'information réactive décrit les propriétés internes du terme: le terme doit respecter les contraintes réactives si son environnement satisfait les contraintes descriptives.

$\Gamma(\uparrow)$	$=$	$\text{Sol}(\text{Vrai}) = D^A$
$\Gamma(\downarrow)$	$=$	$\text{Sol}(\text{Faux}) = \emptyset$
$\Gamma(a(t_a))$	$=$	$\text{Sol}(t_a \geq V_0)$
$\Gamma(a(t_a)\ .\ T)$	$=$	$\Gamma(T) \cap \text{Sol}(t_a \geq V_0 \wedge (\bigwedge_{b \in \text{Act}(T)} t_b > t_a))$
$\Gamma(\|(T_1, \ldots, T_m))$	$=$	$\bigcap_{i=1}^{m} \Gamma(T_i)$
$\Gamma(\trianglelefteq_v(T))$	$=$	$\Gamma^{\geq v}(T)$
$\Gamma(\triangleleft_v(T))$	$=$	$\Gamma^{>v}(T)$
$\Gamma(\theta : T)$	$=$	$\text{Sol}(\theta) \cap \Gamma(T)$
$\Gamma(\text{ROp}(T))$	$=$	$\Gamma(T)$

Tableau 6.1: La fonction Γ appliquée aux termes de $\mathcal{LA}_\beta$

Nous définissons la fonction

$$\Gamma : \mathcal{LA} \longrightarrow D^{A^*}$$

qui, pour tout terme $T \in \mathcal{LA}$, retourne l'ensemble de vecteurs qui forment l'espace temporel défini par les contraintes descriptives d'un terme ainsi que par l'ordonnencement implicite des actions sur les ports d'un terme quelconque. Rappelons que D^{A^*} a été défini dans la section 5.1.4 comme l'ensemble de vecteurs, complets ou incomplets, de valeurs temporelles correspondant aux variables temporelles de T^A. La définition inductive de la fonction Γ pour des termes de $T \in \mathcal{LA}_\beta$ est donnée dans le tableau 6.1 et pour des termes hiérarchiques dans le tableau 6.2.

La notation $\Gamma^{\prec x}$ a la signification que nous lui avons donnée dans le chapitre précédent, équation 5.8, pour un ensemble quelconque de vecteurs $V \subseteq D^{A^*}$. Donc,

$$\Gamma^{\prec x}(T) = \Gamma(T) \cap \text{Sol}(\bigwedge_{a \in \text{Act}(\Gamma(T))} t_a \prec x),$$

où $\prec \in \{\leq, <, \geq, >\}$.

Le prédicat First

Le prédicat $\text{First}(a(v), T)$ est vrai si et seulement si l'exécution de $a(v)$ comme

$$
\begin{array}{lcl}
\Gamma(\mathrm{Seq}(T_1, T_2)) & = & \Gamma(T_1) \\
\Gamma([\mathcal{C}, \epsilon](T_1, \ldots, T_m)) & = & (\bigcap_{i=1}^{m} \Gamma(T_i))_{t_c/t_a}, \forall c \in \mathcal{C}, \forall a \in \epsilon(c) \\
\Gamma(\mathrm{Loop}(T)) & = & \Gamma(T) \\
\Gamma(\sum_{i=1}^{m} T_i) & = & \bigcup_{i=1}^{m} \Gamma(T_i) \\
\Gamma(\bigoplus_{i=1}^{m} T_i) & = & \bigcup_{i=1}^{m} \Gamma(T_i) \\
\Gamma(\mathrm{Ex}(T_c, T_n, T_e)) & = & \Gamma(T_n) \cap \Gamma(T_c) \\
\Gamma(\trianglelefteq_v(f[T_1, \ldots, T_n])) & = & \Gamma^{\geq v}(f[T_1, \ldots, T_n])), \text{ si } f \in \mathrm{Op} \setminus \{+, \oplus\} \\
 & = & \bigcup_{i=1}^{n} \Gamma^{\geq v}(T_i), \text{ si } f \in \{+, \oplus\} \\
\Gamma(\triangleleft_v(f[T_1, \ldots, T_n])) & = & \Gamma^{> v}(f[T_1, \ldots, T_n])), \text{ si } f \in \mathrm{Op} \setminus \{+, \oplus\} \\
 & = & \bigcup_{i=1}^{n} \Gamma^{> v}(T_i), \text{ si } f \in \{+, \oplus\}
\end{array}
$$

Tableau 6.2: La fonction Γ appliquée aux termes de $\mathcal{L}_{\mathcal{H}}$

première action de T ne contredit pas l'information temporelle descriptive du terme T.

$$\mathrm{First}(a(v), T) \overset{\text{def.}}{\Leftrightarrow} \Gamma^{\geq v}(T) \cap \mathrm{Sol}(t_a = v) \neq \emptyset$$

La fonction exec

La fonction exec est définie pour chaque terme T et désigne toutes les actions temporisées que T peut exécuter:

$$\mathrm{exec}(T) = \{a(v) \mid (\exists T' \in \mathcal{LA})(T \xrightarrow{a(v)} T')\}.$$

Nous utilisons aussi les notations suivantes:

$$
\begin{array}{rcl}
\mathrm{exec}^{\delta}(T) & = & \{a(v) \in \mathrm{exec}(T) \mid a \neq \delta\} \\
\mathrm{exec}^{>v}(T) & = & \{a(v') \in \mathrm{exec}(T) \mid v' > v\} \\
\mathrm{exec}^{\geq v}(T) & = & \{a(v') \in \mathrm{exec}(T) \mid v' \geq v\} \\
\mathrm{exec}^{<v}(T) & = & \{a(v') \in \mathrm{exec}(T) \mid v' < v\}
\end{array}
$$

$$\text{exec}^{\leq v}(T) \;=\; \{a(v') \in \text{exec}(T) \mid v' \leq v\}.$$

Le prédicat Wait

Le prédicat $\text{Wait}(v, T)$ est défini pour tout terme $T \in \mathcal{LA}$ et tout $v \geq 0$. $\text{Wait}(v, T)$ est vrai si et seulement si T peut attendre jusqu'au temps v avant d'exécuter ses actions. Formellement, $\text{Wait}(v, T)$ est vrai si et seulement s'il existe une action quelconque $a \in A$ que T peut exécuter au temps v ou plus tard:

$$\text{Wait}(v, T) \stackrel{\text{def.}}{\Leftrightarrow} (\exists a(v') \in \text{exec}(T))(v' \geq v).$$

Autrement dit

$$\text{Wait}(v, T) \Leftrightarrow \text{exec}^{\geq v}(T) \neq \emptyset.$$

Notons ici que dans la section 8.3.1 nous allons prouver que pour tout terme feuille $T \in \mathcal{LA}_\beta$ le prédicat est vrai si et seulement si toute action de $\text{exec}(T)$ peut encore être exécutée au temps v ou plus tard.

Le prédicat Urgent

Finalement, on définit le prédicat $\text{Urgent}(a(v), T)$ qui est vrai si et seulement si l'exécution de l'action a par le terme T est urgente au temps v, c'est-à-dire si T peut exécuter a au plus tard au temps v. Formellement,

$$\text{Urgent}(a(v), T) \stackrel{\text{def.}}{\Leftrightarrow} (\exists T' \in \mathcal{LA}) T \stackrel{a(v)}{\rightarrow} T' \;\wedge\; (\forall v' > v) T \stackrel{a(v')}{\not\rightarrow}.$$

6.2.1.3 Formules dans SOp

Contrairement aux définitions présentées dans la section 4.4, où les règles de sémantique opérationnelles étaient définies sur tout le langage de termes $\mathcal{T}(\text{Op})$, l'ensemble de règles de sémantique opérationnelle SOp que nous définissons ne concerne que les termes du langage $\mathcal{LA} \subset \mathcal{T}(\text{Op})$ défini dans le chapitre précédent. On peut alors considérer que les termes de $\mathcal{T}(\text{Op})$ qui ne satisfont pas aux restrictions syntaxiques

du langage $\mathcal{LA}$ ne peuvent effectuer aucune transition. Formellement,

$$(\forall T \in \mathcal{T}(\mathrm{Op}) \setminus \mathcal{LA}, a \in A, v \in D, T' \in \mathcal{T}(\mathrm{Op}))(T \overset{a(v)}{\not\to} T')$$

Pour que le SDT $\mathcal{O}(\mathcal{LA}, \mathrm{SOp})$ soit bien défini alors, il va falloir prouver que le sous-langage $\mathcal{LA}$ est fermé aux transitions, c'est-à-dire que pour tout terme $T \in \mathcal{LA}$ et toute transition $T \overset{a(v)}{\to} T'$ le successeur T' est aussi un terme de $\mathcal{LA}$. Cette propriété et d'autres propriétés importantes de fermeture seront prouvées dans la section 6.3, après avoir introduit toutes les définitions nécessaires.

Pour le langage algébrique $\mathcal{LA}$ les hypothèses H des règles de sémantique opérationnelle peuvent être de la forme suivante:

1. des formules concernant la relation de transition $\to$, avec $T, S \in \mathcal{LA}$, $a \in A$, et $v \geq 0$:
   - $T \overset{a(v)}{\to} S$;
   - $T \overset{a(v)}{\not\to}$;
2. des formules concernant l'ensemble exec pour $T \in \mathcal{L}$, $v \geq 0$ et $o \in O$:
   - $\mathrm{exec}^{<v}(T) = \emptyset$;
   - $\mathrm{exec}^{>v}(T) = \emptyset$;
   - $\mathrm{exec}^{\geq v}(T) = \emptyset$;
   - $\mathrm{exec}(T) = \{o\}$;
3. des formules $\mathrm{Wait}(v, T)$, pour $T \in \mathcal{L}$ et $v \geq 0$;
4. des formules $\mathrm{First}(a(v), T)$, pour $T \in \mathcal{LA}_\beta$, $a \in A_\beta \setminus \{\delta\}$, et $v \geq 0$;
5. des formules $\mathrm{Urgent}(a(v), T)$, pour $T \in \mathcal{L}$, $a \in A$, et $v \geq 0$;
6. de propositions variés telles que $v \geq v'$, etc.;

7. des quantifications universelles sur des univers finis, par exemple

$$(\forall i \in \{1, ...m\})(\ Wait(v, T_i))$$

ou infinis, par exemple

$$(\forall a(v) \in \text{exec}(T))(v \geq v_0);$$

8. des négations, conjonctions, et disjonctions des formules décrites plus haut.

La conclusion de toute règle de SOp

$$\frac{H}{C} \in \text{SOp},$$

est une formule:

$$C = T \stackrel{a(v)}{\rightarrow} T',$$

où $T \in \mathcal{LA}, a \in \text{A}$, et $v \in D$.

Nous pouvons maintenant définir l'ensemble de règles de sémantique opérationnelle du langage $\mathcal{LA}$. Nous présentons les règles concernant les termes feuilles dans la section 6.2.2 et celles concernant les termes hiérarchiques dans la section 6.2.3.

6.2.2 Sémantique opérationnelle de $\mathcal{LA}_\beta$

Les tableaux 6.3 et 6.4 définissent les règles de sémantique opérationnelle des termes feuilles de $\mathcal{LA}_\beta$ dans le style de Plotkin [74].

Nous utilisons les conventions de notation suivantes:

- variables $v, v', v_1, \ldots \in D$
- par défaut $a, b, \ldots \in A_\beta \setminus \{\delta\}$; lorsqu'une transition $T \stackrel{a(v)}{\rightarrow} T'$ peut être aussi considérée comme un blocage, ceci sera signalé explicitement en notant que $a \in A$;
- par défaut $T, T', T_i, T_i' \in \mathcal{LA}_\beta \setminus \{\uparrow, \downarrow\}$; si T, T' peuvent désigner aussi un terme inactif, ceci sera indiqué explicitement en notant $T, T' \in \mathcal{LA}_\beta$.

- Préfixage

$$\mathbf{P}_1)\ \frac{v \geq V_0}{a(t_a)\ .\ T \overset{a(v)}{\rightarrow} \lhd_v(T)} \qquad \mathbf{P}_2)\ \frac{v \geq V_0}{a(t_a) \overset{a(v)}{\rightarrow} \uparrow}$$

- Composition parallèle

$$\mathbf{Par}_1)\ \frac{T_i \overset{a(v)}{\rightarrow} T_i',\ \forall j \neq i\ \mathrm{Wait}(v, T_j)}{\|(T_1, \ldots, T_i, \ldots, T_m) \overset{a(v)}{\rightarrow} \|(\trianglelefteq_v T_1, \ldots, \trianglelefteq_v(T_{i-1}), T_i', \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_m))}$$

$$\mathbf{Par}_2)\ \frac{T_i \overset{a(v)}{\rightarrow} \uparrow,\ \forall j \neq i\ \mathrm{Wait}(v, T_j)}{\|(T_1, \ldots, T_i, \ldots, T_m) \overset{a(v)}{\rightarrow} \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{i-1}), \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_m))} \qquad m > 2$$

$$\mathbf{Par}_3)\ \frac{T_i \overset{a(v)}{\rightarrow} \uparrow, \mathrm{Wait}(v, T_j)}{\|(T_1, T_2) \overset{a(v)}{\rightarrow} \trianglelefteq_v(T_j)} \quad i \in \{1, 2\}\ , i \neq j$$

$$\mathbf{Par}_4)\ \frac{T_i \overset{\delta(v)}{\rightarrow} \downarrow,\ (\forall j \neq i)(\mathrm{exec}^{\leq v}(T_j) = \emptyset)}{\|(T_1, \ldots, T_m) \overset{\delta(v)}{\rightarrow} \downarrow}$$

- Décalage

$$\mathbf{D}_1)\ \frac{T \overset{a(v')}{\rightarrow} T', v' \geq v}{\trianglelefteq_v(T) \overset{a(v')}{\rightarrow} T'} \quad a \in A, T' \in \mathcal{LA} \qquad \mathbf{D}_2)\ \frac{\mathrm{exec}^{\geq v}(T) = \emptyset}{\trianglelefteq_v(T) \overset{\delta(v)}{\rightarrow} \downarrow}$$

- Décalage strict

$$\mathbf{Ds}_1)\ \frac{T \overset{a(v')}{\rightarrow} T', v' > v}{\lhd_v(T) \overset{a(v')}{\rightarrow} T'} \quad a \in A, T' \in \mathcal{LA} \qquad \mathbf{Ds}_2)\ \frac{\mathrm{exec}^{>v}(T) = \emptyset}{\lhd_v(T) \overset{\delta(v)}{\rightarrow} \downarrow}$$

- Supposition

$$\mathbf{supp}_1)\ \frac{T \overset{a(v)}{\rightarrow} T', \mathrm{First}(a(v), \theta : T)}{\theta : T \overset{a(v)}{\rightarrow} \theta_{v/t_a} : T'} \qquad \mathbf{supp}_2)\ \frac{T \overset{a(v)}{\rightarrow} \uparrow, \mathrm{First}(a(v), \theta : T)}{\theta : T \overset{a(v)}{\rightarrow} \uparrow}$$

$$\mathbf{supp}_3)\ \frac{T \overset{\delta(v)}{\rightarrow} \downarrow}{\theta : T \overset{\delta(v)}{\rightarrow} \downarrow} \qquad \mathbf{supp}_4)\ \frac{\forall a(v) \in \mathrm{exec}(T)\ \neg\mathrm{First}(a(v), \theta : T)}{\theta : T \overset{\delta(\alpha)}{\rightarrow} \downarrow}, \quad \alpha = \alpha(\theta : T)$$

Tableau 6.3: Sémantique opérationnelle pour les opérateurs $.$, $\|$, $\trianglelefteq_v, \lhd_v$, θ

- **Max**

$$\textbf{Max}_1)\ \frac{T \xrightarrow{a(v)} T',\ |H - \{a\}| \geq 1}{\mathrm{Max}(o, H, [m, M], T) \xrightarrow{a(v)} \mathrm{Max}(o, H - \{a\}, [m, M], T')}, \quad a \in A_\beta \setminus \{\delta, o\}$$

$$\textbf{Max}_2)\ \frac{T \xrightarrow{a(v)} T'}{\mathrm{Max}(o, \{a\}, [m, M], T) \xrightarrow{a(v)} m \leq t_o - v \leq M : T'}, \quad a \in A_\beta \setminus \{\delta, o\}$$

$$\textbf{Max}_3)\ \frac{T \xrightarrow{a(v)} \surd}{\mathrm{Max}(o, H, [m, M], T) \xrightarrow{a(v)} \surd} \quad a \in A_\beta \setminus \{o\}$$

$$\textbf{Max}_4)\ \frac{\mathrm{exec}(T) = \{o\}}{\mathrm{Max}(o, H, [m, M], T) \xrightarrow{\delta(\alpha)} \downarrow} \quad \alpha = \alpha(\mathrm{Max}(o, H, [m, M], T))$$

- **Min**

$$\textbf{Min}_1)\ \frac{T \xrightarrow{a(v)} T', a \notin H}{\mathrm{Min}(o, H, [m, M], T) \xrightarrow{a(v)} \mathrm{Min}(o, H, [m, M], T')}, \quad a \in \mathbf{A}_\beta \setminus \{\delta, o\}$$

$$\textbf{Min}_2)\ \frac{T \xrightarrow{a(v)} T', a \in H}{\mathrm{Min}(o, H, [m, M], T) \xrightarrow{a(v)} m \leq t_o - v \leq M : T'}, \quad a \in \mathbf{A}_\beta \setminus \{\delta, o\}$$

$$\textbf{Min}_3)\ \frac{T \xrightarrow{a(v)} \surd}{\mathrm{Min}(o, H, [m, M], T) \xrightarrow{a(v)} \surd} \quad a \in A_\beta \setminus \{o\}$$

$$\textbf{Min}_4)\ \frac{\mathrm{exec}(T) = \{o\}}{\mathrm{Min}(o, H, [m, M], T) \xrightarrow{\delta(\alpha)} \downarrow} \quad \alpha = \alpha(\mathrm{Min}(o, H, [m, M], T))$$

Tableau 6.4: Sémantique opérationnelle des opérateurs Max et Min

Expliquons maintenant les règles de sémantique opérationnelle des tableaux 6.3 et 6.4.

Décalage temporel non-strict: Un terme $\trianglelefteq_v(T)$ décalé par une constante temporelle v ne peut effectuer que les transitions de T qui ont lieu à un temps supérieur ou égal à v ($\trianglelefteq_{v1}$). Si aucune telle transition n'est possible, $\trianglelefteq_v(T)$ signalise un blocage au temps v ($\trianglelefteq_{v2}$).

Décalage temporel strict: L'opérateur $\triangleleft$ a la même sémantique que le décalage $\trianglelefteq$ à l'exception du fait que les transitions admissibles doivent avoir lieu à un temps strictement supérieur à v.

Préfixage d'actions: Un terme $a(t_a).T$ peut exécuter uniquement l'action a, à un temps $v \geq 0$ quelconque. Puisque l'exécution d'une action prend un temps non-nul, le terme T est ensuite décalé strictement au temps v;

Composition parallèle sans communication: La règle $\mathbf{Par}_1$ stipule que dans un terme $\|(T_1, \ldots, T_m)$ un sous-terme ne peut exécuter une action quelconque à un temps donné v que si tous les l'autres sous-termes peuvent attendre jusqu'au temps v. Autrement dit il n'y a aucune autre action dont l'exécution soit plus urgente. La règle $\mathbf{Par}_2$ correspond à la situation où un des sous-termes T_i finit son exécution avec succès au temps v. La composition parallèle m-aire est alors remplacée par la composition $(m-1)$-aire des sous-termes restants, décalés par l'opérateur $\trianglelefteq_v$. Remarquons que cette règle ne s'applique que si le terme a au moins trois opérandes. Le cas de la composition parallèle binaire est explicité par la règle $\mathbf{Par}_3$: lorsqu'un des deux sous-termes T_i transite vers l'état $\uparrow$ au temps v, la composition parallèle est désactivée et remplacée par le sous-terme restant, décalé par l'opérateur $\trianglelefteq_v$. Dès qu'un des sous-termes bloque, selon $\mathbf{Par}_4$, le terme composé bloque aussi. Le successeur d'un terme composé avec l'opérateur $\|$ est décalé par l'opérateur $\trianglelefteq_v$, contrairement aux termes préfixés, dont les successeurs sont décalés par l'opérateur $\triangleleft$. On permet ainsi la concurrence des actions exécutées en parallèle, tout en assurant la monotonie temporelle des séquences d'exécutions d'actions.

La contrainte descriptive θ: Soit $\theta : T$ un terme et supposons que $T \stackrel{a(v)}{\rightarrow} T'$. Selon $\mathbf{supp}_1$, si l'exécution de a au temps v comme première action ne contredit pas les suppositions θ, le terme $\theta : T$ peut l'exécuter. On remplace alors toute référence à la variable temporelle t_a par le temps d'occurrence v de l'action a. $\theta : T$ finit son exécution lorsque T peut exécuter sa dernière action sans violer les contraintes temporelles ($\mathbf{supp}_2$). Le terme $\theta : T$ bloque si le sous-terme T bloque ($\mathbf{supp}_3$), ou bien si aucune transition possible de T ne satisfait aux contraintes de θ ($\mathbf{supp}_4$).

L'opérateur réactif Max: Le terme $\mathrm{Max}(o, H, [m, M], T)$ contraint le temps de l'action puits o par rapport au temps d'occurrence de l'action source qui sera exécutée en dernier parmi les actions source de l'ensemble H. Les deux premières règles donnent les conditions dans lesquelles le terme $\mathrm{Max}(o, H, [m, M], T)$ peut évoluer vers un terme actif. Si $T \stackrel{a(v)}{\rightarrow} T'$ et si a n'est pas l'action cible o, alors selon la règle $\mathbf{Max}_1$, l'opérateur Max est transféré au successeur T' de T, avec quelques modifications: l'action a est éliminée de l'ensemble d'actions sources. Si a était l'unique action source de H et implicitement aussi la dernière action source, alors, selon la règle $\mathbf{Max}_2$, le terme T' sera contraint par la relation temporelle $m \leq t_o - v \leq M$. Ceci implique que l'action o devra être exécutée à l'intérieur de l'intervalle réactif. $\mathrm{Max}(o, H, [m, M], T)$ termine avec succès si et seulement si le sous-terme T termine avec succès ($\mathbf{Max}_3$). Le terme bloque si son sous-terme T bloque ($\mathbf{Max}_3$), ou bien si la seule action que T peut exécuter est l'action puits ($\mathbf{Max}_4$).

L'opérateur réactif Min: Le terme $\mathrm{Min}(o, H, [m, M], T)$ contraint le temps de l'action puits o par rapport au temps d'occurrence de l'action source de H qui sera exécutée en premier. La sémantique de l'opérateur Min ne diffère de celle de Max que par la règle concernant l'exécution d'une action source. Supposons que $T \stackrel{a(v)}{\rightarrow} T'$, où $a \neq o$. Si a est une action source, selon la règle $\mathbf{Min}_2$, implicitement elle doit être la première action de H à être exécutée: l'opérateur réactif est alors remplacé par la contrainte temporelle $m \leq t_o - v \leq M$. Ainsi l'exécution de l'action puits o se fera à l'intérieur de l'intervalle réactif.

6.2.3 Sémantique opérationnelle de $\mathcal{LA}_{\mathcal{H}}$

Les tableaux 6.5, 6.6, et 6.8 présentent l'ensemble de règles de sémantique opérationnelle des opérateurs hiérarchiques dans le style de Plotkin [74].

Les conventions de notation suivantes seront utilisées:

- par défaut $a, b, ... \in A \setminus \{\delta\}$ donc, différentes de δ; le cas contraire sera précisé explicitement;
- par défaut $T, T', T_i, T_i', \ldots \in \mathcal{LA} \setminus \{\uparrow, \downarrow\}$ (sauf indication contraire);
- $\trianglelefteq(\uparrow) \stackrel{\text{not.}}{=} \uparrow$,
- $\triangleleft(\uparrow) \stackrel{\text{not.}}{=} \uparrow$,
- $+(T) \stackrel{\text{not.}}{=} T$,
- $\oplus(T) \stackrel{\text{not.}}{=} T$,
- $[\mathcal{C}, \epsilon](T) \stackrel{\text{not.}}{=} T$,
- $+(T_1, \ldots, T_{i-1}, \uparrow, T_{i+1}, \ldots, T_n) \stackrel{\text{not.}}{=} +(T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_n)$,
- $\oplus(T_1, \ldots, T_{i-1}, \uparrow, T_{i+1}, \ldots, T_n) \stackrel{\text{not.}}{=} \oplus(T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_n)$,
- $[\mathcal{C}, \epsilon](T_1, \ldots, T_{i-1}, \uparrow, T_{i+1}, \ldots, T_n) \stackrel{\text{not.}}{=} [\mathcal{C}, \epsilon](T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_n)$.

où $n \geq 2$.

Dans les sections suivantes nous donnons des explications détaillées des règles de sémantique opérationnelle des termes hiérarchiques.

La composition séquentielle: La composition séquentielle $\text{Seq}(T_1, T_2)$ de deux termes est un processus qui exécute d'abord toutes les actions de T_1 (voir la règle $\mathbf{Seq_1}$), qui démarre l'exécution du deuxième terme T_2 dès que le premier terme a fini son exécution avec succès (règle $\mathbf{Seq_2}$), et qui bloque lorsque le premier terme T_1 bloque (règle $\mathbf{Seq_3}$).

- Choix non-déterministe

$$\sum_1) \quad \frac{T_j \overset{a(v)}{\rightarrow} T'_j}{\sum_{i=1}^{m}(T_i) \overset{a(v)}{\rightarrow} T'_i}$$

$$\sum_2) \quad \frac{T_i \overset{a(v)}{\rightarrow} \uparrow}{\sum_{i=1}^{m}(T_i) \overset{a(v)}{\rightarrow} \uparrow}$$

$$\sum_3) \quad \frac{T_i \overset{\delta(v)}{\rightarrow} \downarrow \ \wedge \ (\forall j \neq i)(T_j \overset{\delta(v)}{\rightarrow} \downarrow \ \vee \neg \mathrm{Wait}(v, T_j))}{\sum_{i=1}^{m}(T_i) \overset{\delta(v)}{\rightarrow} \downarrow}$$

- Composition séquentielle

$$\mathbf{Seq_1}) \quad \frac{T_1 \overset{a(v)}{\rightarrow} T'_1}{\mathrm{Seq}(T_1, T_2) \overset{a(v)}{\rightarrow} \mathrm{Seq}(T'_1, T_2)} \qquad \mathbf{Seq_2}) \quad \frac{T_1 \overset{a(v)}{\rightarrow} \uparrow}{\mathrm{Seq}(T_1, T_2) \overset{a(v)}{\rightarrow} \lhd_v(T_2)}$$

$$\mathbf{Seq_3}) \quad \frac{T_1 \overset{\delta(v)}{\rightarrow} \downarrow}{\mathrm{Seq}(T_1, T_2) \overset{\delta(v)}{\rightarrow} \downarrow}$$

- Boucle

$$\mathbf{B_1}) \quad \frac{T \overset{a(v)}{\rightarrow} T'}{\mathrm{Loop}(T) \overset{a(v)}{\rightarrow} \mathrm{Seq}(T', \mathrm{Loop}(T))} \qquad \mathbf{B_2}) \quad \frac{T \overset{a(v)}{\rightarrow} T'}{\mathrm{Loop}(T) \overset{a(v)}{\rightarrow} T'}$$

$$\mathbf{B_3}) \quad \frac{T \overset{a(v)}{\rightarrow} \uparrow}{\mathrm{Loop}(T) \overset{a(v)}{\rightarrow} \lhd_v(\mathrm{Loop}(T))} \qquad \mathbf{B_4}) \quad \frac{T \overset{a(v)}{\rightarrow} \uparrow}{\mathrm{Loop}(T) \overset{a(v)}{\rightarrow} \uparrow}$$

$$\mathbf{B_5}) \quad \frac{T \overset{\delta(v)}{\rightarrow} \downarrow}{\mathrm{Loop}(T) \overset{\delta(v)}{\rightarrow} \downarrow}$$

Tableau 6.5: Sémantique opérationnelle des opérateurs de choix non-déterministe, de composition séquentielle, et de boucle

Composition parallèle avec communication

Notations:

$T \quad = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$

$I(a) = \{i \mid \mathrm{Act}(T_i) \cap \epsilon(a) \neq \emptyset\}$

$J(a) = \{1, \ldots m\} \setminus I(a)$

$a_i \quad = \epsilon(a, T_i), i \in I(a)$

$\mathcal{C}' = \{c \in \mathcal{C} \mid \epsilon'(c) \neq \emptyset\}$

$\forall c \in \mathcal{C}',\ \epsilon'(c) \stackrel{\text{def}}{=} \epsilon(c) \cap (\bigcup_{i \in I(c)} \mathrm{Act}(T_i') \bigcup_{j \in J(c)} \mathrm{Act}(T_j)),$

$$\textbf{Com}_1)\ \frac{(\forall i \in I(a))(T_i \stackrel{a_i(v)}{\rightarrow} T_i'),\ (\forall j \in J(a))(\mathrm{Wait}(v, T_j))}{T \stackrel{a(v)}{\rightarrow} [\mathcal{C}', \epsilon']_{\substack{i \in I(a) \\ j \in J(a)}} (T_i', \trianglelefteq_v(T_j))}, \quad a \in \mathrm{Act}(T), T_i' \in \mathcal{LA}$$

$$\textbf{Com}_2)\ \frac{T_i \stackrel{\delta(v)}{\rightarrow}\downarrow, \forall j \neq i\ \mathrm{Wait}(v, T_j)}{T \stackrel{\delta(v)}{\rightarrow}\downarrow}$$

$$\textbf{Com}_3)\ \frac{T_i \stackrel{oc(v)}{\rightarrow} T_i',\ T_j \stackrel{ic(v)}{\not\rightarrow},\ (\forall k \neq i)(\mathrm{Wait}(v, T_k))}{T \stackrel{\delta(v)}{\rightarrow}\downarrow}, \qquad oc \mid ic, T_i' \in \mathcal{L}$$

$$\textbf{Com}_4)\ \frac{\mathrm{Urgent}(T_i, ic(v)),\ T_j \stackrel{a_j(v)}{\not\rightarrow},\ (\forall k \neq i)(\mathrm{Wait}(v, T_k))}{T \stackrel{\delta(v)}{\rightarrow}\downarrow} \qquad ic \mid a_j$$

Tableau 6.6: Sémantique opérationnelle de l'opérateur de composition parallèle avec communication

Choix retardé

Notations:

$T \stackrel{\text{def}}{=} \bigoplus_{i=1}^{m} T_i$,

$J^\delta \stackrel{\text{def}}{=} \{j \in \{1,\ldots,m\} \mid \delta \notin \text{exec}(T_j)\}$,

$v^\delta \stackrel{\text{def}}{=} \max_{i\in\{1,\ldots,m\}}\{v_i \mid T_i \stackrel{\delta(v_i)}{\rightarrow}\downarrow\}$

$I \subseteq \{1,\ldots,m\}$

$$\mathbf{CR}_1)\ \frac{\mathbf{Hyp}_1(T)}{\mathbf{Conc}_1(T)} \stackrel{\text{def}}{=} \frac{(\forall i \in I)(T_i \stackrel{ic(v)}{\rightarrow} T_i'),\ (\forall j \in \{1,\ldots,m\}\setminus I)(T_j \stackrel{ic(v)}{\not\rightarrow}\uparrow)}{T \stackrel{ic(v)}{\rightarrow} \bigoplus_{i\in I} T_i'}$$

$$\mathbf{CR}_2)\ \frac{\mathbf{Hyp}_2(T)}{\mathbf{Conc}_2(T)} \stackrel{\text{def}}{=} \frac{T_j \stackrel{ic(v)}{\rightarrow}\uparrow,\ T_i \stackrel{ic(v)}{\rightarrow} T_i',\ T_i' \neq\uparrow}{T \stackrel{\delta(v)}{\rightarrow}\downarrow}$$

$$\mathbf{CR}_3)\ \frac{\mathbf{Hyp}_3(T)}{\mathbf{Conc}_3(T)} \stackrel{\text{def}}{=} \frac{T_j \stackrel{ic(v)}{\rightarrow}\uparrow,\ (\forall i \in \{1,\ldots,m\})(T_i \stackrel{ic(v)}{\rightarrow}\uparrow\ \vee\ T_i \stackrel{ic(v)}{\not\rightarrow})}{T \stackrel{ic(v)}{\rightarrow}\uparrow}$$

$$\mathbf{CR}_4)\ \frac{\mathbf{Hyp}_4(T)}{\mathbf{Conc}_4(T)} \stackrel{\text{def}}{=} \frac{T_i \stackrel{oc(v)}{\rightarrow} T_i'}{T \stackrel{\delta(v)}{\rightarrow}\downarrow},\ T_i' \in \mathcal{L}$$

$$\mathbf{CR}_5)\ \frac{\mathbf{Hyp}_5(T)}{\mathbf{Conc}_5(T)} \stackrel{\text{def}}{=} \frac{J^\delta = \emptyset}{T \stackrel{\delta(v^\delta)}{\rightarrow}\downarrow}$$

$$\mathbf{CR}_6)\ \frac{\mathbf{Hyp}_6(T)}{\mathbf{Conc}_6(T)} \stackrel{\text{def}}{=} \frac{J^\delta = \{i\}, T_i \stackrel{a(v)}{\rightarrow} T_i', v > v^\delta}{T \stackrel{a(v)}{\rightarrow} T_i'},\ T_i' \in \mathcal{L}, a \in A$$

$$\mathbf{CR}_7)\ \frac{\mathbf{Hyp}_7(T)}{\mathbf{Conc}_7(T)} \stackrel{\text{def}}{=} \frac{2 \leq |J^\delta| < m, \mathbf{Hyp}_k(\bigoplus_{j\in J^\delta} \trianglelefteq_{v^\delta}(T_j))}{\mathbf{Conc}_k(T)},\ 1 \leq k \leq 6$$

Tableau 6.7: Sémantique opérationnelle de l'opérateur $\oplus$

- **Exception**

$$\mathbf{Ex}_1)\ \frac{T_n \overset{ic(v)}{\rightarrow} T'_n, T_c \overset{ic(v)}{\rightarrow} T'_c}{\mathrm{Ex}(T_n, T_c, T_e) \overset{ic(v)}{\rightarrow} \mathrm{Ex}(T'_n, T'_c, T_e)}$$

$$\mathbf{Ex}_2)\ \frac{T_n \overset{a(v)}{\rightarrow} T'_n, T_c \overset{a(v)}{\not\rightarrow}, \mathrm{Wait}(v, T_c)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{a(v)}{\rightarrow} \mathrm{Ex}(T'_n, \trianglelefteq_v(T_c), T_e)}$$

$$\mathbf{Ex}_3)\ \frac{T_n \overset{ic(v)}{\not\rightarrow}, T_c \overset{ic(v)}{\rightarrow} T'_c, \mathrm{Wait}(v, T_n)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{ic(v)}{\rightarrow} \mathrm{Ex}(\trianglelefteq_v(T_n), T'_c, T_e)}$$

$$\mathbf{Ex}_4)\ \frac{T_c \overset{ic(v)}{\rightarrow} \uparrow, \mathrm{Wait}(v, T_n)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{ic(v)}{\rightarrow} \trianglelefteq_v(T_e)} \qquad \mathbf{Ex}_5)\ \frac{T_n \overset{a(v)}{\rightarrow} \uparrow, \mathrm{Wait}(v, T_c)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{a(v)}{\rightarrow} \uparrow}$$

$$\mathbf{Ex}_6)\ \frac{T_c \overset{\delta(v)}{\rightarrow} \downarrow, \mathrm{Wait}(v, T_n)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{\delta(v)}{\rightarrow} \downarrow} \qquad \mathbf{Ex}_7)\ \frac{T_n \overset{\delta(v)}{\rightarrow} \downarrow, \mathrm{Wait}(v, T_c)}{\mathrm{Ex}(T_n, T_c, T_e) \overset{\delta(v)}{\rightarrow} \downarrow}$$

Tableau 6.8: Sémantique opérationnelle de l'opérateur Ex

La composition parallèle avec communication: L'opérateur $[\mathcal{C}, \epsilon]$ définit une communication de type *lecture multiple/écriture unique* qui supporte le paradigme de *causalité*. Nous allons expliquer ces concepts avant de présenter les règles de sémantique opérationnelle de l'opérateur.

Rappelons qu'un terme $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$ est la composition parallèle des sous-termes $T_1, \ldots, T_m$ qui doivent se synchroniser via l'ensemble d'actions de communication $\mathcal{C} \subseteq C$: à chaque action $c \in \mathcal{C}$ correspond un ensemble d'actions

$$\epsilon(c) \subseteq \bigcup_{i=1}^{m} \mathrm{Act}(T_i)$$

tel que tout sous-terme a tout au plus une action dans l'ensemble $\epsilon(c)$. Formellement,

$$|\epsilon(c) \cap \mathrm{Act}(T_i)| \leq 1,$$

pour toute action $c \in \mathcal{C}$ et tout $i : 1 \leq i \leq m$. Puisque les ensembles d'actions des sous-termes T_i sont disjoints par définition, toutes les actions de $\epsilon(c)$ associées à c sont différentes les unes des autres.

On dit que les sous-termes T_i qui ont une action $a_i \in \mathrm{Act}(T) \cap \epsilon(c)$ *communiquent* entre eux via c. L'action a_i est alors notée par $\epsilon(c, T_i) = a_i$ et est appelée l'action *interne* de T_i associée à c.

T exécute une action de communication $c \in \mathcal{C} \times D$ si et seulement si tous les sous-termes T_i qui communiquent via c exécutent simultanément leur action interne $a_i = \epsilon(c, T_i)$ correspondante. Remarquons qu'une action $a_i \in \mathrm{Act}(T_i)$ peut correspondre à plus d'une seule action de communication.

On dit qu'un processus *lit* une action si elle est spécifiée comme action d'entrée dans ce processus; l'action est *écrite* par le processus si elle est spécifiée comme action de sortie dans le processus. L'opérateur hiérarchique de composition parallèle définit une communication de type *écriture unique/lecture multiple*: une même action peut être lue simultanément par un nombre arbitraire de processus mais elle ne peut être écrite que par un seul processus, tout au plus. Alternativement, elle est une action

de l'environnement global (*lecture seulement*). Concrètement ce principe de communication se traduit par la restriction suivante: pour chaque action $c \in \mathcal{C}$ l'ensemble $\epsilon(c)$ contient tout au plus une action de direction sortie, toutes les autres étant de direction entrée.

Lorsque des termes sont composés en parallèle hiérarchiquement à plusieurs niveaux, on doit appliquer ce même principe de communication récursivement. Une action de communication c d'un terme hiérarchique T peut devenir interne lorsqu'associée à une action de communication c' d'un niveau hiérarchique supérieur. Considérons l'exemple suivant:

Exemple 6.1 *Soient $T_1, T_2, T_3 \in \mathcal{LA}_\beta$ les termes feuilles suivants:*

$$\begin{aligned} T_1 &= i_1(t_{i_1}) || i_2(t_{i_2}) \\ T_2 &= o_1(t_{o_1}) \\ T_3 &= i_3(t_{i_3}).i_4(t_{i_4}).i_5(t_{i_5}). \end{aligned}$$

Soient $S_1, S_2 \in \mathcal{LA}$ et des termes hiérarchiques tels que

$$\begin{aligned} S_1 &= [\mathcal{C}_1, \epsilon_1](T_1, T_2), \\ \mathcal{C}_1 &= \{c_1\}, \\ \epsilon_1(c_1) &= \{i_1, o_1\}, \end{aligned}$$

et

$$\begin{aligned} S_2 &= [\mathcal{C}_2, \epsilon_2](S_1, T_3), \\ \mathcal{C}_2 &= \{c_2, c_3\}, \\ \epsilon_2(c_2) &= \{c_1, i_3\}, \\ \epsilon_2(c_3) &= \{i_2, i_4\}. \end{aligned}$$

Dans le terme S_1, les actions i_1 et o_1 sont associées à l'action de communication c_1 et deviennent des actions internes. L'ensemble d'actions visibles de S_1 est alors

$\mathrm{Act}(S_1) = \{c_1, i_2\}$. *Dans le terme* S_2, S_1 *est composé avec* T_3. *L'exécution de* i_2 *et* i_4 *se synchronise via* c_3. *On remarque que l'action de communication* c_1 *est l'action interne de* S_1 *qui correspond à* c_2.

Tout terme $T = f(T_1, \ldots, T_m) \in \mathcal{LA}_{\mathcal{H}}$, $m \geq 1$, où $f \in \mathrm{Op}_{\mathcal{H}}$ est un opérateur hiérarchique, est récursivement composé à partir de termes feuilles. Ainsi toute action $a \in \mathrm{Act}(T)$ est soit une action simple ou bien correspond à un ensemble d'actions simples, appelés *descendants* de a. L'ensemble de descendants d'une action a est noté $\mathrm{desc}_T(a)$ et est défini récursivement comme suit:

- $\mathrm{desc}_T(i) = \{i\}$;
- $\mathrm{desc}_T(o) = \{o\}$;
- $\mathrm{desc}_T(c) = \bigcup_{a_i = \epsilon(c, T_i)} \mathrm{desc}_{T_i}(a_i)$, si $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$, $m \geq 2$;
- $\mathrm{desc}_T(c) = \bigcup_{T_i} \mathrm{desc}_{T_i}(c)$, pour tout sous-terme T_i tel que $c \in \mathrm{Act}(T_i)$.

En appliquant le principe de communication hiérarchiquement, on déduit qu'un terme T ne peut exécuter une action de communication c que si toutes les actions simples de $\mathrm{desc}(c)$ peuvent être exécutées par les termes feuilles qui sont à la base de la composition hiérarchique de T. Par conséquent, selon le principe de l'écriture exclusive, pour toute action de communication $c \in C$ il ne peut y avoir qu'une seule action de sortie tout au plus dans l'ensemble $\mathrm{desc}(c)$:

$$(\forall a \in A)(|\mathrm{desc}(a) \cap O| \leq 1).$$

Une action de communication dont tous les descendants sont des actions d'entrée est notée par *ic*. Une action de communication dont l'ensemble des descendants contient une action de sortie est notée *oc*.

Exemple 6.2 *Soit S_2 le terme de l'exemple 6.1. La fonction* desc *appliquée aux actions de* $\mathrm{Act}(S_2) = \{c_2, c_3, i_5\}$ *retourne les ensembles d'actions suivants:*

$$\begin{aligned}
\mathrm{desc}_{S_2}(c_2) &= \{i_3\} \cup \mathrm{desc}_{S_1}(c_1) \\
&= \{i_3\} \cup \{i_1, o_1\} \\
&= \{i_3, i_1, o_1\}; \\
\mathrm{desc}_{S_2}(c_3) &= \{i_2, i_4\}; \\
\mathrm{desc}_{S_2}(i_5) &= \{i_5\}.
\end{aligned}$$

La composition parallèle avec communication satisfait au principe de causalité: lorsque plusieurs processus partagent une action, dont un processus en mode écriture, le processus qui écrit peut choisir le temps d'exécution de l'action à l'intérieur de son intervalle réactif. Les autres processus en mode lecture vérifient alors si cette exécution satisfait aux suppositions qu'ils ont faites sur l'environnement, sinon le processus composé bloque. Donc, lorsqu'une action de sortie o communique avec un ensemble d'actions (d'entrée), le temps d'occurrence de l'action de communication correspondante est déterminé uniquement par l'action de sortie. Pour qu'une action ic puisse être exécutée il doit y avoir un instant où tous les sous-termes qui communiquent via ic puissent exécuter leur action interne correspondante. Par contre, lorsqu'une action de communication oc correspond à une action de sortie $o \in \mathrm{Act}(T_i)$ alors à tout instant où T_i peut exécuter o, les autres sous-termes communicants doivent pouvoir exécuter leur action interne $a_i(v)$ correspondante. Cette caractéristique est, à notre connaissance, unique parmi les opérateurs de composition parallèle avec communication que l'on rencontre dans la littérature.

Finalement, on peut étendre la fonction ϵ aussi aux actions de T qui ne sont pas des actions de communication: on définit $\epsilon(a) = \{a\}$ pour toute action $a \in \mathrm{Act}(T) \setminus \mathcal{C}$. Plus encore, on peut définir $\epsilon(a) = \{a\}$, pour toute action de $A \setminus \{\delta\}$.

On note $a_1 \mid a_2$ [56] et on dit que deux actions $a_1, a_2 \in \mathrm{Act}(T)$ *communiquent*, si

elles correspondent à la même action de communication. Formellement,

$$a_1 \mid a_2 \Leftrightarrow (\exists c \in \mathcal{C})(a_1 \in \epsilon(c) \wedge a_2 \in \epsilon(c)).$$

Remarquons ici que $\mid$ est commutative, transitive, et réflexive. La relation $\mid$ est donc une relation d'équivalence sur l'ensemble d'actions d'un terme. Nous pouvons maintenant expliquer les règles de sémantique opérationnelle de la composition parallèle avec communication. Soit $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m) \in \mathcal{L}$ et $a \in \mathrm{Act}(T)$ une action quelconque de T. Dans le tableau 6.6 nous introduisons les notations suivantes:

- $I(a) = \{i \in \{1, \ldots, m\} \mid \mathrm{Act}(T_i) \cap \epsilon(a) \neq \emptyset\}$ est l'ensemble d'indices des sous-termes T_i qui communiquent via l'action $a \in \mathrm{Act}(T)$. En particulier, si $a \in \mathrm{Act}(T) \setminus \mathcal{C}$, alors il existe un unique sous-terme $T_i \in I(a)$, notamment le sous-terme tel que $a \in \mathrm{Act}(T_i)$ (parce que les ensembles d'actions des sous-termes T_i sont disjoints par définition);

- $J(a) = \{j \in \{1, \ldots m\} \mid \mathrm{Act}(T_j) \cap \epsilon(a) = \emptyset\}$ est l'ensemble d'indices des sous-termes T_j qui ne communiquent pas via l'action a;

- $a_i = \epsilon(a, T_i)$ est l'action interne du sous-terme T_i qui correspond à l'action a. L'action a_i n'est définie que pour les sous-termes T_i tels que $i \in I(a)$. De plus, si $a \notin \mathcal{C}$ et $I(a) = \{i\}$, alors $a_i = a$.

La règle $\mathbf{Com}_1$ décrit les conditions d'avancement du terme T, c'est-à-dire les transitions $T \overset{a(v)}{\rightarrow} T'$ avec $a \neq \delta$. Pour que T puisse exécuter une action quelconque $a(v) \in \mathrm{Act}(T)$ il faut que tous les sous-termes T_i qui communiquent via l'action a puissent exécuter leur action interne correspondante $a_i = \epsilon(a, T_i)$ au temps v et que tous les autres sous-termes T_j puissent attendre jusqu'au temps v. Formellement,

$$(\forall i \in I(a))(T_i \overset{a_i(v)}{\rightarrow} T_i') \ \wedge \ (\forall j \in J(a))\mathrm{Wait}(v, T_j).$$

Si ces conditions sont satisfaites T peut exécuter $a(v)$ et transite ensuite vers

$$T' = [\mathcal{C}', \epsilon']_{\substack{i \in I(a), \\ j \in J(a)}}(T_i', \trianglelefteq_v(T_j)),$$

où T_i' est le successeur de T_i après l'exécution de l'action $a_i(v)$. Nous appliquons ici les conventions de notation 6.2.3, c'est-à-dire aucun terme $T_i' = \uparrow$, n'est considéré dans la composition du terme T' et la composition parallèle d'un seul sous-terme dénote le sous-terme même.

On distingue alors les situations suivantes:

1. si T' a au moins deux sous-termes différents de $\uparrow$, alors T' est une vraie composition parallèle;
2. si $J(a) = \emptyset$ et $T_i' = \uparrow$, pour tout $i \in I(a)$, alors $T' = \uparrow$;
3. si $J(a) = \emptyset$ et $T_i' = \uparrow$, pour tout $i \in I(a) \setminus \{k\}$, alors $T' = T_k'$;
4. finalement, si $J(a) = \{k\}$ et $T_i' = \uparrow$, pour tout $i \in I(a)$, alors $T' = \trianglelefteq_v(T_k)$.

Puisque $\mathrm{Act}(T) \neq \mathrm{Act}(T')$ nous devons redéfinir l'ensemble d'actions de communication $\mathcal{C}'$ et la fonction de communication ϵ' pour le terme T'. Supposons, par exemple, que T exécute une action de communication $c \in \mathcal{C}$. On se pose alors la question si $\mathcal{C}'$ doit contenir l'action c ou non.

D'une part, définir $\mathcal{C}'$ simplement comme $\mathcal{C} \setminus \{c\}$ pourrait devenir inadéquat si T décrit un comportement répétitif. Considérons l'exemple suivant:

Exemple 6.3 *Soient* $T_1 = \mathrm{Loop}(o(t_o))$ *et* $T_2 = \mathrm{Loop}(i(t_i))$ *deux termes.* T_1 *écrit l'action de sortie* $o \in O$ *et* T_2 *répète la lecture d'une action d'entrée* $i \in I$ *un nombre indéfini de fois. Supposons que nous voulons spécifier le fait que* T_2 *lit l'action que* T_1 *écrit. Une manière naturelle de spécifier ceci serait de définir un terme* $T = [\mathcal{C}, \epsilon](T_1, T_2)$ *tel que* $\mathcal{C} = \{c\}$ *et* $\epsilon(c) = \{o, i\}$.

Alors $\mathrm{Act}(T) = \{c\}$ *et* T *ne peut exécuter* $c(v)$ *que s'il existe des transitions* $T_1 \xrightarrow{o(v)} \triangleleft_v(\mathrm{Loop}(o(t_o)))$ *et* $T_2 \xrightarrow{i(v)} \triangleleft_v(\mathrm{Loop}(i(t_i)))$.

Si C' *etait défini comme* $\mathcal{C} \setminus \{c\}$, *on obtiendrait la transition suivante:*

$$T \xrightarrow{c(v)} T' = [\emptyset, \epsilon'](\triangleleft_v(\mathrm{Loop}(o(t_o))), \triangleleft_v(\mathrm{Loop}(i(t_i))))$$

et puisque $\mathcal{C}' = \emptyset$, il n'y aurait plus de communication entre les sous-termes de T'. Donc, on ne peut pas simplement éliminer l'action c de l'ensemble $\mathcal{C}'$.

D'autre part, si nous décidons de garder l'action c dans C' et de définir simplement comme $\mathcal{C}' \stackrel{\text{def}}{=} \mathcal{C}$ nous préservons dans T' de l'information sur des actions qui ne seront jamais exécutées. L'exemple suivant décrit cette situation:

Exemple 6.4 *Soit $T = [\{c_1, c_2\}, \epsilon](T_1, T_2)$ tel que les sous-termes*

$$T_1 = i_1(t_{i_1}).i_2(t_{i_2}),\ T_1 = o_1(t_{o_1}).o_2(t_{o_2})$$

communiquent selon la fonction suivante:

$$\epsilon(c_1) = \{i_1, o_1\},\ \epsilon(c_2) = \{i_2, o_2\}.$$

Alors T peut exécuter l'action $c_1(v)$ pour tout $v \in D$ et transite ensuite vers le terme

$$T' = [\mathcal{C}', \epsilon'](\lhd_v(i_2(t_{i_2}), \lhd_v(o_2(t_{o_2})).$$

Si $\mathcal{C}' = \mathcal{C}$ alors l'action c_1 devient redondante puisque les sous-termes de T' ne communiquent pas via c_1.

La solution est la suivante: soit une transition quelconque

$$[\mathcal{C}, \epsilon](T_1, \ldots, T_n) \stackrel{a(v)}{\rightarrow} [\mathcal{C}', \epsilon']_{\substack{i \in I(a) \\ j \in J(a)}} (T_i', \trianglelefteq_v(T_j))$$

telle que tous les termes qui communiquent via l'action a peuvent exécuter leur action interne correspondante $a_i = \epsilon(a, T_i)$:

$$(\forall i \in I(a)) T_i \stackrel{a_i(v)}{\rightarrow} T_i'$$

et que le prédicat $\text{Wait}_v(T_j)$ est vrai pour tous les sous-termes $T_j, j \in J(a)$ qui ne communiquent pas avec a. On définit alors $\mathcal{C}'$ comme suit:

$$\mathcal{C}' = \mathcal{C} \setminus \{c \in \mathcal{C} \mid \epsilon(c) \cap (\bigcup_{i \in I(a)} \text{Act}(T_i') \bigcup_{j \in J(a)} \text{Act}(T_j)) = \emptyset\}$$

Autrement dit, on soustrait de l'ensemble $\mathcal{C}$ les actions de communication c telles que leur ensemble d'actions internes $\epsilon(c)$ ne contient aucune action interne dans les sous-termes de T_i' et $\trianglelefteq_v(T_j)$ de T'.

Avec cette définition, dans l'exemple 6.3 on obtient que $\mathcal{C}' = \mathcal{C}$ et dans l'exemple 6.4 que $\mathcal{C}' = \mathcal{C} \setminus \{c_1\}$.

Puisque $\mathcal{C}' \subseteq \mathcal{C}$, la fonction de communication

$$\epsilon' : \mathcal{C}' \longrightarrow \bigcup_{i \in I(a)} \mathrm{Act}(T_i') \bigcup_{j \in J(a)} \mathrm{Act}(T_j)$$

est tout simplement la projection de ϵ sur $\mathcal{C}'$.

Exemple 6.5 *Soient $T_1, T_2, T_3 \in \mathcal{LA}_\beta$ les termes feuilles suivants:*

$$\begin{aligned} T_1 &= o_1(t_{o_1}).o_2(t_{o_2}), \\ T_2 &= o_3(t_{o_3}).o_4(t_{o_4}), \\ T_3 &= i_1(t_{i_1}).i_2(t_{i_2}). \end{aligned}$$

Nous définissons le terme hiérarchique $T \in \mathcal{LA}$ comme $T = [\mathcal{C}, \epsilon](T_1 + T_2, T_3)$, où

$$\begin{aligned} \mathcal{C} &= \{c_1, c_2, c_3, c_4\}, \\ \epsilon(c_1) &= \{o_1, i_1\}, \\ \epsilon(c_2) &= \{o_2, i_2\}, \\ \epsilon(c_3) &= \{o_3, i_1\}, \\ \epsilon(c_4) &= \{o_4, i_2\}. \end{aligned}$$

Remarquons que l'action interne i_1 est associée aux actions de communication c_1 et c_3. Le sous-terme $T_1 + T_2$ va choisir de manière non-déterministe de suivre la branche T_1 ou bien la branche T_2. Les actions d'entrée de T_3 vont alors se synchroniser soit avec les actions de sortie de T_1 ou bien avec celles de T_2. Ainsi, si T_1 effectue une transition $T_1 \overset{o_1(v_1)}{\rightarrow} \trianglelefteq_{v_1}(o_2(t_{o_2}))$, alors i_1 va se synchroniser avec o_1 via l'action de

communication c_1. T *va donc effectuer la transition* $T \xrightarrow{c_1} T'$, *où*

$$\begin{aligned} T' &= [\mathcal{C}', \epsilon'](\lhd_{v_1}(o_2(t_{o_2})), \lhd_{v_1}(i_2(t_{i_2}))), \\ \mathcal{C}' &= \{c_2\}, \\ \epsilon'(c_2) &= \epsilon(c_2). \end{aligned}$$

Alternativement, si T_2 *effectue une transition* $T_2 \xrightarrow{o_3(v_1)} \lhd_{v_1}(o_4(t_{o_4}))$, *alors* i_1 *va se synchroniser avec* o_3 *via l'action de communication* c_3. T *va donc effectuer la transition* $T \xrightarrow{c_3} T'$, *où*

$$\begin{aligned} T' &= [\mathcal{C}', \epsilon'](\lhd_{v_1}(o_4(t_{o_4})), \lhd_{v_1}(i_2(t_{i_2}))), \\ \mathcal{C}' &= \{c_3\}, \\ \epsilon'(c_3) &= \epsilon(c_3). \end{aligned}$$

Considérons maintenant les situations de blocage décrites par les règles $\mathbf{Com}_2$, $\mathbf{Com}_3$, et $\mathbf{Com}_4$.

La règle $\mathbf{Com}_2$ stipule que le terme T bloque si un de ses sous-termes T_i exécute une action de blocage $\delta(v)$ et tous les autres sous-termes T_j peuvent attendre jusqu'au temps v. La condition $\text{Wait}(v, T_j)$ est nécessaire pour préserver la monotonie des traces de T.

Les deux dernières règles reflètent le paradigme des contraintes de supposition/engagement et le principe de causalité. Si la communication entre les sous-termes de T est établie lors de l'exécution d'une action interne $oc(v)$ par un des sous-termes T_i, c'est ce sous-terme qui a priorité par rapport à tous les autres sous-termes communicants. Les actions qui communiquent avec oc doivent pouvoir se synchroniser avec toute exécution possible de l'action de sortie oc, sinon le terme composé signale un blocage (règle $\mathbf{Com}_3$). Cependant, on doit s'assurer que tous les termes peuvent attendre jusqu'au temps v pour respecter la monotonie temporelle des traces d'actions.

La synchronisation forcée par l'exécution d'une action de communication ic est plus souple: il suffit qu'un instant quelconque existe où toutes les actions internes qui

communiquent avec ic peuvent être exécutées. Autrement, supposons que l'évolution temporelle du terme T est déjà arrivée au temps v, qui est le dernier temps où ic peut être exécutée. S'il existe au moins une action qui communique avec ic et qui ne peut pas encore être exécutée, la communication ne peut pas être établie. Le terme composé T signale alors un blocage au temps v (règle $\mathbf{Com}_4$). Autrement dit, une action de sortie doit pouvoir se synchroniser en tout temps avec ses actions internes, tandis que pour une action d'entrée il doit y exister au moins un instant où la synchronisation soit possible. Ce type de communication supporte le principe de causalité.

L'opérateur Loop: Dès son démarrage un terme Loop(T) choisit de manière non-déterministe d'exécuter T et ensuite de redémarrer encore une exécution de Loop(T) (règles $\mathbf{B}_1$, $\mathbf{B}_3$) ou bien d'exécuter une seule fois le terme T (règles $\mathbf{B}_2$, $\mathbf{B}_4$). Loop(T) décrit ainsi la composition séquentielle d'un nombre aléatoire $n \geq 1$ d'*instances* du terme T. De plus, Loop(T) bloque si et seulement si le sous-terme T bloque (règle $\mathbf{B}_5$).

Lorsque le terme T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$ et $T' \neq\uparrow$, Loop(T) exécute $a(v)$ et transite soit vers T' (règle $\mathbf{B}_1$), ou bien ou bien vers Seq(T', Loop(T)). De façon similaire, lorsque $T' =\uparrow$, alors Loop(T) peut transiter soit vers $\uparrow$ (règle $\mathbf{B}_3$) ou bien vers $\lhd_v$(Loop(T)) (règle $\mathbf{B}_4$). Donc, les règles $\mathbf{B}_1$ et $\mathbf{B}_3$ décrivent les situations d'arrêt du comportement répétitif, tandis que $\mathbf{B}_2$ et $\mathbf{B}_4$ définissent le comportement récurrent de Loop(T).

Puisque la boucle peut décrire un processus infini il est important d'assurer l'avancement du temps lors de son exécution. Ceci est assuré en utilisant l'opérateur de décalage strict $\lhd$ dans la règle $\mathbf{B}_4$ et en utilisant la composition séquentielle dans la règle $\mathbf{B}_2$. Ainsi, on impose qu'une instance du terme T est démarrée à un temps strictement supérieur au temps de démarrage de l'instance qui le précède.

Le choix non-déterministe: Un terme $T = +(T_1, \ldots, T_m)$ choisit d'exécuter un de ses sous-termes de manière non-déterministe. Le choix est fait indépendamment

de l'urgence des actions des sous-termes: si un sous-terme quelconque T_i peut exécuter une action $a(v)$, alors T peut choisir d'exécuter $a(v)$, même si les autres sous-termes ne peuvent pas attendre jusqu'au temps v (règles Σ_1 et Σ_2). Le terme T peut signaler un blocage $\delta(v)$ si et seulement si au moins un de ses sous-termes T_i bloque au temps v et tous ses autres sous-termes soit bloquent au temps v ou bien ne peuvent pas attendre jusqu'au temps v (règle Σ_3). Dans une telle situation il n'existe plus aucune branche qui peut encore exécuter des actions (différentes de δ) au temps v et le terme composé bloque.

Le choix retardé: Dans la phase de sélection d'un terme $T = \oplus(T_1, \ldots, T_m)$, les branches T_i exécutent, en parallèle, des actions d'entrée communes. Lorsque plusieurs branches T_i peuvent exécuter la même action d'entrée temporisée $ic(v)$ et transitent vers des termes $T_i' \neq \uparrow$, on dit que ces branches ont été *sélectionnées* et la phase de sélection se poursuit avec les successeurs T_i' (règle $\mathbf{CR}_1$). Les branches qui ne peuvent pas exécuter l'action d'entrée spécifiée (règle $\mathbf{CR}_1$) ou bien qui bloquent (règle $\mathbf{CR}_7$), sont éliminées du choix retardé. Si une seule branche T_i peut exécuter une action d'entrée $ic(v)$, et transite ensuite vers un terme $T_i' \neq \uparrow$, on résume la phase de sélection et l'on continue avec l'exécution de T_i' (règle $\mathbf{CR}_1$, $|I| = 1$). On utilise ici les notations 6.2.3, puisque $\oplus(T_i')$ désigne alors le terme T_i'.

Nous imposons une restriction supplémentaire durant la phase de sélection: nous ne permettons pas qu'une ou plusieurs branches influencent leur environnement. Ceci se traduit par deux conditions:

1. nous ne permettons pas des situations où une ou plusieurs branches finissent leur exécution pendant que d'autres branches restent encore actives, et

2. nous ne permettons pas l'exécution d'actions de direction sortie pendant la phase de sélection.

Lorsqu'une de ces restrictions n'est pas satisfaite, un blocage est signalé.

Considérons la première condition d'abord. Dans la règle $\mathbf{CR}_1$ cette restriction

se traduit par l'hypothèse

$$(\forall j \in \{1, \ldots, m\})(T_j \overset{ic(v)}{\not\rightarrow} \uparrow).$$

Dans la règle $\mathbf{CR}_2$ on impose l'exécution de l'action de blocage lorsqu'au moins deux branches T_i et T_j sont sélectionnées, dont une finit son exécution et l'autre non. Pour que T ne signale pas un blocage lorsqu'une de ses branches sélectionnées T_i transite vers $\uparrow$ il faut que, soit toutes les branches sélectionnées finissent également leur exécution, ou bien que T_i soit la seule branche sélectionnée (règle $\mathbf{CR}_3$).

La deuxième restriction, c'est-à-dire l'interdiction d'exécuter des actions de direction sortie pendant la phase de sélection est reflétée dans les règles $\mathbf{CR}_4$ et $\mathbf{CR}_6$. Selon la règle $\mathbf{CR}_4$ un blocage est signalé dès qu'un sous-terme T_i de T exécute une action de sortie oc. Si, par contre, toutes les autres branches T_j bloquent avant que T_i exécute l'action de sortie oc, alors on considère que T_i est la seule branche sélectionnée et aucun blocage n'est signalé (règle $\mathbf{CR}_6$).

Considérons maintenant les situations où une ou plusieurs branches de T bloquent. Si toutes les branches de T exécutent $\delta(v)$, T aussi exécute $\delta(v)$ et finit son exécution dans l'état de blocage $\downarrow$ (règle $\mathbf{CR}_5$).

Si toutes les branches sauf une, T_i, bloquent (règle $\mathbf{CR}_6$), T_i devient la seule branche sélectionnée. Alors, pour toute transition $T_i \overset{a(v')}{\rightarrow} T_i'$ avec $v' \geq v^\delta$, T effectue la transition $T \overset{a(v')}{\rightarrow} T_i'$. Ici, comme dans la règle $\mathbf{CR}_7$, v^δ est défini comme le maximum de tous les temps d'exécution de δ par les sous-termes de T qui bloquent:

$$v^\delta \overset{\text{def}}{=} \max_{i \in \{1,\ldots,m\}} \{v_i \mid T_i \overset{\delta(v_i)}{\rightarrow} \downarrow\}$$

Remarquons ici que si $a = ic$ est une action d'entrée, alors cette situation est également traitée par les règles $\mathbf{CR}_1$ et $\mathbf{CR}_3$.

Si certaines branches bloquent mais qu'au moins deux branches ne bloquent pas, la phase de sélection se poursuit avec le choix retardé des branches restantes, décalées par l'opérateur $\trianglelefteq_{v^\delta}$ (règle $\mathbf{CR}_7$). D'un point de vue conceptuel alors, c'est comme si T effectuait une transition *silencieuse* ou *spontanée* (c'est-à-dire, sans exécuter aucune

action) vers le terme $\bigoplus_{j \in J^\delta} \trianglelefteq_v(T_j)$, où J^δ est défini comme l'ensemble d'indices de toutes les branches qui ne bloquent pas:

$$J^\delta \stackrel{\text{def}}{=} \{j \in \{1, \ldots, m\} \mid \delta \notin \text{exec}(T_j)\}.$$

Puisque notre algèbre ne supporte pas des transitions silencieuses nous devons exprimer ce concept différemment. Dans cette situation on veut que

$$\begin{aligned} \text{exec}(T) &= \text{exec}(\bigoplus_{j \in j} \trianglelefteq_v(T_j)) \\ \text{succ}(a(v'), T) &= \text{succ}(a(v'), \bigoplus_{j \in J} \trianglelefteq_v(T_j)), \end{aligned}$$

pour toute action $a(v') \in \text{exec}(T)$. Pour ce faire nous vérifions toutes les hypothèses $\mathbf{Hyp}_k(\bigoplus_{j \in J^\delta} \trianglelefteq_v(T_j))$ des règles de sémantique opérationnelle qui correspondraient au terme $\bigoplus_{j \in J^\delta} \trianglelefteq_v(T_j)$, avec $1 \leq k \leq 6$. Lorsqu'une de ces hypothèses $\mathbf{Hyp}_k$ est satisfaite nous pouvons appliquer la règle $\mathbf{CR}_k$ et nous dérivons la conclusion $\mathbf{Conc}_k(T)$. Autrement dit, nous concluons que T va exécuter l'action a_k et transiter vers le successeur qui correspond au terme $\bigoplus_{j \in J^\delta} \trianglelefteq_v(T_j)$ selon la règle de $\mathbf{CR}_k$. Comme dans la règle $\mathbf{CR}_6$, on remarque que si $a = ic$, alors cette situation est aussi couverte par les règles $\mathbf{CR}_1, \mathbf{CR}_2$, et $\mathbf{CR}_3$.

L'opérateur d'exception: $\text{Ex}(T_n, T_c, T_e)$ est un terme qui, à priori, se comporte comme le sous-terme T_n, nommé processus *normal*. Lorsque le processus T_c, dit *condition d'exception* finit son exécution avant T_n, on interrompt l'exécution de T_n et on active le processus T_e, nommé *exception* (règle $\mathbf{Ex}_4$). Pour assurer la monotonie du temps, T_e est alors décalé par l'opérateur $\trianglelefteq_v$. Dans le cas contraire, le terme composé est terminé (règle $\mathbf{Ex}_5$). Le processus T_c est un observateur, c'est-à-dire il ne doit contenir que des actions d'entrée. Les règles $\mathbf{Ex}_{2,3}$ décrivent l'exécution en parallèle des processus T_n et T_c. T_n et T_c peuvent donc lire la même action et avancer de manière synchronisée (règle $\mathbf{Ex}_1$).

6.3 Fermeture

Soit $L \subseteq \mathcal{T}(\mathrm{Op})$ un sous-ensemble quelconque du langage de termes clos $\mathcal{T}(\mathrm{Op})$. La *fermeture* de L par rapport à la relation de transition $\rightarrow$, notation $\mathrm{Ferm}(L)$, est définie comme

$$\mathrm{Ferm}(L) = L \cup \{T \in \mathcal{T}(\mathrm{Op}) \mid (\exists T_0 \in L)(T_0 \stackrel{*}{\rightarrow} T)\}.$$

La relation $\rightarrow$ est *fermée* dans L si et seulement si

$$\mathrm{Ferm}(L) = L.$$

On remarque que pour tout $L \subseteq \mathcal{T}(\mathrm{Op})$

$$\mathrm{Ferm}(\mathrm{Ferm}(L)) = \mathrm{Ferm}(L).$$

et que pour tous $L_1, L_2 \subseteq \mathcal{T}(\mathrm{Op})$

$$\mathrm{Ferm}(L_1 \cap L_2) = \mathrm{Ferm}(L_1) \cap \mathrm{Ferm}(L_2), \tag{6.3}$$

$$\mathrm{Ferm}(L_1 \cup L_2) = \mathrm{Ferm}(L_1) \cup \mathrm{Ferm}(L_2). \tag{6.4}$$

Il est trivialement vrai que $\rightarrow$ est fermée dans $\mathcal{T}(\mathrm{Op})$. Par contre, pour tout sous-ensemble strict de $L \subset \mathcal{T}(\mathrm{Op})$ la fermeture est une propriété non triviale.

Dans cette section nous allons prouver que $\rightarrow$ est fermée dans $\mathcal{LA}$.

Dans nos preuves nous utiliserons souvent le principe de l'*induction structurelle*: supposons que nous voulons prouver que tous les termes $T \in L$ ont une certaine propriété $\mathcal{P}$, où $L \subseteq \mathcal{T}(\mathrm{Op})$ est un sous-ensemble quelconque fermé de termes. Formellement ceci s'exprime comme suit:

$$(\forall T \in L)\mathcal{P}(T).$$

On commence alors par prouver la *base d'induction* qui consiste à montrer que des termes élémentaires de L ont la propriété $\mathcal{P}$. Les termes élémentaires sont les termes à partir desquels tous les autres termes de L sont composés donc qui n'ont pas de

sous-termes dans L. Par exemple pour le sous-langage $\mathcal{LA}$ les termes élémentaires sont $a(t_a) \in \mathbf{A} \times T^{\mathbf{A}}$.

On prouve ensuite le *pas d'induction*, qui consiste à montrer que si tous les sous-termes $T_1, \ldots, T_n$ d'un terme quelconque $T = f[T_1, \ldots, T_n] \in L$ avec $n \geq 1$ ont la propriété $\mathcal{P}$, alors T a cette propriété également:

$$(\forall T = f[T_1, \ldots, T_n] \in L)((\forall i \in \{1, \ldots, n\})\mathcal{P}(T_i) \Rightarrow \mathcal{P}(T)).$$

Finalement, pour prouver le théorème de fermeture 6.2 nous avons encore besoin du lemme suivant:

Lemme 6.1 *Soit $T \in \mathcal{LA}$ un terme quelconque qui effectue une transition quelconque $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \in A$ et $v \geq V_0$. Alors*

1. *si $a = \delta$ alors $v = \alpha(T)$ et $T' = \downarrow$;*
2. *si $T \in \mathcal{LA}_\beta$ est un terme feuille alors* $\mathrm{Act}(T') = \mathrm{Act}(T) \setminus \{a\}$;
3. *si $T \in \mathcal{LA}_\mathcal{H}$ est un terme hiérarchique alors* $\mathrm{Act}(T') \subseteq \mathrm{Act}(T)$.

Preuve: la preuve est faite par induction structurelle et est présentée en annexe (section 12.1).

Théorème 6.2 *La relation de transition $\rightarrow$ définie par les règles de* SOp *est fermée dans $\mathcal{LA}, \mathcal{LA}_\beta$, et $\mathcal{LA}^r$. Formellement,*

1. *pour tout terme $T \in \mathcal{LA}$ et toute transition $T \stackrel{a(v)}{\rightarrow} T'$ on a que $T' \in \mathcal{LA}$;*
2. *pour tout terme $T \in \mathcal{LA}_\beta$ et toute transition $T \stackrel{a(v)}{\rightarrow} T'$ on a que $T' \in \mathcal{LA}_\beta$;*
3. *pour tout terme $T \in \mathcal{LA}^r_\beta$ et toute transition $T \stackrel{a(v)}{\rightarrow} T'$ on a que $T' \in \mathcal{LA}^r_\beta$;*
4. *pour tout terme $T \in \mathcal{LA}^r$ et toute transition $T \stackrel{a(v)}{\rightarrow} T'$ on a que $T' \in \mathcal{LA}^r$.*

Preuve: pour prouver que la première affirmation est vraie nous devons montrer que

1. la syntaxe de T' est conforme aux tableaux 5.1 et 5.2 et

2. T' satisfait aux restrictions syntaxiques définies dans la section 5.2.

La preuve du fait que la condition 1 est satisfaite est évidente: il suffit de considérer une par une toutes les règles de SOp et vérifier que le successeur de la transition décrite dans la conclusion de la règle satisfait les restrictions syntaxiques imposées par les grammaires 5.1 et 5.2. Les restrictions syntaxiques supplémentaires définies dans la section 5.2 concernent les ensembles d'actions des opérandes des différents opérateurs de Op. Il suffit alors d'appliquer le lemme 6.1 et la définition de la fonction Act pour vérifier cette deuxième condition. On déduit alors que $\rightarrow$ est fermée dans $\mathcal{LA}$.

Pour prouver que $\rightarrow$ est fermée dans $\mathcal{LA}_\beta$ il suffit de remarquer que $\rightarrow$ est fermée dans $\mathcal{LA}$ et que, selon les règles SOp, aucun successeur d'un terme feuille ne contient des opérateurs hiérarchiques.

Finalement, par définition, un terme quelconque $T \in \mathcal{LA}$ est réactif si et seulement si aucune de ses traces ne contient l'action de blocage δ. Or, les traces de T peuvent être exprimées comme suit:

$$\mathbb{T}(T) = \{a(v).\mu \mid (\exists T' \in \mathcal{LA})(T \stackrel{a(v)}{\rightarrow} T') \ \wedge \ \mu \in \mathbb{T}(T')\}$$

Si T est réactif, forcément tout successeur de T doit aussi être réactif. Puisque nous avons déjà prouvé que $\rightarrow$ est fermée dans $\mathcal{LA}_\beta$ et $\mathcal{LA}$ on peut déduire que $\rightarrow$ est fermée dans $\mathcal{LA}^r_\beta$ et $\mathcal{LA}^r$ aussi. ∎

Donc, nous pouvons définir des systèmes de déduction de termes sur tous ces sous-langages fermés et associer un graphe de processus

$$\text{graphe}(T) = (\text{Ferm}(T), \rightarrow, A, D, T)$$

pour tout terme T d'un de ces sous-langages.

6.4 Déterminisme dans $\mathcal{LA}$

Nous énonçons ici des propriétés de déterminisme des graphes de processus associés aux termes de $\mathcal{LA}$ par les règles de SOp. Par abus de notation, nous disons qu'un langage de spécification algébrique est déterministe si les graphes de processus correspondant aux termes du langage sont déterministes.

Théorème 6.3 *Le sous-langage de termes $T \in \mathcal{LA}$ qui ne contiennent aucun opérateur de choix $+$, $\oplus$, ou* Loop *est déterministe. Notons ce sous-langage par $\mathcal{LA}_{\text{det}}$.*

Preuve: La preuve est triviale: il suffit de remarquer que toutes les règles de sémantique opérationnelle des opérateurs différents de $+$ et $\oplus$ ont des hypothèses mutuellement exclusives. Dans le cas des l'opérateurs de composition parallèle l'exclusivité est assurée par le fait que les ensembles d'actions de termes composés en parallèle sont, par définition, disjoints. Pour les autres opérateurs, l'exclusivité est évidente. ∎

Corollaire 6.4 *Le langage de termes de feuilles $\mathcal{LA}_\beta$ est déterministe.*

Ceci conclut ce chapitre dans lequel nous avons défini la sémantique opérationnelle du langage de termes $\mathcal{LA}$. Dans le chapitre suivant nous définirons le sous-langage de termes accessibles $\mathcal{L} \subset \mathcal{LA}$ qui est à la base de la définition de l'algèbre de processus ACTC.

Chapitre 7

Le sous-langage $\mathcal{L}$

Dans ce chapitre nous définissons le sous-langage de termes *accessibles* $\mathcal{L} \subset \mathcal{LA}$ et prouvons des propriétés syntaxiques et sémantiques importantes de ce sous-langage. Les résultats théoriques présentés dans ce document concernent le sous-langage $\mathcal{L}$ uniquement: l'algèbre ACTC est définie sur $\mathcal{L}$. Ceci ne réduit pas l'expressivité sémantique de l'algèbre. En effet, tout chronogramme peut être spécifié par un terme de $\mathcal{L}$.

Nous commençons par prouver que la relation de transition définie par les règles SOp est fermée dans $\mathcal{L}$ (théorème 7.1). Ce résultat est essentiel pour qu'on puisse définir un SDT et/ou algèbre de processus sur le langage $\mathcal{L}$.

Ensuite, nous montrons que la fonction α est bien définie sur $\mathcal{L}$ (proposition 7.3) et nous utilisons ce résultat pour prouver la monotonie temporelle de toute trace de tout terme de $\mathcal{L}$ (théorème 7.8).

Finalement, nous prouvons des propriétés sémantiques de la fonction Γ ainsi que des propriétés de termes réactifs de $\mathcal{L}$. Ces résultats sont utilisés dans la preuve de la décidabilité de l'équivalence de termes $\approx$, qui sera définie dans le chapitre suivant.

7.1 Définitions et notations

Commençons par donner la définition du langage de termes *accessibles* $\mathcal{L} \subset \mathcal{LA}$:

Définition 7.1 *Un terme $T \in \mathcal{LA}$ est un terme de $\mathcal{L}$ si et seulement si ou bien $T \in \mathcal{LA}_0$ ou bien il existe un terme initial $T_0 \in \mathcal{LA}_0$ tel que*

$$T_0 \xrightarrow{*} T.$$

$\mathcal{L}$ contient tous les termes initiaux de $\mathcal{LA}$ ainsi que tous les termes $T \in \mathcal{LA}$ qui sont accessibles à partir d'au moins un terme initial par une séquence arbitraire de transitions

$$\mathcal{L} \stackrel{\text{def}}{=} \mathcal{LA}_0 \cup \{T \in \mathcal{LA} \mid (\exists T_0 \in \mathcal{LA}_0)(T_0 \xrightarrow{*} T)\}.$$

(la notation $\xrightarrow{*}$ est telle que définie dans la section 4.1).

Nous notons par $\mathcal{L}_\beta, \mathcal{L}_\mathcal{H}, \mathcal{L}^r$, et $\mathcal{L}^r_\beta$, respectivement, les ensembles de termes suivants:

- $\mathcal{L}_\beta \stackrel{\text{def}}{=} \mathcal{L} \cap \mathcal{LA}_\beta$;
- $\mathcal{L}_\mathcal{H} \stackrel{\text{def}}{=} \mathcal{L} \cap \mathcal{LA}_\mathcal{H}$;
- $\mathcal{L}^r \stackrel{\text{def}}{=} \mathcal{L} \cap \mathcal{LA}^r$;
- $\mathcal{L}^r_\beta \stackrel{\text{def}}{=} \mathcal{L}_\beta \cap \mathcal{L}^r$;
- $\mathcal{L}_0 \stackrel{\text{not.}}{=} \mathcal{LA}_0$.

Les résultats théoriques que nous présentons dans ce document concernent uniquement le sous-langage $\mathcal{L}$ de termes accessibles: l'algèbre de processus ACTC et son axiomatisation sont définies sur $\mathcal{L}$. Cette restriction n'a pas d'impact sur l'applicabilité pratique de l'algèbre: en fait, les chronogrammes peuvent être spécifiés uniquement à partir de $\mathcal{L}$. Nous verrons dans la prochaine section que les seules restrictions syntaxiques qui distinguent $\mathcal{L}$ de $\mathcal{LA}$ sont les suivantes:

1. si $T = f[T_1, \ldots, T_n] \in \mathcal{L}$ alors tous les sous-termes actifs de T ont le même temps de démarrage, qui est aussi le temps de démarrage de T;

2. si $T = \trianglelefteq_v(S) \in \mathcal{L}$ ou $T = \triangleleft_v(S) \in \mathcal{L}$ alors $\alpha(S) \leq \alpha(T)$.

Ces restrictions correspondent, en fait, au sens que le temps de démarrage a dans la sémantique de tout système composé: lorsqu'un système composé démarre son exécution c'est naturel de considérer que ses composants aussi sont démarrés. La deuxième propriété assure la monotonie temporelle des traces de termes de $\mathcal{L}$.

Prouvons maintenant que la relation de transition $\rightarrow$ définie par les règles de SOp est fermée dans $\mathcal{L}$.

Théorème 7.1 *La relation de transition de termes $\rightarrow$ définie par les règles de* SOp *est fermée dans les sous-langages $\mathcal{L}$, $\mathcal{L}_\beta$, et $\mathcal{L}^r$.*

Preuve: Soit $T \in \mathcal{L}$ un terme quelconque de $\mathcal{L}$ et $T \stackrel{a(v)}{\rightarrow} T'$ une transition quelconque telle que définie par les règles de SOp. Alors, par définition, il existe un terme initial $T_0 \in \mathcal{LA}_0$ et une trace $\mu = a_1(v_1) \ldots a_n(v_n)$ d'actions temporisées $a_i(v_i) \in A \times D$ tels que

$$T_0 \stackrel{\mu}{\rightarrow} T.$$

Mais alors la concaténation $\mu.a(v)$ est une trace telle que

$$T_0 \stackrel{\mu.a(v)}{\rightarrow} T'.$$

Donc, selon la définition de $\mathcal{L}$, $T' \in \mathcal{L}$, et on déduit que $\mathrm{Ferm}(\mathcal{L}) = \mathcal{L}$.

Selon l'égalité 6.3 et puisque nous avons déjà prouvé que $\rightarrow$ est fermée dans $\mathcal{LA}_\beta$ et $\mathcal{LA}^r$ (théorème 6.2) on peut déduire que $\rightarrow$ est également fermée dans $\mathcal{L}_\beta, \mathcal{L}^r$, et $\mathcal{L}^r_\beta$. ∎

7.2 Propriétés sémantiques et syntaxiques de α

Nous commençons par montrer la corrélation qui existe entre le temps de démarrage $\alpha(T)$ d'un terme $T = f[T_1, \ldots, T_n] \in \mathcal{L}$ et le temps de démarrage $\alpha(T_i)$ de ses sous-termes.

Lemme 7.2 *Pour toute transition*

$$T \stackrel{a(v)}{\rightarrow} T' = f[T_1, \ldots, T_n]$$

avec $T, T' \in \mathcal{L}$, une des quatre situations suivantes s'applique: soit

1. *$T' \in \{\uparrow, \downarrow\}$ ou bien*
2. *$T' = \trianglelefteq_v(S)$, pour un terme quelconque $S \in \mathcal{L} \setminus \{\uparrow, \downarrow\}$ ou encore*
3. *$T' = \triangleleft_v(S)$, pour un terme quelconque $S \in \mathcal{L} \setminus \{\uparrow, \downarrow\}$ ou*
4. *$T' = f[T_1, \ldots, T_n]$ avec $f \in \mathrm{Op} \setminus \{\trianglelefteq, \triangleleft\}$ et $\alpha(T') = \alpha(T_i) = v$, pour tout sous-terme actif T_i de T'.*

Preuve: il suffit de considérer toutes les règles de SOp et vérifier que les successeurs des transitions décrites ont la propriété énoncée dans le lemme. ∎

Illustrons cette propriété des termes de $\mathcal{L}$ par deux exemples.

Exemple 7.1 *Soit $T = a(t_a).T_1 || T_2 \in \mathcal{L}_\beta$ un terme feuille. Le sous-terme $a(t_a).T_1$ peut effectuer la transition:*

$$a(t_a).T_1 \stackrel{a(v)}{\rightarrow} \triangleleft_v(T_1),$$

pour tout $v \geq V_0$. Supposons d'abord que T_2 n'a pas d'actions qui doivent être exécutées avant le temps v. Le terme T peut alors effectuer la transition

$$T \stackrel{a(v)}{\rightarrow} \triangleleft_v(T_1) || \trianglelefteq_v(T_2) \stackrel{\text{not.}}{=} T'.$$

Par définition alors

$$\begin{aligned} \alpha(T') &\stackrel{\text{def}}{=} \min(\alpha(\lhd_v(T_1)), \alpha(\trianglelefteq_v(T_2)) && (7.1)\\ &= \min(v, v) && (7.2)\\ &= v. && (7.3) \end{aligned}$$

Donc, le temps de démarrage de T' et de ses deux sous-termes est égale à v.

Supposons maintenant qu'il existe une action $b \in \mathrm{Act}(T_2)$ dont l'exécution est urgente avant le temps v. Par exemple, soit $v = 5$ et

$$T_2 = t_b \leq 3 \ \wedge t_c > 1 : b(t_b) \| c(t_c).$$

Donc, il existe une transition

$$T_2 \stackrel{b(v')}{\rightarrow} t_c > 1 : \trianglelefteq_{v'}(c(t_c),$$

avec $v' \leq 3 < v$ et T_2 ne peut exécuter l'action b à aucun instant supérieur ou égal à v. Puisque T_1 peut exécuter l'action a au temps v, le prédicat $\mathrm{Wait}(v', T_1)$ *est vrai et T effectue la transition suivante:*

$$T \stackrel{b(v')}{\rightarrow} T' = \trianglelefteq_{v'}(T_1) \parallel t_c > 1 : \trianglelefteq_{v'}(c(t_c)).$$

Donc, par définition, le temps de démarrage de T' et de ses sous-termes est égal à v'.

Exemple 7.2 *Soit $T = a_1(t_{a_1}).a_2(t_{a_2}) || S \in \mathcal{L}_\beta$ un terme feuille. Selon les règles de sémantique opérationnelle* SOp, *T peut effectuer la séquence suivante de transitions*

$$T \stackrel{a_1(v_1)}{\rightarrow} T' \stackrel{a_2(v_2)}{\rightarrow} T'',$$

où

$$\begin{aligned} T' &= \lhd_{v_1}(a_2(t_{a_2})) || \trianglelefteq_{v_1}(T_2),\\ T'' &= \trianglelefteq_{v_2}(\trianglelefteq_{v_1}(S)) \end{aligned}$$

en autant que $v_2 \geq v_1$ *et que les sous-termes* T_2 *et* $\trianglelefteq_{v_1}(T_2)$ *n'aient pas des actions plus urgentes à exécuter lors de leurs transitions respectives. Donc,*

$$\alpha(T') \stackrel{\text{def}}{=} \min(\alpha(\triangleleft_{v_1}(a_2(t_{a_2}))), \alpha(\trianglelefteq_{v_1}(T_2)), \tag{7.4}$$

$$= \min(v_1, v_1) \tag{7.5}$$

$$= v_1. \tag{7.6}$$

et $\alpha(T'') \stackrel{\text{def}}{=} v_2$.

La proposition suivante est un corollaire immédiat du lemme 7.2:

Proposition 7.3 *Pour tous* $T, T' \in \mathcal{L} \setminus \{\uparrow, \downarrow\}, a \in A$ *et* $v \in V$ *on a que*

$$T \stackrel{a(v)}{\rightarrow} T' \Rightarrow \alpha(T') = v.$$

En ce qui concerne les termes initiaux nous pouvons énoncer les propositions suivantes:

Proposition 7.4 *Si* $T \in \mathcal{L}_0$ *est un terme initial, alors il n'existe aucune transition* $T_0 \stackrel{a(v)}{\rightarrow} T$ *qui mène à* T. *Formellement,*

$$T \in \mathcal{L}_0 \subset \mathcal{L} \Rightarrow (\forall T_0 \in \mathcal{L}, a \in A, v \in D)(T_0 \stackrel{a(v)}{\not\rightarrow} T).$$

Preuve: La preuve de l'implication est basée sur la remarque suivante: selon les règles de SOp, pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ avec $S, S' \in \mathcal{L}$ et $a(v) \in A \times D$ le terme successeur S' contient au moins un opérateur $\trianglelefteq_v$ ou $\triangleleft_v$. Donc, puisque

$$\mathcal{L}_0 \stackrel{\text{def}}{=} \{S \in \mathcal{LA} \mid S \text{ ne contient aucun opérateur de décalage}\},$$

aucun terme qui est le successeur d'un autre terme ne peut être un terme initial. ∎

Proposition 7.5 *Si* $T = f[T_1, \ldots, T_n] \in \mathcal{L}_0$ *est un terme initial quelconque alors*

$$\alpha(T) = \alpha(T_1) = \ldots = \alpha(T_n) = V_0.$$

Preuve: la proposition suit directement du fait que, par définition, tout sous-terme d'un terme initial $T \in \mathcal{L}_0$ est aussi un terme initial de $\mathcal{L}_0$ et de la défintion de α. ∎

7.3 Monotonie temporelle

Dans la définition de tout système de transitions temporisé il est essentiel d'assurer la monotonie temporelle de toute trace d'actions. Dans notre cas il s'agit de prouver que pour toute séquence de transitions

$$T_1 \overset{a_1(v_1)}{\rightarrow} T_2 \overset{a_2(v_2)}{\rightarrow} T_3,$$

les temps $v_1, v_2 \in D$ sont tels que

$$v_1 \leq v_2.$$

Assurer et/ou prouver cette propriété est un problème non trivial, particulièrement dans le cas d'une algèbre de processus temporisée où la relation de transition $\rightarrow$ est définie uniquement à partir de considérations syntaxiques par un ensemble de règles de sémantique opérationnelle.

Dans cette section nous prouvons la monotonie de traces de tout terme $T \in \mathcal{L}$.

Avant d'aborder le théorème de la monotonie, nous avons encore besoin du lemme 7.6 qui affirme que pour tout terme $T \in \mathcal{L}$, si T est composé à partir des sous-termes quelconques $T_1, \ldots, T_n$, alors T ne peut exécuter une action quelconque $a(v)$ que si un de ses sous-termes exécute une action quelconque $b(v)$ (égale ou différente de a), ou bien si $a = \delta$ et $v = \alpha(T)$.

Lemme 7.6 *Pour tous* $T, T_1, \ldots, T_n \in \mathcal{L}$, $f \in \mathrm{Op}$, $a(v) \in A \times D$, *si* $T = f[T_1, \ldots, T_n]$ *et si* $a(v) \in \mathrm{exec}(T)$, *alors soit*

1. $a = \delta$ *et* $v = \alpha(T)$, *ou bien*

2. $\exists b(v) \in \mathrm{exec}(T_i)$, *où* $b(v) \in A \times D$ *et* $1 \leq i \leq n$.

Preuve: il suffit de considérer une par une toutes les règles de SOp et vérifier que la condition est satisfaite. ∎

Basée sur la définition de α et le lemme précédent, la proposition suivante affirme qu'un terme ne peut exécuter aucune action avant son temps de démarrage:

Proposition 7.7 *Pour tous $T, T' \in \mathcal{L}, a \in A$ et $v \in D$,*

$$T \stackrel{a(v)}{\rightarrow} T' \Rightarrow v \geq \alpha(T).$$

Preuve: la proposition sera prouvée par induction structurelle. On remarque d'abord que les termes $a(t_a)$ satisfont la proposition de manière triviale. Prouvons le pas d'induction. Soit

$$T = f[T_1, \ldots, T_n] \in \mathcal{L}$$

un terme quelconque de $\mathcal{L}$ qui effectue une transition quelconque

$$T \stackrel{a(v)}{\rightarrow} T'.$$

Si $a = \delta$ alors il suit directement du lemme 7.6 que $v = \alpha(T)$. Si $a \neq \delta$ on déduit selon le même lemme 7.6 qu'il doit exister une transition

$$T_i \stackrel{b(v)}{\rightarrow} T_i'$$

d'un des sous-termes actifs de T. On applique alors l'hypohtèse d'induction et on déduit que

$$\alpha(T_i) \leq v.$$

Or, par définition

$$\alpha(T) = \min_{T_j \text{ sous-terme actif}} \alpha(T_j) \leq \alpha(T_i).$$

Donc, $\alpha(T) \leq v$, ce qu'il fallait prouver. ∎

Nous sommes maintenant en mesure d'énoncer et prouver le théorème de la monotonie temporelle (aussi appelé théorème de l'avancement du temps):

Théorème 7.8 [Théorème de la monotonie temporelle] *Pour tous $T_1, T_2, T_2 \in \mathcal{L}, a_1, a_2 \in A$ et $v_1, v_2 \in D$ si*

$$T_1 \stackrel{a_1(v_1)}{\rightarrow} T_2 \stackrel{a_2(v_2)}{\rightarrow} T_3$$

alors $v_1 \leq v_2$.

Preuve: Le théorème est une conséquence directe de la proposition 7.3 et de la proposition 7.7. ∎

7.4 Propriétés sémantiques de Γ

Dans la section 6.2.1.2 nous avons défini la fonction Γ, qui associe à chaque terme $T \in \mathcal{L}$ une contrainte temporelle $\Gamma(T) \in \Theta$. La description intuitive de la fonction Γ est qu'elle retourne l'information temporelle contenue dans un terme. Dans cette section nous allons donner une interprétation sémantique de cette définition, en prouvant un nombre de propriétés que la fonction Γ satisfait.

Nous commençons par la proposition suivante qui exprime une condition nécessaire pour qu'un terme feuille quelconque $T \in \mathcal{L}_\beta$ puisse exécuter une action $a(v)$ quelconque:

Proposition 7.9 *Pour tous $T, T' \in \mathcal{L}$, $a \in A \setminus \{\delta\}$, et $v \in D$,*

$$T \xrightarrow{a(v)} T' \Rightarrow \text{First}(a(v), T).$$

Preuve: présentée en annexe, section 12.2.1:

Un corollaire de cette proposition est la propriété suivante de tout terme $T \in \mathcal{L}$:

Proposition 7.10 *Soit $T \in \mathcal{L}$ un terme quelconque et $a(v) \in \text{exec}(T)$ une action temporisée différente de δ que T peut exécuter. Alors*

$$(\forall a(v) \in \text{exec}^\delta(T))(\text{Sol}(t_a = v) \cap \Gamma(T) \neq \emptyset).$$

Une conséquence immédiate de cette proposition est le fait que si $\Gamma(T) = \emptyset$ pour un terme quelconque $T \in \mathcal{L}$, alors T se trouve dans une situation de blocage, c'est-à-dire que T ne peut exécuter aucune action de $A \setminus \{\delta\}$. Dans la proposition suivante nous prouvons que la sémantique opérationnelle de ACTC et telle que tout terme $T \neq \uparrow, \downarrow$ peut exécuter au moins une action. Un corollaire de cette propriété est que si un terme T quelconque ne peut exécuter aucune action de $A \setminus \{\delta\}$, T doit exécuter l'action de blocage δ.

Proposition 7.11 *Soit $T \in \mathcal{L} \setminus \{\uparrow, \downarrow\}$ un terme quelconque. Alors* $\text{exec}(T) \neq \emptyset$.

Preuve: La preuve se fait par induction structurelle et sera présentée en annexe, section 12.2.2.

Corollaire 7.12 *Soit $T \in \mathcal{L}$ un terme quelconque. Alors*

$$\text{exec}(T) \setminus \{\delta\} = \emptyset \text{ implique } \text{exec}(T) = \{\delta\}.$$

La proposition suivante est une conséquence immédiate du corollaire 7.12 et de la proposition 7.10:

Proposition 7.13 *Pour tout terme $T \in \mathcal{L}$ si $\Gamma(T) = \emptyset$ alors* $\text{exec}(T) = \delta$.

7.5 Propriétés de base de termes réactifs

Rappelons que, selon le lemme 6.1, lorsqu'un terme feuille $T \in \mathcal{L}_\beta$ exécute une action $a \neq \delta$ son successeur T' contient toutes les actions de T, sauf a. Implicitement, pour qu'un terme feuille réactif transite vers l'état inactif $\uparrow$ il doit avoir exécuté toutes ses actions. Notons que les termes composés avec l'opérateur $+$, par exemple, n'ont pas cette propriété.

Nous généralisons la propriété énoncée dans la proposition précédente, concernant l'exécution d'actions, aux exécutions de traces d'actions. Nous énonçons ainsi une propriété importante de tout terme feuille réactif:

Proposition 7.14 *Soit $T \in \mathcal{L}_\beta$ un terme réactif. Alors, il existe une trace de T, $\mu \in \mathbb{T}(T)$, qui contient toutes les actions de T. Formellement,*

$$(\forall T \in \mathcal{L}_\beta)(T \text{ réactif } \Rightarrow (\exists \mu \in \mathbb{T}(T))(\text{Act}(\mu) = \text{Act}(T))).$$

Preuve: Puisque T est réactif, aucune de ses traces ne mène vers un blocage, autrement dit aucune de ses traces ne contient l'action de blocage δ. Évidemment alors, tout successeur d'un terme réactif est un terme réactif. Selon la proposition 7.11, tout terme qui ne bloque pas et qui est différent de $\uparrow$ ou $\downarrow$ peut exécuter au

moins une action $a \in \mathrm{Act}(T)$. Donc, puisque tout terme T de $\mathcal{L}_\beta$ a un nombre fini d'actions alors il doit exister une trace $\mu = a_0(v_0) \ldots a_n(v_n) \in \mathbb{T}(T)$ qui mène vers l'état d'acceptation $\uparrow$. Formellement,

$$T \overset{a_0(v_0)}{\rightarrow} T_1 \ldots a_i(v_i) \rightarrow T_{i+1} \ldots a_n(v_n) \rightarrow \uparrow .$$

Or, selon la Proposition 6.1, puisque $\mathrm{Act}(\uparrow) = \emptyset$,

$$\begin{aligned} \mathrm{Act}(T) &= \{a_0\} \cup \mathrm{Act}(T_0) = \ldots \\ &= \{a_0, \ldots, a_i\} \cup \mathrm{Act}(T_i) = \ldots \\ &= \{a_0, \ldots, a_n\}, \end{aligned}$$

ce qu'il fallait prouver. ∎

Nous avons maintenant défini un sous-langage de termes $\mathcal{L} \subset \mathcal{LA} \subset \mathcal{T}(\mathrm{Op})$ avec une syntaxe et une sémantique précises. Pour pouvoir définir l'algèbre de processus il reste à introduire, dans le chapitre suivant, une relation d'équivalence sémantique de termes, notée $\approx \subseteq \mathcal{L} \times \mathcal{L}$ qui est aussi une congruence sur $\mathcal{L}$.

Chapitre 8

La relation de bisimulation $\approx_{\mathrm{ACTC}}$

Dans les chapitres précédents nous avons défini la syntaxe et la sémantique opérationnelle du langage termes de $\mathcal{L}$ pour la spécification de chronogrammes. Pour pouvoir définir l'algèbre de processus ACTC à partir du langage algébrique $\mathcal{L}$ nous devons encore définir une relation d'équivalence sémantique de termes qui soit une congruence pour l'ensemble d'opérateurs de $\mathcal{F}$. Nous appelons cette relation la *bisimulation–ACTC* et nous la notons par $\approx_{\mathrm{ACTC}}$. Quand aucune confusion n'est possible nous simplifions cette notation et désignons la bisimulation–ACTC simplement par $\approx$. La relation $\approx \subseteq \mathcal{L} \times \mathcal{L}$ est une bisimulation forte avec deux restrictions syntaxiques supplémentaires, introduites pour assurer la congruence de $\approx$ par rapport aux opérateurs de Op. Nous prouverons que $\approx$ est une relation d'équivalence et expliquons le raisonnement général à suivre pour prouver que deux termes quelconques $T_1, T_2 \in \mathcal{L}$ sont équivalents selon la relation $\approx$. Finalement, nous prouverons dans la section 8.10 que $\approx$ est une congruence par rapport à tous les opérateurs de composition de Op.

8.1 Définitions et notations

Rappelons que dans la section 4.2 nous avons donné la définition de la relation d'équivalence de termes appelée bisimulation forte ([62]) et notée par $\simeq$. La relation $\simeq$ a été alors présentée comme une relation d'équivalence de STEI $= (A, \mathbf{S}, \rightarrow, i)$, c'est-à-dire de systèmes à transitions dont l'ensemble d'états est $\mathbf{S}$, la relation de transition d'états $\rightarrow \subseteq \mathbf{S} \times A \times \mathbf{S}$ est étiquetée par des actions de l'ensemble d'actions A, et dont l'état initial est $i \in \mathbf{S}$. Rappelons aussi que nous identifions, par abus de langage, un état i à son STEI associé, lorsque le système à transitions sous-jacent STE $= (A, \mathbf{S}, \rightarrow)$ est sous-entendu.

Ensuite, dans la section 4.6.1 nous avons introduit la notion de STEI temporisé $(A, \mathbf{S}, \rightarrow, i, D)$, où D est un domaine temporel et la relation de transition est étiquetée par des actions temporisées $(a, v) \in A \times D$, notées $a(v)$. Nous avons également remarqué qu'un STE temporisé est en même temps un STE simple: il suffit de considérer que l'ensemble d'actions est le produit cartésien $A \times D$. Toutes les définitions concernant les STE et STEI simples s'appliquent alors implicitement aussi aux STE et STEI temporisés. Donc, la définition de la bisimulation forte s'applique également aux STEI temporisés.

En donnant une sémantique opérationnelle aux termes du langage $\mathcal{LA}$, nous associons à tout terme $T \in \mathcal{L} \subset \mathcal{LA}$ un STEI temporisé $(A, \mathcal{L}, \rightarrow, T, D)$. La relation de bisimulation forte de termes de $\mathcal{L}$, notée $\simeq \subseteq \mathcal{L} \times \mathcal{L}$, est alors définie comme la plus grande relation de termes telles que pour toute paire $(T, S) \in \simeq$

1. pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$ telle que $(T', S') \in \simeq$ et

2. pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $(T', S') \in \simeq$.

Avant de définir la relation $\approx$ rappelons encore la définition de la relation d'équivalence

de traces $\sim \subseteq \mathcal{L} \times \mathcal{L}$ donnée dans la section 4.1:

$$T \sim S \overset{\text{def}}{\Leftrightarrow} \mathbb{T}(T) = \mathbb{T}(S).$$

Puisque les graphes de processus associés aux termes de $\mathcal{L}$ sont temporisés, l'ensemble de traces $\mathbb{T}(T)$ pour un terme quelconque $T \in \mathcal{T}$ est défini comme:

$$\mathbb{T}(T) \overset{\text{def}}{=} \{\mu = a_0(v_0).a_1(v_1)... \in (A \times D)^\infty \mid$$
$$(\exists\, S_0 S_1... \in \mathcal{L}^\infty)\ (\forall i \geq 0)(s_i \overset{a_{i+1}(v_{i+1})}{\rightarrow} s_{i+1}) \wedge (T \overset{a_0(v_0)}{\rightarrow} S_0)\}.$$

Donnons maintenant la définition de la notion de *bisimulation*–ACTC.

Définition 8.1 *Une relation de $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ est une bisimulation*–ACTC *si et seulement si*

1. *$\mathcal{R}$ est une bisimulation forte, autrement dit $\mathcal{R} \subseteq \simeq$,*
2. *$\forall (S_1, S_2) \in \mathcal{R}$ on a que $\alpha(S_1) = \alpha(S_2)$, et*
3. *$\forall (S_1, S_2) \in \mathcal{R}$ on a que $\Gamma(S_1) = \Gamma(S_2)$.*

Autrement dit une bisimulation–ACTC contient des paires de termes qui sont fortement bisimilaires, qui démarrent en même temps et qui ont le même espace de solutions des contraintes descriptives. Ces deux propriétés supplémentaires sont nécessaires pour assurer la congruence de $\approx$ par rapport aux opérateurs de Op.

La relation $\approx \subseteq \mathcal{L} \times \mathcal{L}$ est définie comme la plus grande *bisimulation–ACTC*:

Définition 8.2 *La relation $\approx \subseteq \mathcal{L} \times \mathcal{L}$ est l'union de toutes les bisimulations*–ACTC*:*

$$\approx \;=\; \bigcup_{\mathcal{R}=\text{bisimulation}-\text{ACTC}} \mathcal{R}.$$

L'exemple suivant montre que la condition 2 de la définition 8.1 d'une bisimulation–ACTC est nécessaire pour que $\approx$ soit une congruence par rapport à l'opérateur θ :

Exemple 8.1 *Soient* $T = \trianglelefteq_2(a(5))$ *et* $S = \trianglelefteq_3(a(5))$ *deux termes de* $\mathcal{L}$. *Comme dans l'exemple 8.3 nous utilisons la notation*

$$a(v) \stackrel{\text{not.}}{=} t_a = v : a(t_a)$$

pour sa simplicité, où $v \geq 0$ *est une constante de* D *et* $a \in A_\beta \setminus \{\delta\}$. *Soit* $\theta = (t_a = 4)$ *une contrainte de* Θ. *Nous remarquons que (1)* $T \simeq S$ *et que (2)* $\alpha(T) \neq \alpha(S)$. *Ensuite, puisque* $\Gamma(\theta : T) = \Gamma(\theta : S)$ *on applique la règle* $\mathbf{supp}_4$ *et on déduit que* T *et* S *se trouvent dans une situation de blocage.* T *exécute alors l'action* δ *au temps* $\alpha(T) = 2$ *et* S *exécute* δ *au temps* $\alpha(S) = 3$. *Évidemment, alors,* $\theta : T \not\approx \theta : S$.

Montrons maintenant dans l'exemple qui suit que la condition 3 de la définition 8.1 est nécessaire pour assurer la congruence de $\approx$ par rapport aux opérateurs de Op.

Exemple 8.2 *Soient* $T, S \in \mathcal{L}_\beta$ *les termes suivants:*

$$\begin{aligned} T &= \text{Max}(b, \{a\}, [2,3],\ a(t_a) \| b(t_b)), \\ S &= 2 \leq t_b - t_a \leq 3\ :\ a(t_a) \| b(t_b)). \end{aligned}$$

On remarque que T *et* S *sont fortement bisimilaires, donc* $T \simeq S$. *De plus ce sont des termes initiaux, donc* $\alpha(T) = \alpha(S) = V_0$. *Mais* $\Gamma(T) \neq \Gamma(S)$, *puisque*

$$\begin{aligned} \Gamma(T) &= \text{Sol}(t_a \geq V_0 \ \wedge\ t_b \geq V_0), \\ \Gamma(S) &= \text{Sol}(t_a \geq V_0 \ \wedge\ t_b \geq V_0 \ \wedge\ 2 \leq t_b - t_a \leq 3). \end{aligned}$$

Soit $\theta \in \Theta$ *la contrainte* $t_b = t_a + 5$. *Alors,* $\theta : T$ *peut effectuer la séquence de transitions suivante:*

$$\theta : T \stackrel{a(v)}{\rightarrow} (2 \leq t_b - v \leq 3 \ \wedge\ t_b = v + 5) : \trianglelefteq_v(b(t_b)) \stackrel{\delta(v)}{\rightarrow}\downarrow,$$

pour tout $v \geq V_0$. *Le terme* $\theta : S$, *par contre, ne peut effectuer que la transition suivante:*

$$\theta : S \stackrel{\delta(V_0)}{\rightarrow}\downarrow .$$

Donc, clairement $\theta : T \not\approx \theta : S$.

La congruence de $\approx$ sur le langage $\mathcal{L}$ sera prouvée dans le théorème 8.10.

Les inclusions suivantes sont évidentes:

$$\approx \subseteq \simeq \subseteq \sim . \tag{8.1}$$

Autrement dit, pour tous $T, S \in \mathcal{L}$:

$$T \approx S \Rightarrow T \simeq S \Rightarrow T \sim S \tag{8.2}$$

Théorème 8.1 $\approx$ *existe, elle est une bisimulation–*ACTC *et une relation d'équivalence de termes de l'algèbre* ACTC.

Preuve: $\approx$ est définie comme une union infinie d'ensembles. Il suffit alors que cette union soit bornée supérieurement pour qu'elle existe. Or, $\approx \subseteq \simeq$. Donc, $\approx$ est bien définie.

Montrons maintenant que $\approx$ est une bisimulation–ACTC. Soit (T, S) une paire quelconque de $\approx_{\text{ACTC}}$. Alors il existe une relation $\mathcal{R}$ telle que $(T, S) \in \mathcal{R} \subseteq \approx$ et telle que $\mathcal{R}$ soit une bisimulation–ACTC. La paire (T, S) satisfait donc les trois propriétés énumérées dans la définition 8.1. On déduit que $\approx$ est une bisimulation–ACTC.

Il reste à prouver que $\approx$ est une relation d'équivalence. Pour ceci nous devons montrer qu'elle est (1) commutative, (2) transitive, et (3) réflexive. Les deux premières propriétés suivent directement de la définition de $\approx$. La réflexivité suit du fait que $\approx$ est un sous-ensemble de $\simeq$, qui est elle-même une relation d'équivalence. ∎

Dans la sous-section suivante nous présentons le raisonnement général à suivre pour montrer que deux termes quelconques sont équivalents selon la relation $\approx$. Des exemples seront donnés pour faciliter la compréhension du contenu de cette section.

8.2 Preuves d'équivalence – Exemples

Pour prouver que deux termes quelconques $S, T \in \mathcal{L}$ sont équivalents, il suffit de trouver une relation binaire de termes $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$, telle que $(S, T) \in \mathcal{R}$ et telle que

$\mathcal{R}$ soit une bisimulation–ACTC. Puisque $\approx$ est définie comme l'union de toutes les bisimulations–ACTC, on peut alors conclure que

$$(S, T) \in \mathcal{R} \subseteq \approx ,$$

donc, que S et T sont équivalents.

Une fois la relation $\mathcal{R}$ définie, on doit s'assurer que

1. $\mathcal{R}$ est une bisimulation forte et que
2. pour toute paire de termes $(S, T) \in \mathcal{R}$
   (a) $\alpha(S) = \alpha(T)$ et
   (b) $\Gamma(S) = \Gamma(T)$.

Montrer que $\mathcal{R}$ est une bisimulation forte revient à prouver que S et T peuvent exécuter les mêmes actions aux mêmes instants et que $\mathcal{R}$ est fermée à gauche est à droite aux transitions (définition 4.2). Si la relation $\mathcal{R}$ est symétrique par définition, il suffit de montrer qu'elle est unilatéralement fermée aux transitions, c'est-à-dire qu'elle a une seule des deux propriétés énumérées dans la définition d'une bisimulation (4.2).

Pour démontrer que S et T peuvent exécuter les mêmes actions temporisées, c'est-à-dire que $\text{exec}(S) = \text{exec}(T)$, nous devons considérer, une par une, toutes les règles de transitions définies dans la sémantique opérationnelle SOp qui concernent les termes S et T.

En général, pour pouvoir prouver que $\mathcal{R}$ est fermée aux transitions, nous définissons $\mathcal{R}$ comme

$$\mathcal{R} = \mathcal{R}_{\text{spec}} \cup \approx .$$

$\mathcal{R}_{\text{spec}}$ est une relation spécifiquement définie pour contenir la paire (S, T). Nous incluons $\approx$ dans $\mathcal{R}$ pour pouvoir utiliser toutes les propriétés de $\approx$ déjà connues, telles que le fait que toute paire de termes identiques (T, T) est dans $\approx$, donc dans $\mathcal{R}$.

De plus, puisque $\approx$ est une bisimulation–ACTC, toutes les paires $(S,T) \in \approx \subseteq \mathcal{R}$ satisfont trivialement déjà aux conditions (1),(2), et (3) de la définition 8.1. Il suffit donc de considérer seulement les paires $(S,T) \in \mathcal{R}_{\text{spec}}$.

Par exemple, pour prouver que $\approx$ est une congruence par rapport à l'opérateur de préfixage d'actions (voir annexe, section 12.3.5.2), nous définissons les relations

$$\begin{aligned} \mathcal{R}_{\text{spec}} &= \{(a(t_a).S, a(t_a).T) \mid a \in A, (S,T) \in \approx \} \\ \mathcal{R} &= \mathcal{R}_{\text{spec}} \cup \approx \ . \end{aligned}$$

Pour prouver que $\mathcal{R}$ est une bisimulation–ACTC il suffit alors de considérer une paire quelconque

$$(a(t_a).T, a(t_a).S) \in \mathcal{R}_{\text{spec}}$$

et de prouver que

1. pour toute transition $a(t_a).T \stackrel{b(v)}{\rightarrow} T'$ il existe une transition $a(t_a).S \stackrel{b(v)}{\rightarrow} S'$ telle que $(T', S') \in \mathcal{R}$;
2. pour toute transition $a(t_a).S \stackrel{b(v)}{\rightarrow} S'$ il existe une transition $a(t_a).T \stackrel{b(v)}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$;
3. $\alpha(a(t_a).T) = \alpha(a(t_a).S$ et
4. $\Gamma(a(t_a).T) = \Gamma(\alpha(a(t_a).S)$.

Si, comme dans cet exemple, la relation $\mathcal{R}_{\text{spec}}$ est symétrique par définition, il suffit de prouver la première implication, la deuxième étant analogue.

Montrons maintenant pourquoi il est nécessaire d'inclure la relation $\approx$ dans $\mathcal{R}$. Dans notre exemple, toute transition de $a(t_a).T$ est telle que $a(t_a).T \stackrel{a(v)}{\rightarrow} \trianglelefteq_v(T)$, où $v \in D$. De même, $a(t_a).S$ exécute l'action $a(v)$ et transite vers $\trianglelefteq_v(S)$. Or, par définition, $T \approx S$. Nous pouvons alors utiliser la propriété, que nous aurons déjà prouvée auparavant, selon laquelle $\approx$ est une congruence par rapport aux opérateurs

de décalage, et déduire que

$$(\lhd_v(S'), \lhd_v(T')) \in \approx \subseteq \mathcal{R}.$$

Typiquement, comme dans l'exemple présenté ci-haut, $\mathcal{R}_{\text{spec}}$ contient des paires de termes avec une *syntaxe* spécifique. Parfois, par contre, la définition de $\mathcal{R}_{\text{spec}}$ peut être plus compliquée. Prenons, par exemple, la relation $\mathcal{R}_{\text{spec}}$ de la preuve de congruence de $\approx$ par rapport à l'opérateur Loop (annexe, section 12.3.5.7). Nous définissons alors d'abord une relation $\mathcal{R}_{\text{synt}} \in \mathcal{L} \times \mathcal{L}$ dont les termes ont une syntaxe spécifique. La relation $\mathcal{R}_{\text{spec}}$ est ensuite définie comme suit:

$$\mathcal{R}_{\text{spec}} = \{(S,T) \mid (\exists (S_0, T_0) \in \mathcal{R}_{\text{synt}})(S \approx S_0 \wedge T \approx T_0)\}$$

Ainsi, $\mathcal{R}_{\text{spec}}$ contient des termes avec une certaine *sémantique*.

Dans ce qui suit nous allons montrer que, pour prouver qu'une paire quelconque $(S,T) \in \mathcal{R}_{\text{spec}}$ a les propriétés (1)–(3) d'une bisimulation–ACTC(8.1), il suffit de considérer juste les paires de termes $(S_0, T_0) \in \mathcal{R}_{\text{synt}}$. Remarquons que, puisque $\approx$ est réflexive, $\mathcal{R}_{\text{synt}} \subseteq \mathcal{R}_{\text{spec}}$.

Soient $S, T, S_0, T_0 \in \mathcal{L}$ des termes quelconques tels que $T \approx T_0$ et $S \approx S_0$ et tels que $(S_0, T_0) \in \mathcal{R}_{\text{spec}}$ et la paire (S_0, T_0) satisfait aux conditions (1)–(3) d'une bisimulation ACTC.

Premièrement, si $\alpha(S_0) = \alpha(T_0)$, alors pour toute paire (S,T) avec $S \approx S_0$ et $T \approx T_0$ on a que

$$\alpha(S) \stackrel{\text{def. 8.1}}{=} \alpha(S_0) = \alpha(T_0) \stackrel{\text{def. 8.1}}{=} \alpha(T).$$

Selon le même raisonnement, si $\Gamma(S_0) = \Gamma(T_0)$ alors pour toute paire (S,T) avec $S \approx S_0$ et $T \approx T_0$ on a que $\Gamma(S) = \Gamma(T)$. De même, si $\text{exec}(S_0) = \text{exec}(T_0)$, alors $\text{exec}(S) = \text{exec}(T)$.

Soit $T \stackrel{a(v)}{\rightarrow} T'$ une transition quelconque de T. Alors, puisque $T \approx T_0$, il existe un transition $T_0 \stackrel{a(v)}{\rightarrow} T_0'$ telle que $T' \approx T_0'$. Donc, il existe une transition $S_0 \stackrel{a(v)}{\rightarrow} S_0'$ avec $(S_0', T_0') \in \mathcal{R}$. De plus, puisque $S \approx S_0$ il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$ telle que $S' \approx S_0'$.

Or, $(S'_0, T'_0) \in \mathcal{R}$ implique soit que

1. il existe une paire $(S''_0, T''_0) \in \mathcal{R}_{\text{synt}}$ telle que $S'_0 \approx S''_0$ et $T'_0 \approx T''_0$ ou bien que
2. $(S'_0, T'_0) \in \approx$.

Dans le premier cas on déduit, par transitivité de la relation $\approx$, que $S' \approx S''_0$ et $T \approx T''_0$. Donc, $(S', T') \in \mathcal{R}_{\text{spec}} \subseteq \mathcal{R}$. Dans le deuxième cas, puisque $\approx$ est transitive, on déduit que $(S', T') \in \approx \subseteq \mathcal{R}$, ce qu'il fallait prouver.

8.3 Propriétés de congruence

Dans cette section nous prouvons que la relation d'équivalence $\approx$ est une congruence sur tout le langage $\mathcal{L}$ (théorème 8.10). Il existe des résultats théoriques [44] qui permettent de vérifier la congruence d'une relation de bisimulation sur un langage algébrique spécifié par une sémantique opérationnelle, en autant que la forme des règles de sémantique opérationnelle respecte un format spécifique. Ce format, appelé format *panf* [44], est trop restrictif pour exprimer la sémantique du langage de spécification de ACTC. Nous sommes donc obligés de prouver la congruence de $\approx$ en détail, à partir de la définition de $\approx$. Des résultats préliminaires concernant le prédicat Wait, qui sont nécessaires à la preuve de la congruence de $\approx$, seront présentés dans la section suivante. Ensuite, dans la section 8.3.2, nous énonçons le théorème de congruence de $\approx$.

8.3.1 Le prédicat Wait

Rappelons que pour tout $T \in \mathcal{L}$ et tout $v \geq 0$, le prédicat $\text{Wait}(v, T)$ est vrai si et seulement s'il existe une transition $T \xrightarrow{a(v')} T'$ avec $T' \in \mathcal{L}, a \in A$, et $v' \geq v$. Évidemment, si un terme T peut attendre jusqu'à un temps v, alors il peut attendre moins longtemps:

$$(\forall v' \leq v)(\text{Wait}(v, T) \Rightarrow \text{Wait}(v', T))$$

Pour simplifier la vérification de la validité du prédicat Wait nous avons établi un nombre de propriétés que ce prédicat satisfait.

La proposition suivante affirme que lorsque deux termes quelconques T et S ont les mêmes traces, les prédicats $\text{Wait}(v, T)$ et $\text{Wait}(v, S)$ sont équivalents.

Proposition 8.2 *Soient* $(S, T) \in \sim$ *deux termes quelconques équivalents du point de vue des traces. Alors,*

$$(\forall v \geq 0)(\text{Wait}(v, S) \Leftrightarrow \text{Wait}(v, T)).$$

Preuve: $\text{Wait}(v, S)$ est vrai si et seulement s'il existe une transition $S \stackrel{a(v')}{\rightarrow} S'$, avec $a \in A$ et $v' \geq v$. Puisque $S \sim T$, ils ont les mêmes traces. Donc T aussi peut exécuter $a(v)$. Donc, le prédicat $\text{Wait}(v, T)$ est vrai aussi. L'implication inverse est analogue. ∎

En tenant compte de l'équation 8.2 on peut alors énoncer le corollaire suivant:

Corollaire 8.3 *Soient* $(S, T) \in \approx$ *deux termes bisimilaires quelconques. Alors,*

$$(\forall v \geq V_0)(\text{Wait}(v, S) \Leftrightarrow \text{Wait}(v, T)).$$

Des propriétés analogues peuvent être énoncées pour le prédicat *Urgent.*

Proposition 8.4 *Soient* $(S, T) \in \sim$ *deux termes quelconques équivalents du point de vue des traces. Alors,*

$$(\forall v \geq V_0, \forall a \in A)(\text{Urgent}(a(v), S) \Leftrightarrow \text{Urgent}(a(v), T)).$$

Preuve: La preuve est triviale: il suffit de remarquer que si $S \sim T$, alors $a(v) \in \text{exec}(T) \Leftrightarrow a(v) \in \text{exec}(S), \forall a \in A, v \geq V_0$. ∎

Corollaire 8.5 *Soient* $(S, T) \in \approx$ *deux termes bisimilaires quelconques. Alors,*

$$(\forall v \geq V_0, \forall a \in A)(\text{Urgent}(a(v), S) \Leftrightarrow \text{Urgent}(a(v), T)).$$

La proposition suivante définit une méthode récursive pour le calcul du prédicat $\text{Wait}(v, \cdot)$ de certains termes composés, à partir de leurs sous-termes. Notons que dans le cas d'un terme T composé avec les opérateurs θ, Max, Min, et *Excep* on ne peut pas établir une équivalence entre le prédicat $\text{Wait}(v, T)$ et une formule basée sur le prédicat appliqué aux sous-termes de T; nous ne pouvons prouver qu'une implication unilatérale.

Proposition 8.6 *Soient $T, T_i, i = 1, 2, 3$ des termes quelconques de $\mathcal{L}_\beta^+$. Alors,*

1. $\text{Wait}(v, a(t_a).T) \Leftrightarrow v \geq V_0$,
2. *(a)* $v' \geq v \Rightarrow \text{Wait}(v, \lhd_{v'}(T))$,

   (b) $v' < v \Rightarrow (\text{Wait}(v, \lhd_{v'}(T)) \Leftrightarrow \text{Wait}(v, T))$,
3. *(a)* $v' \geq v \Rightarrow \text{Wait}(v, \trianglelefteq_{v'}(T))$,

   (b) $v' < v \Rightarrow (\text{Wait}(v, \trianglelefteq_{v'}(T)) \Leftrightarrow \text{Wait}(v, T))$,
4. $\text{Wait}(v, \theta : T) \Rightarrow \text{Wait}(v, T)$,
5. $\text{Wait}(v, \text{Max}(o, H, [m, M], T)) \Rightarrow \text{Wait}(v, T)$,
6. $\text{Wait}(v, \text{Min}(o, H, [m, M], T)) \Rightarrow \text{Wait}(v, T)$,
7. $\text{Wait}(v, \|(T_1, \ldots, T_m)) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v, T_1)$,
8. $\text{Wait}(v, +(T_1, \ldots, T_m)) \Leftrightarrow \bigvee_{i=1}^{m} \text{Wait}(v, T_i)$,
9. $\text{Wait}(v, \text{Seq}(T_1, T_2)) \Leftrightarrow \text{Wait}(v, T_1)$,
10. $\text{Wait}(v, [\mathcal{C}, \epsilon](T_1, \ldots, T_m)) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v, T_i)$,
11. $\text{Wait}(v, \oplus(T_1, \ldots, T_m)) \Rightarrow \bigvee_{i=1}^{m} \text{Wait}(v, T_i)$,
12. $\text{Wait}(v, \text{Ex}(T_n, T_c, T_e)) \Leftrightarrow \text{Wait}(v, T_n) \wedge \text{Wait}(v, T_c)$,

13. $\text{Wait}(v, \text{Loop}(T)) \Leftrightarrow \text{Wait}(v, T)$.

pour tous $v \geq 0$, $a \in A$, *et* $v' \geq 0$.

Preuve: La première égalité est triviale. Les égalités 2 et 3 sont analogues. Nous prouverons ici la seconde égalité concernant l'opérateur $\lhd$. Les autres preuves se trouvent en annexe, section 12.3.1.

Prouvons l'implication à gauche. Supposons d'abord que $v' \geq v$. Le prédicat $\text{Wait}(v, \lhd_{v'}(T))$ est alors vrai, puisque $\lhd_{v'}(T)$ peut exécuter au moins une action, à la limite l'action de blocage δ. Selon la sémantique opérationnelle de l'opérateur $\lhd$, toutes ces transitions ont lieu au temps v' ou plus tard.

Dans la situation où $v' < v$, selon la sémantique de l'opérateur de décalage, $\text{exec}^{\geq v}(T) = \text{exec}^{\geq v}(\lhd_{v'}(T))$. Donc, les prédicats $\text{Wait}(v, \lhd_v(T'))$ et $\text{Wait}(T)$ sont équivalents, ce qu'il fallait prouver. ∎

Remarquons ici que, si seules les actions différentes de δ sont considérées, alors les prédicats $\text{Wait}(v, \lhd_{v'}(T))$ et $\text{Wait}(v, T)$ sont équivalents, indépendamment si $v' \geq v$ ou non.

La proposition suivante exprime le fait que pour tout terme T, qui détecte une situation de blocage à un temps v, l'exécution de l'action de blocage $\delta(v)$ est urgente et T ne peut exécuter aucune autre action après le temps v. T peut, par contre, exécuter des actions différentes de δ au temps v. Dans ce cas, le blocage sera signalé après l'exécution de ces actions. Autrement dit, l'exécution de $\delta(v)$ est *impérative*: $\delta(v)$ sera forcement exécutée par T ou un de ses successeurs.

Proposition 8.7 *Soit* $T \in \mathcal{L}_\beta$ *un terme feuille quelconque. Si* $\delta(v) \in \text{exec}(T)$, *alors* $\neg\text{Wait}(v', T)$, *pour tout* $v' > v$. *De plus, pour toute action* $a(v) \in \text{exec}(T)$, *telle que* $a \neq \delta$, *toutes les traces* $\mu = a(v)... \in \mathbb{T}(T)$, *doivent finir dans un blocage, autrement dit* $\mu = a(v)...\delta(v)$.

Preuve: voir l'annexe, section 12.3.2.

Considérons maintenant une variante différente du prédicat Wait, que nous notons Wait_β:

$$\text{Wait}_\beta(v,T) \stackrel{\text{def.}}{\Leftrightarrow} (\forall a \in \text{exec}(T))(\exists v' \geq v)(a(v') \in \text{exec}(T)).$$

Donc, $\text{Wait}_\beta(v,T)$ est vrai si et seulement si toutes les actions que T peut exécuter avant le temps v peuvent encore être exécutées au temps v ou plus tard. En utilisant le prédicat Urgent, ceci revient à dire que

$$\text{Wait}_\beta(v,T) \stackrel{\text{def.}}{\Leftrightarrow} (\forall a \in \text{exec}(T))(\forall v' < v)\neg\text{Urgent}(a(v'),T).$$

C'est trivialement vrai que $\text{Wait}_\beta(v,T)$ implique $\text{Wait}(v,T)$. Si T est un terme feuille nous pouvons affirmer encore plus: nous prouvons dans la proposition 8.8 que pour tout terme $T \in \mathcal{L}_\beta$ et tout $v \geq 0$ les prédicats $\text{Wait}(v,T)$ et $\text{Wait}_\beta(v,T)$ sont équivalents. Intuitivement, l'équivalence se justifie par le fait qu'un terme feuille décrit une machine à transitions sans branchement. Avant d'énoncer cette proposition, considérons l'exemple suivant qui illustre bien pourquoi l'équivalence entre Wait et Wait_β ne s'étend pas aux termes hiérarchiques.

Exemple 8.3 *Soit $T \in \mathcal{L}$ défini comme:*

$$T = a(2)||(b(1) + c(3)).$$

La notation $a(2) \stackrel{\text{not.}}{=} t_a = 2 : a(t_a)$ a été introduite pour sa simplicité.

Au temps $v = 1$ le terme $b(1) + c(3)$ choisit d'une manière non-déterministe d'exécuter $b(1)$ ou d'attendre jusqu'au temps 3 pour exécuter $c(3)$. Par définition, le prédicat $\text{Wait}(2, b(1) + c(3))$ et donc vrai. Selon la sémantique opérationnelle de la composition parallèle, T peut alors exécuter $a(2)$ et ensuite $c(3)$, tout comme il peut choisir d'une manière non-déterministe d'exécuter la trace $b(1).a(2)$.

Le prédicat $\text{Wait}_\beta(2, b(1) + c(3))$, par contre, est faux parce que l'action b ne peut être exécutée qu'au temps $v = 1$. Si, dans la sémantique opérationnelle de la composition parallèle on avait employé le prédicat Wait_β plutôt que Wait, alors T n'aurait pas pu exécuter la trace $a(2).c(3)$, ce qui ne se justifie pas.

Proposition 8.8 *Soit $T \in \mathcal{L}_\beta$ un terme feuille quelconque. Alors, pour tout $v \geq V_0$, le prédicat* $\text{Wait}(v, T)$ *est vrai si et seulement si toutes les actions de* $\text{exec}(T)$ *peuvent encore être exécutées au temps v ou plus tard. Formellement,*

$$\text{Wait}(v, T) \Leftrightarrow \text{Wait}_\beta(v, T).$$

Preuve: L'implication à gauche est triviale. L'implication à droite est prouvée par induction structurelle. Les termes préfixés, qui satisfont le théorème trivialement, sont à la base du raisonnement inductif. Le pas de l'induction consiste à considérer pour chaque opérateur $f \in \text{Op}_\beta$ un terme $T = f[T_1, \ldots, T_m]$ et de prouver que, si tous les opérandes T_i ont la propriété énoncée dans le théorème, T a cette propriété aussi.

Nous présentons ici la preuve pour l'opérateur de composition parallèle $||$. Les preuves correspondantes aux opérateurs temporels sont présentées en annexe, section 12.3.3.

Soit $T = ||(T_1, \ldots, T_m)$ un terme dont les sous-termes T_i ont la propriété énoncée, c'est-à-dire pour tout $i, \leq i \leq m$ et tout $v \geq 0$

$$\text{Wait}(v, T_i) \Leftrightarrow \text{Wait}_\beta(v, T_i).$$

Selon la proposition 8.6 nous avons que

$$\text{Wait}(v, T) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v, T_m).$$

Puisque, par l'hypothèse d'induction, $\text{Wait}_\beta(v, T_i) \Leftrightarrow \text{Wait}(v, T_i)$, on déduit que

$$\text{Wait}(v, T) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}_\beta(v, T_m).$$

Pour prouver le pas d'induction il suffit maintenant de montrer que

$$\bigwedge_{k=1}^{m} \text{Wait}_\beta(v, T_k) \Rightarrow \text{Wait}_\beta(v, T).$$

Nous avons déjà prouvé cette propriété pour le prédicat Wait en annexe (section 12.3.1). La preuve pour le prédicat Wait_β est analogue. Donc, nous pouvons conclure que T a la propriété énoncée dans la proposition. ∎

Une conséquence immédiate de cette équivalence entre Wait et Wait_β est que dans tout terme feuille une action $a(v)$ est exécutée si seulement si il n'existe aucune autre action qui soit plus urgente.

On peut utiliser le prédicat Wait_β pour exprimer une propriété supplémentaire de l'opérateur $\oplus$: pour tous $T_1, \ldots, T_m \in \mathcal{L}$ et tout $v \geq 0$

$$\text{Wait}(v, \oplus(T_1, \ldots, T_m)) \Leftarrow \bigvee_{i=1}^{m} \text{Wait}_\beta(v, T_i). \tag{8.3}$$

Preuve: La preuve de cette propriété se trouve dans la section 12.3.4.

8.3.2 Le théorème de congruence

La relation d'équivalence sémantique $\approx \subseteq \mathcal{L} \times \mathcal{L}$ est à la base de la définition formelle de l'algèbre de processus temporisée ACTC. Tel que nous l'avons montré dans la section 4.5, il est essentiel que la relation $\approx$ soit une congruence pour l'ensemble d'opérateurs de Op. Dans la section 8.1 nous avons alors défini la relation $\approx$ comme un sous-ensemble de $\simeq$ par l'ajout des deux restrictions supplémentaires:

$$(T \approx S) \Leftrightarrow (T \simeq S \wedge \alpha(T) = \alpha(S) \wedge \Gamma(T) = \Gamma(S)).$$

Nous avons alors montré par des exemples que ces deux conditions sont nécessaires pour que $\approx$ soit une congruence. Dans cette section nous prouverons que ces conditions sont aussi suffisantes, donc que $\approx$ est une congruence pour tous les opérateurs de Op.

Nous commençons par la preuve du lemme suivant:

Lemme 8.9 *Pour tout* $T = f[T_1, \ldots, T_n], S = f[S_1, \ldots, S_n] \in \mathcal{L}$, *si* $f \notin \{\trianglelefteq, \triangleleft\}$ *et si*

$$(\forall i = 1, n)(T_i \approx S_i)$$

alors

$$\alpha(T) = \alpha(S).$$

Preuve: la preuve suit directement des définitions. ∎

Nous pouvons maintenant énoncer et prouver le théorème de la congruence de $\approx$:

Théorème 8.10 *La relation de bisimulation–*ACTC $\approx$ *est une congruence pour tous les opérateurs de* Op *sur le langage* $\mathcal{L}$. *Plus précisément:*

1. *pour tout* $v \geq V_0$ *et tous* $T, S \in \mathcal{L}$

$$T \approx S \Rightarrow \trianglelefteq_v(T) \approx \trianglelefteq_v(S)$$
$$T \approx S \Rightarrow \triangleleft_v(T) \approx \triangleleft_v(S)$$

2. *pour tout* $a \in A$ *et* $T, S \in \mathcal{L}_\beta$,

$$T \approx S \Rightarrow a(t_a).T \approx a(t_a).S$$

3. *pour tous* $T_1, \ldots, T_m, S_1, \ldots, S_m \in \mathcal{L}_\beta$,

$$(\forall i = 1, \ldots, m)(T_i \approx S_i) \Rightarrow \|(T_1, \ldots, T_m) \approx \|(S_1, \ldots, S_m);$$

4. *pour tous* $T_1, S_1 \in \mathcal{L}_\beta$, *et tout* $\theta \in \Theta$ *tel que* $\mathrm{Act}(\theta) \subseteq \mathrm{Act}(T_1)$ *et* $\mathrm{Act}(\theta) \subseteq \mathrm{Act}(S_1)$,

$$T_1 \approx S_1 \Rightarrow \theta : T_1 \approx \theta : S_1;$$

5. *pour tout opérateur réactif* $\mathrm{ROp}= (o, H, [m, M], \cdot)$ *et tous* $T, S \in \mathcal{L}_\beta$, *tels que* $\{o\} \cup H \subseteq \mathrm{Act}(T)$ *et* $\{o\} \cup H \subseteq \mathrm{Act}(S)$,

$$T \approx S \Rightarrow \mathrm{ROp}(T) \approx \mathrm{ROp}(S)$$

6. *pour tous* $n \geq 2$ *et* $T_1, \ldots T_n, S_1, \ldots, S_n \in \mathcal{L}$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow \sum_{i=1}^{n} T_i \approx \sum_{i=1}^{n} S_i$$

7. *pour tous* $T_1, T_2, S_1, S_2 \in \mathcal{L}$

$$(\forall i = 1, 2)(T_i \approx S_i) \Rightarrow \mathrm{Seq}(T_1, T_2) \approx \mathrm{Seq}(S_1, S_2)$$

8. *pour tous* $T, S \in \mathcal{L}$,

$$T \approx S \Rightarrow \text{Loop}(T) \approx \text{Loop}(S).$$

9. *pour tous* $n \geq 2$ *et* $T_1, \ldots, T_n, S_1, \ldots, S_n \in \mathcal{L}$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow \oplus(T_1, \ldots, T_n) \approx \oplus(S_1, \ldots, T_n)$$

10. *pour tous* $n \geq 2$, *tous* $T_1, \ldots, T_n, S_1, \ldots, S_n \in \mathcal{L}$, *et tout opérateur de composition parallèle avec communication* $[\mathcal{C}, \epsilon]$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow [\mathcal{C}, \epsilon](T_1, \ldots, T_n) \approx [\mathcal{C}, \epsilon](S_1, \ldots, S_n)$$

11. *pour tous* $T_{n1}, T_{c1}, T_{e1}, T_{n2}, T_{c2}, T_{e2} \in \mathcal{L}$,

$$(T_{n1} \approx T_{n2}) \wedge (T_{c1} \approx T_{c2}) \wedge (T_{e1} \approx T_{e2}) \Rightarrow \&$$
$$\text{Ex}(T_{n1}, T_{c1}, T_{e1}) \approx \text{Ex}(T_{n2}, T_{c2}, T_{e2})$$

Preuve: à titre d'exemple nous présentons ici la preuve de la congruence de $\approx$ par rapport à l'opérateur ":". Les autres preuves sont données dans l'annexe, section 12.3.5.

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(\theta : T_1, \theta : S_1) \mid T_1, S_1 \in \mathcal{L}_\beta, T_1 \approx S_1, \theta \in \Theta\} \cup \approx .$$

Suivant le raisonnement présenté dans la section 8.2 il suffit de considérer une paire $(T, S) \in \mathcal{R}$ telle que $T = \theta : T_1$ et $S = \theta : S_1$, avec $(T_1, S_1) \in \approx$ et $\theta \in \Theta$ et de prouver que (T, S) satisfait aux conditions énoncées dans la définition 8.1.

Premièrement, selon la proposition 8.9,

$$\alpha(T) = \alpha(S).$$

Ensuite,

$$\Gamma(T) = \Gamma(T_1) \cap \text{Sol}(\theta),$$
$$\Gamma(S) = \Gamma(S_1) \cap \text{Sol}(\theta).$$

Or, puisque $T_1 \approx S_1$, par définition $\Gamma(T_1) = \Gamma(S_1)$. On déduit que $\Gamma(T) = \Gamma(S)$.

Finalement, puisque la relation $\mathcal{R}$ est symétrique par définition, il ne reste qu'à prouver que pour toute transition $T \xrightarrow{a(v)} T'$ il existe une transition $S \xrightarrow{a(v)} S'$ telle que $(T', S') \in \mathcal{R}$.

Supposons d'abord que la transition de T suit la règle $\mathbf{supp}_1$. Il existe alors une transition $T_1 \xrightarrow{a(v)} T_1'$ avec $a \neq \delta$ et $T_1 \neq \uparrow$ telle que le prédicat $\text{First}(a(v), T)$ est vrai. Comme $T_1 \approx S_1$ on déduit qu'il existe aussi une transition $S_1 \xrightarrow{a(v)} S_1'$ telle que $T_1' \approx S_1'$.

Prouvons que le prédicat $\text{First}(a(v), S)$ est vrai.

$$\text{First}(a(v), S) \overset{\text{def}}{\Leftrightarrow} (\Gamma^{\geq v}(S) \cap \text{Sol}(t_a = v)) \neq \emptyset.$$

Or, nous avons déjà montré que $\Gamma(S) = \Gamma(T)$. Donc,

$$\text{First}(a(v), S) \Leftrightarrow \text{First}(a(v), T).$$

On peut alors appliquer la règle $\mathbf{supp}_1$ pour déduire qu'il existe une transition $S \xrightarrow{a(v)} S'$. Finalement, on a que

$$\begin{aligned} T' &= \theta_{v/t_a} : T_1', \\ S' &= \theta_{v/t_a} : S_1'. \end{aligned}$$

On conclut alors que $(T', S') \in \mathcal{R}$, ce qu'il fallait prouver.

Si la transition $T \xrightarrow{a(v)} T'$ est définie par la règle $\mathbf{supp}_2$ on suit un raisonnement analogue à celui présenté pour la règle $\mathbf{supp}_1$. La seule remarque supplémentaire qui doit être faite est que le seul terme équivalent à $\uparrow$ est le terme $\uparrow$ même. De plus, $(\uparrow, \uparrow) \in \approx \subseteq \mathcal{R}$.

Le cas où la transition de T est définie par la règle $\mathbf{supp}_3$ est trivial, il suffit de remarquer que $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$.

Pour la règle $\mathbf{supp}_4$ on note que, puisque $T_1 \approx S_1$, alors $\text{exec}(T_1) = \text{exec}(S_1)$. Ensuite, puisque nous avons déjà prouvé que pour toute action temporisée $a(v)$ les

prédicats $\text{First}(a(v), T)$ et $\text{First}(a(v), S)$ sont équivalents, on a que

$$S \overset{\delta(\alpha(S))}{\rightarrow}\downarrow \text{ et } T \overset{\delta(\alpha(T))}{\rightarrow}\downarrow .$$

Or, nous avons déjà montré que $\alpha(T) = \alpha(S)$ et $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$. Ceci résume la preuve de la congruence de $\approx$ par rapport à l'opérateur θ :. ∎

Les exemples 8.1 et 8.2 montrent que la bisimulation forte $\simeq$ n'est pas une congruence par rapport à l'opérateur :. Les mêmes exemples montrent que les relations de bisimulation forte $\simeq$ et la relation d'équivalence de traces $\sim$ ne sont pas des congruences. Nous pouvons, par contre, prouver des résultats plus restreints, dont nous aurons besoin dans la section 10.2.4:

Proposition 8.11 *Les affirmations suivantes sont vraies:*

1. $(\forall v \geq V_0)(\forall T, S \in \mathcal{L})(T \sim S \Rightarrow \trianglelefteq_v(T) \sim \trianglelefteq_v(S))$ *et* $(T \simeq S \Rightarrow \trianglelefteq_v(T) \simeq \trianglelefteq_v(S))$;
2. $(\forall v \geq V_0)(\forall T, S \in \mathcal{L})(T \sim S \Rightarrow \triangleleft_v(T) \sim \triangleleft_v(S))$ *et* $(T \simeq S \Rightarrow \triangleleft_v(T) \simeq \triangleleft_v(S))$;
3. $(\forall i \in I)(T_i \sim S_i) \Rightarrow (\sum_{i\in I} T_i \sim \sum_{i\in I} S_i)$.

Preuve: Les preuves des affirmations 1 et 2 sont analogues. Nous prouvons les affirmations 1 et 3 en annexe, section 12.3.6. ∎

Nous disposons maintenant d'un langage de spécification de chronogrammes $\mathcal{L}$ avec une sémantique opérationnelle SOp et d'une relation d'équivalence opérationnelle $\approx$, qui est une congruence pour tous les opérateurs de composition de $\mathcal{L}$. À partir de ces notions nous sommes en mesure de définir l'algèbre de processus ACTC dans le chapitre suivant.

Chapitre 9

L'algèbre de processus temporisée ACTC

Dans cette section nous suivons la méthodologie présentée dans la section 4.5 pour définir l'algèbre de processus ACTC à partir du langage de spécification algébrique $\mathcal{L}$, de la sémantique opérationnelle des termes de SOp, et de la relation d'équivalence sémantique de termes $\approx$.

Nous donnons ensuite une axiomatisation **A** de ACTC et prouvons qu'elle est cohérente par rapport à la relation $\approx$ pour tous langage de termes $\mathcal{L}$. Les questions de la décidabilité de $\approx$ et de la complétude de **A** seront abordées dans le chapitre suivant.

9.1 Définition de ACTC

L'algèbre de processus ACTC $= (\mathcal{L}_{\mathrm{ACTC}}, \mathrm{Op}_{\mathrm{ACTC}})$ est définie sur le domaine des classes d'équivalence du langage de spécification $\mathcal{L}$ modulo la relation $\approx$:

$$\mathcal{L}_{\mathrm{ACTC}} \stackrel{\mathrm{def}}{=} \mathcal{L}/\approx .$$

L'ensemble de fonctions $\mathrm{Op}_{\mathrm{ACTC}}$ est l'interprétation des symboles de fonction de la signature Op du langage $\mathcal{L}$ comme fonctions sur $\mathcal{L}_{\mathrm{ACTC}}$:

$$\mathrm{Op}_{\mathrm{ACTC}} = \{f_{\mathrm{ACTC}} : \mathcal{L}^n_{\mathrm{ACTC}} \longrightarrow \mathcal{L}_{\mathrm{ACTC}} \mid f \in \mathrm{Op}, \mathrm{ar}(f) = n, n \geq 0\},$$

où $\mathrm{ar}(f)$ est l'arité du symbole de fonction f tel que définie dans le langage $\mathcal{L}$. Rappelons que les actions temporisées $a(t_a)$ peuvent être considérées comme des fonctions d'arité 0. L'interprétation f_{ACTC} d'un symbole de fonction $f \in \mathcal{F}$ est définie comme suit:

$$\begin{aligned} a(t_a)_{\mathrm{ACTC}} &= [\mathrm{graphe}(a(t_a))] \bmod \approx; \\ f_{\mathrm{ACTC}}(T^1_{\mathrm{ACTC}}, \ldots, T^n_{\mathrm{ACTC}}) &= f[T^1, \ldots, T^n]_{\mathrm{ACTC}}; \\ &= [\mathrm{graphe}(f[T^1, \ldots, T^n])] \bmod \approx . \end{aligned}$$

Nous avons ainsi formellement défini l'algèbre de processus ACTC à partir du langage algébrique de spécification de chronogrammes $\mathcal{L}$, de la sémantique opérationnelle SOp associée aux termes de $\mathcal{L}$ et de la relation d'équivalence sémantique de termes $\approx$ définie sur $\mathcal{L}$.

9.2 Axiomatisation de ACTC

9.2.1 Rappel de définitions

Rappelons que dans la section 4.3 nous avons défini les notions de spécification équationnelle ϵ qui associe à un langage de termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$ de signature $\mathcal{F}$ et variables $\mathcal{X}$ un ensemble d'axiomes $\mathbf{A}$ ainsi qu'un ensemble de règles d'inférence R_E qui permettent de déduire des égalités de termes d'une manière purement syntaxique. Le système d'axiomes $\mathbf{A}$, aussi appelé *axiomatisation* est un ensemble d'égalités de termes de $\mathcal{T}(\mathcal{F}, \mathcal{X})$. L'ensemble des règles d'inférence R_E d'une spécification équationnelle est typiquement l'ensemble de règles d'inférence de la logique équationnelle. R_E contient les règles suivantes:

1. les propriétés d'*équivalence* de $=$:
   pour tous $t, s, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   - *réflexivité*: $t = t$
   - *symétrie*: $t = s$ implique $s = t$
   - *transitivité*: $t = s$ et $s = r$ implique $t = r$
2. la règle de *substitution*:
   pour tous $t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et toute substitution $\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow T(\mathcal{F}, \mathcal{X})$,
   $$t_1 = t_2 \text{ implique } \varphi(t_1) = \varphi(t_2);$$
3. la règle de *contexte*:
   pour tout $n \geq 1$ et tous $t_1, \ldots, t_n, s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   $$(\forall i : 1 \leq i \leq n)(t_i = s_i), \text{ implique } f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n).$$

Remarquons que si le langage de termes est clos, c'est à dire $\mathcal{T}(\mathcal{F})$ ne contient pas de variables, alors R_E contient uniquement les règles d'équivalence et de contexte.

On dit qu'une égalité de termes $s = t$ peut être *dérivée* dans $\mathcal{E}$ et on écrit $\mathcal{E} \vdash s = t$, ou bien $\mathbf{A} \vdash s = t$ si et seulement si soit

- $s = t \in \mathbf{A}$ ou bien
- l'égalité $s = t$ peut être déduite à partir des axiomes de $\mathbf{A}$ en utilisant les règles d'inférence de R_E.

Un système de spécification équationnelle ϵ induit une relation d'*équivalence équationnelle* ou *syntaxique* de termes, notée $\mathcal{R}_\epsilon$, telle que

$$\mathcal{R}_\epsilon = \{(s, t) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X}) \mid \mathcal{E} \vdash s = t\}.$$

Dans cette section nous donnons une spécification équationnelle au langage de termes $\mathcal{L}_{\text{ACTC}}$ en définissant un ensemble d'axiomes $\mathbf{A}$ pour $\mathcal{L}$. L'ensemble d'axiomes

$\mathbf{A}$ contient le sous-ensemble d'axiomes $\mathbf{A}_\beta$, concernants les opérateurs de Op_β, et les axiomes hiérarchiques $\mathbf{A}_\mathcal{H}$, définis pour les opérateurs de $\mathrm{Op}_\mathcal{H}$. Dans la section 9.2.3 nous présentons les axiomes $\mathbf{A}_\beta$ du langage de base et les axiomes du langage hiérarchique, $\mathbf{A}_\mathcal{H}$, dans la section 9.2.4.

Dans la section 9.2.5 nous prouvons que l'axiomatisation $\mathbf{A}$ est *cohérente* par rapport à la relation de bisimulation forte $\approx$, c'est à dire

$$(\forall T, S \in \mathcal{L})(\mathbf{A} \vdash T = S \Rightarrow T \approx S).$$

Implicitement alors $\mathbf{A}$ est cohérente par rapport à la bisimulation forte $\simeq$ et la relation d'équivalence de traces $\sim$.

Nous commençons par présenter dans la section 9.2.2 l'ensemble de propriétés algébriques d'opérateurs binaires communément utilisés pour définir des structures algébriques telles que les groupes, les anneaux ou les algèbres [30]. Ces propriétés sont ensuite étendues aux opérateurs d'arité arbitraire.

9.2.2 Propriétés algébriques élémentaires

Soit $\mathcal{A} = (i(\mathcal{F}), E)$ une $\mathcal{F}$–algèbre, où $\mathcal{F}$ est une signature quelconque et i est une interprétation des symboles de fonction de $\mathcal{F}$ comme fonctions sur un domaine quelconqe E.

Soient $+$ et $\times$ deux opérateurs quelconques de $\mathcal{A}$ et notons par $a, b, c, \ldots$ les éléments de E. Définissons d'abord les propriétés algébriques élémentaires d'opérateurs binaires.

L'opérateur $\times$ est *commutatif* si et seulement si, pour tous $a, b \in E$, $a \times b = b \times a$.

L'opérateur $\times$ est *associatif* si et seulement si, pour tous $a, b, c \in E$, $a \times (b \times c) = (a \times b) \times c$.

$u \in E$ est un *élément identité* pour l'opérateur $\times$ si et seulement si, pour tout $a \in E$, $a \times u = u \times a = a$.

$v \in E$ est un *élément absorbant* pour l'opérateur $\times$ si et seulement si, pour tout $a \in E$, $a \times v = v \times a = v$.

L'opérateur $\times$ est *idempotent* si et seulement si pour tout $a \in E$, $a \times a = a$.

L'opérateur $\times$ est *distributif à gauche* par rapport à l'opérateur $+$ si et seulement si $a \times (b + c) = (a \times b) + (a \times c)$, pour tous $a, b, c \in E$.

L'opérateur $\times$ est *distributif à droite* par rapport à l'opération $+$ si et seulement si $(b + c) \times a = (b \times a) + (c \times a)$, pour tous $a, b, c \in E$.

L'opérateur $\times$ est *distributif bilatéralement*, ou simplement distributif, par rapport à l'opération $+$ si et seulement si $\times$ est distributif à gauche et à droite par rapport à $+$.

L'opérateur $\times$ est *réductible à gauche* si et seulement si, pour tout $a, b, c \in E$, $c \times a = c \times b$ implique $a = b$.

L'opérateur $\times$ est *réductible à droite* si et seulement si, pour tout $a, b, c \in E$, $a \times c = b \times c$ implique $a = b$.

L'opérateur $\times$ est *bilatéralement réductible*, ou simplement *réductible*, si et seulement si $\times$ réductible à gauche et à droite.

Les propriétés suivantes sont évidentes:

Proposition 9.1 *Soit $\times$ et $+$ deux opérateurs quelconque d'une $\mathcal{F}$-algèbre $\mathcal{A}$. Alors,*

1. *si $\times$ est commutatif et unilatéralement distributif par rapport à $+$, alors $\times$ est bilatéralement distributif par rapport à $+$;*

2. *si $\times$ est unilatéralement réductible et commutatif alors $\times$ est bilatéralement réductible.*

Preuve: La preuve suit directement des définitions. ∎

Nous étendons maintenant les propriétés d'opérateurs binaires aux opérateurs d'arité arbitraire.

Rappelons qu'un opérateur $\times$ est d'arité arbitraire, s'il est défini pour un nombre m quelconque d'opérandes, où $m \geq 1$, notation $\times : E^* \longrightarrow E$. Ainsi $\times$ détermine

pour tout $m \geq 1$, un opérateur m-aire $\times^m : E^m \longrightarrow E$. Lorsqu'aucune confusion ne sera créée nous surchargeons la notation $\times$ pour désigner en même temps l'opérateur d'arité arbitraire ainsi que les opérateurs m-aires $\times^m$, pour un $m \geq 0$ quelconque.

Le théorème 9.3 montre qu'il suffit qu'un opérateur d'arité arbitraire $\times$ soit associatif pour que ses propriétés algébriques soient déduites à partir des propriétés algébriques de l'opérateur binaire correspondant $\times^2$. Ceci nous permet de simplifier l'axiomatisation de ACTC: pour les opérateurs d'arité arbitraire $\|$, $[]$, $+$, et $\oplus$, qui sont associatifs, il suffit d'énoncer des propriétés binaires.

Rappelons d'abord quelques définitions élémentaires.

Une fonction $\sigma : \{1, \ldots, m\} \longrightarrow \{1, \ldots, m\}$ est une *permutation* de l'ensemble $\{1, 2, \ldots, m\}$ si et seulement si σ est une bijection, donc pour tout $i \in \{1, \ldots, m\}$ il existe un unique $j \in \{1, \ldots, m\}$, tel que $\sigma(j) = i$. L'ensemble de permutations de $\{1, \ldots, m\}$ est noté $\mathcal{P}_m$. Une permutation notée $\sigma_{i,j}$ est une *inversion* si et seulement si

$$\begin{aligned} \sigma_{i,j}(i) &= j, \\ \sigma_{i,j}(j) &= i, \\ \sigma_{i,j}(k) &= k, \forall k \notin \{i, j\}. \end{aligned}$$

Donnons maintenant les définitions des propriétés algébriques d'opérateurs d'arité arbitraire. Soient $\times, + : E^* \longrightarrow E$ deux opérateurs d'arité arbitraire quelconques,

L'opérateur $\times : E^* \longrightarrow E$ est *associatif* si et seulement si pour tout $m \geq 3$ et tous $a_1, \ldots, a_m \in E$:

$$\begin{aligned} \times(a_1, \ldots, a_m) &= \times(a_1, \times(a_2, \ldots, a_m)) && (9.1) \\ &= \times(\times(a_1, \ldots, a_{m-1}), a_m). && (9.2) \end{aligned}$$

Les égalités 9.1 et 9.2 sont aussi appelées associativité à droite et à gauche, respectivement.

En prouvant l'équivalence suivante, nous donnons une définition alternative de l'associativité:

Proposition 9.2 *L'opérateur* $\times : E^* \longrightarrow E$ *est associatif si et seulement si pour tous* $m \geq 3$, $1 \leq i < j \leq m$ *et tous* $a_1, \ldots, a_m \in E$*:*

$$\times(a_1, \ldots, a_{i-1}, a_i, \ldots, a_j, a_{j+1}, \ldots, a_m) = \times(a_1, \ldots, a_{i-1}, \times(a_i, \ldots, a_j), a_{j+1}, \ldots, a_m).$$

Preuve: L'implication gauche est évidente. L'implication droite se prouve par induction généralisée sur m. La base de l'induction est le cas $m = 3$ qui est trivial. Supposons que l'implication est vrai pour tout $m \leq k$, où $k > 3$ est un entier quelconque (hypothèse d'induction). Prouvons alors qu'elle est vraie pour $m = k + 1$ aussi. Soient i et j tels que $1 < i < j \leq m + 1$. Alors

$$\begin{aligned}
\times(a_1, \ldots, a_{k+1}) &\overset{\text{assoc.}}{=} \times(a_1, \times(a_2, \ldots, a_{k+1})) \\
&\overset{\text{hyp. d'ind.}}{=} \times(a_1, \times(a_2, \ldots, a_{i-1}, \times(a_i, \ldots, a_j), a_{j+1}, \ldots, a_{k+1})) \\
&\overset{\text{hyp. d'ind.}}{=} \times(a_1, a_2, \ldots, a_{i-1}, \times(a_i, \ldots, a_j), a_{j+1}, \ldots, a_{k+1}).
\end{aligned}$$

Le cas $1 = i < j < m$ est est similaire: on utilise la propriété d'associativité à droite au lieu de l'associativité à gauche. Le cas où $i = 1$ et $j = m+1$ se prouve directement à partir des définitions. ∎

L'opérateur $\times : E^* \longrightarrow E$ est *commutatif* si et seulement si

$$\times(a_1, \ldots, a_m) = \times(a_{\sigma(1)}, \ldots, a_{\sigma(m)}),$$

pour tout $m \geq 2$, pour tous $a_1, \ldots, a_m \in E$ et toute permutation $\sigma \in \mathcal{P}_m$ de l'ensemble $\{1, \ldots, m\}$.

Soient $\times, + : E^* \longrightarrow E$ deux opérateurs quelconques d'arité arbitraire. Alors $\times$ est *distributif* par rapport à $+$ si et seulement si pour tous $m, n \geq 1$, pour tout $i : 1 \leq i \leq m$, et tous $a_1, \ldots, a_m, b_1, \ldots, b_n \in E$:

$$\times(a_1, \ldots, a_{i-1}, +(b_1, \ldots, b_n), a_{i+1}, \ldots, a_m) =$$

$$+(\times(a_1, \ldots, a_{i-1}, b_1, a_{i+1}, \ldots, a_m), \ldots, \times(a_1, \ldots, a_{i-1}, b_n, a_{i+1}, \ldots, a_m)).$$

Un *élément identité* pour l'opérateur $\times : E^* \longrightarrow E$ est défini comme $u \in E$ tel que pour tout $m \geq 2$ et tous $a_1, \ldots, a_m \in E$,

$$\begin{aligned} \times(a_1, \ldots, a_m, u) &= \times(a_1, \ldots, u, a_m) \\ &\vdots \\ &= \times(u, a_1, \ldots, a_m) \\ &= \times(a_1, \ldots, a_m), \end{aligned}$$

et pour tout $a \in e$, $a \times u = u \times a = a$.

Théorème 9.3 *Soient $\times, + : E^* \longrightarrow E$ deux opérateurs d'arité arbitraire quelconques. Si $\times$ et $+$ sont associatifs alors*

- *$\times$ est commutatif $\Leftrightarrow$ $\times^2$ est commutatif;*
- *u est un élément identité pour $\times$ $\Leftrightarrow$ u est un élément identité pour $\times^2$;*
- *v est un élément absorbant pour $\times$ $\Leftrightarrow$ v est un élément absorbant pour $\times^2$;*
- *$\times$ est distributif par rapport à $+$ $\Leftrightarrow$ $\times^2$ est distributif par rapport à $+^2$;*
- *$\times$ est réductible $\Leftrightarrow$ $\times^2$ est réductible;*
- *$\times$ est idempotent $\Leftrightarrow$ $\times^2$ est idempotent.*

Preuve: voir section 12.4.1 de l'annexe.

Remarquons ici que les propriétés algébriques décrites dans cette section peuvent être généralisées: l'identité $=$ peut être remplacée par une relation d'équivalence quelconque $\mathcal{R} \subseteq E^2$. On dit alors que les propriétés sont définies *modulo* la relation d'équivalence $\mathcal{R}$. Par exemple, un opérateur quelconque $\times : E^2 \longrightarrow E$ est dit commutatif modulo $\mathcal{R}$ si pour tous $a, b \in E$

$$\times(a, b) \mathcal{R} \times (b, a).$$

9.2.3 Axiomes du langage de termes feuilles $\mathbf{A}_\beta$

Nous énonçons ici le système d'axiomes $\mathbf{A}_\beta$ du langage de base $\mathcal{L}_\beta$. Le théorème 9.6, qui sera prouvé en annexe dans la section 12.4.2, affirme que $\mathbf{A}_\beta$ est cohérent par rapport à la relation $\approx$.

Dec1) Maximisation: pour tous $v, v' \geq V_0$ et tout $T \in \mathcal{L}$, si $v \geq v'$ alors

$$\begin{aligned} \trianglelefteq_v(\trianglelefteq_{v'}(T)) &= \trianglelefteq_v(T) \\ \triangleleft_v(\triangleleft_{v'}(T)) &= \triangleleft_v(T). \end{aligned}$$

Dec2) Identité de gauche restreinte: pour tout $T \in \mathcal{L}$

$$\trianglelefteq_{\alpha(T)}(T) = T.$$

Dec3) Distributivité des opérateurs de décalage par rapport aux opérateurs de Op\{.}: pour tout $v \geq V_0$, tout $m \geq 1, T_1, \ldots, T_m \in \mathcal{L}$, et tout opérateur $f \in \mathrm{Op}\backslash\{.\}$

1. $\triangleleft_v(f(T_1, \ldots, T_m)) = f(\triangleleft_v(T_1), \ldots, \triangleleft_v(T_m))$;
2. $\trianglelefteq_v(f(T_1, \ldots, T_m)) = f(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_m))$.

Pref1) Préfixage: pour tout terme préfixé $a(t_a).S \in \mathcal{L}_\beta$,

$$a(t_a).S = \bigwedge_{b \in \mathrm{Act}(S)} t_a < t_b : (a(t_a) \| S).$$

Par1) Commutativité: pour tous $T_1, T_2 \in \mathcal{L}_\beta$,

$$\|(T_1, T_2) = \|(T_2, T_1).$$

Par2) Associativité: pour tous $T_1, \ldots, T_m \in \mathcal{L}_\beta$, et tout $m > 2$

$$\|(T_1, \ldots, T_m) = \|(T_1, \|(T_2, \ldots, T_m)) = \|(\|(T_1, \ldots, T_{m-1}), T_m).$$

cons1) Conjonction: pour tous $\theta_1, \theta_2 \in \Theta$, tout $T \in \mathcal{L}_\beta$,

$$\theta_1 : \theta_2 : T = \theta_1 \wedge \theta_2 : T.$$

cons2) Élément identité: pour tout $T \in \mathcal{L}_\beta$,

$$\text{Vrai} : T = T.$$

cons3) Identité partielle: pour tout $T \in \mathcal{L}_\beta$, et tout instant v tel que $0 \leq v \leq \alpha(T)$, et toute action $a \in A(T)$:

$$t_a \geq v : T \ = \ T.$$

cons4) Distributivité de θ par rapport à $\|$: pour tout $\theta^i \in \Theta$, $T_1, \ldots, T_m \in \mathcal{L}_\beta$ tels que $\text{Act}(\theta^i) \subseteq \text{Act}(T_i)$, pour tous $m \geq 2, 1 \leq i \leq m$,

$$\bigwedge_{i=1}^{n} \theta^i : \|(T_1, \ldots, T_n) = \|(\theta^1 : T_1, \ldots, \theta^n : T_n).$$

cons5) Distributivité de θ par rapport aux opérateurs réactifs: pour tout $\theta \in \Theta$, tout $T \in \mathcal{L}_\beta$, et tout opérateur réactif ROp,

$$\theta : \text{ROp}(T) = \text{ROp}(\theta : T).$$

ROp1) Commutativité: pour tous $\text{ROp}_1, \text{ROp}_2$ opérateurs réactifs et tout $T \in \mathcal{L}_\beta$,

$$\text{ROp}_1(\text{ROp}_2(T)) = \text{ROp}_2(\text{ROp}_1(T)).$$

ROp2) Distributivité des opérateurs réactifs par rapport à $\|$: pour tous $T_1, \ldots, T_m \in \mathcal{L}_\beta$, où $m \geq 2$ et tout opérateur réactif $\text{ROp}(o, H, [m, M], \cdot)$ s'il existe un terme T_i, avec $1 \leq i \leq m$, tel que $H \cup \{o\} \subseteq \text{Act}(T_i)$, alors

$$\text{ROp}(o, H, [m, M], \|(T_1, \ldots, T_m)) = \|(T_1, \ldots, \text{ROp}(o, H, [m, M], T_i), \ldots, T_m).$$

9.2.4 Axiomes du langage hiérarchique $\mathbf{A}_\mathcal{H}$

Nous définissons ici le système d'axiomes du langage de termes hiérarchiques, noté $\mathbf{A}_\mathcal{H}$. La cohérence de $\mathbf{A}_\mathcal{H}$ par rapport à $\approx$ est énoncée dans le théorème 9.6, qui sera prouvé en annexe, section 12.4.2.

C1) Commutativité: pour tous $T_1, T_2 \in \mathcal{L}$,

$$+(T_1, T_2) = +(T_2, T_1).$$

C2) Associativité: pour tout $m > 2$, tous $T_1, \ldots, T_m \in \mathcal{L}$,

$$+(T_1, \ldots, T_m) = +(T_1, +(T_2, \ldots, T_m)) = +(+(T_1, \ldots, T_{m-1}), T_m).$$

C3) Idempotence : pour tout $T \in \mathcal{L}$

$$+(T, T) = T.$$

Seq1) Associativité: pour tous $T_1, T_2, T_3 \in \mathcal{L}$,

$$\text{Seq}(\text{Seq}(T_1, T_2), T_3) = \text{Seq}(T_1, \text{Seq}(T_2, T_3)).$$

Seq2) Distributivité à droite de Seq par rapport à $+$: pour tous $T_1, T_2, T_3 \in \mathcal{L}$,

$$\text{Seq}(+(T_1, T_2), T_3) = +(\text{Seq}(T_1, T_3), \text{Seq}(T_2, T_3)).$$

Seq3) Distributivité à gauche de Seq par rapport à $\oplus$: pour tous $T_1, T_2, T_3 \in \mathcal{L}$, si T_1 est un *observateur*, alors

$$\text{Seq}(T_1, \oplus(T_2, T_3)) = \oplus(\text{Seq}(T_1, T_2), \text{Seq}(T_1, T_3)).$$

Seq4) Distributivité à droite de Seq par rapport à $\oplus$: pour tous $T_1, T_2, T_3 \in \mathcal{L}$,

$$\text{Seq}(\oplus(T_1, T_2), T_3) = \oplus(\text{Seq}(T_1, T_3), \text{Seq}(T_2, T_3)).$$

Seq5) Distributivité restreinte de Seq par rapport à Ex: pour tout $T, T_n, T_c, T_e \in \mathcal{L}$, si T est un *observateur*, alors

$$\text{Seq}(T, \text{Ex}(T_n, T_c, T_e)) = \text{Ex}(\text{Seq}(T, T_n), \text{Seq}(T, T_c), T_e).$$

DC1) Commutativité: pour tous $T_1, T_2 \in \mathcal{L}$,

$$\oplus(T_1, T_2) = \oplus(T_2, T_1).$$

DC2) Associativité: pour tout $m \geq 3$ et tous $T_1, \ldots, T_m \in \mathcal{L}$,

$$\oplus(T_1, \oplus(T_2, \ldots, T_m)) = \oplus(\oplus(T_1, \ldots, T_{m-1}), T_m) = \oplus(T_1, \ldots T_m).$$

DC3) Idempotence restreinte: pour tout $T \in \mathcal{L}$, si T est un observateur alors,

$$\oplus(T, T) = T.$$

PCom1) Commutativité: pour tout terme $[\mathcal{C}, \epsilon](T_1, T_2) \in \mathcal{L}$,

$$[\mathcal{C}, \epsilon](T_1, T_2) = [\mathcal{C}, \epsilon](T_2, T_1).$$

PCom2) Associativité: pour tout terme $T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m) \in \mathcal{L}$, avec $m \geq 3$,

$$[\mathcal{C}, \epsilon](T_1, \ldots, T_m) = [\mathcal{C}, \epsilon]([\emptyset, \cdot](T_1, \ldots, T_{m-1}), T_m) = [\mathcal{C}, \epsilon](T_1, [\emptyset, \cdot](T_2, \ldots, T_m)).$$

PCom3) Réduction à la composition parallèle sans communication: pour tous $T_1, T_2 \in \mathcal{L}_\beta$,

$$[\emptyset, \cdot](T_1, T_2) = \|(T_1, T_2).$$

Ex1) Idempotence restreinte: pour tout observateur $T \in \mathcal{L}$ et tout $T_e \in \mathcal{L}$,

$$\mathrm{Ex}(T, T, T_e) = T.$$

Nous pouvons maintenant déduire des propriétés algébriques des opérateurs d'arité arbitraire $\|$, $\sum$, $\oplus$, et $[\ ]$:

Proposition 9.4 *Les opérateurs d'arité arbitraire* $+, \oplus, [\ ] : \mathcal{L}^* \longrightarrow \mathcal{L}$ *ont les propriétés suivantes:*

1. *les opérateurs* $\|, +, \oplus, [\ ] : \mathcal{L}^* \longrightarrow \mathcal{L}$ *sont commutatifs;*

2. *l'opérateur* $+ : \mathcal{L}^* \longrightarrow \mathcal{L}$ *est idempotent;*

3. *pour tout* $T \in \mathcal{L}$, *si* T *est un observateur alors*

$$\oplus(T, \ldots, T) = T.$$

Preuve: les preuves suivent directement des axiomes de **A** et de la proposition 9.3.

■

De plus, nous pouvons énoncer les propriétés suivantes:

Proposition 9.5 *Pour tout* $m \geq 2$ *et tous* $T_1, \ldots, T_m \in \mathcal{L}$, $\theta_{1,2} \in \Theta$, *et tout* $T \in \mathcal{L}_\beta$ *on a que*

1. $[\emptyset, \cdot](T_1, \ldots, T_m) = \|(T_1, \ldots, T_m)$;
2. $\theta_1 : \theta_2 : T = \theta_2 : \theta_1 : T$.

Preuve: Pour prouver la propriété 1 on utilise les axiomes d'associativité **PCom2** et **Par2**, l'axiome **PCom3**, et la règle d'inférence de contexte du système de la spécification équationnelle définie par **A**:

$$
\begin{aligned}
[\emptyset, \cdot](T_1, \ldots, T_m) &\overset{\textbf{PCom2}}{=} [\emptyset, \cdot](T_1, [\emptyset, \cdot](T_2, \ldots, T_m)) \\
&\overset{\textbf{PCom2, contexte}}{=} [\emptyset, \cdot](T_1, [\emptyset, \cdot](T_2, \ldots, [\emptyset, \cdot](T_{m-1}, T_m)) \\
&\vdots \\
&\overset{\textbf{PCom3, contexte}}{=} [\emptyset, \cdot](T_1, [\emptyset, \cdot](T_2, \ldots, \|(T_{m-1}, T_m)) \\
&\overset{\textbf{PCom3, contexte}}{=} \|(T_1, \|(T_2, \ldots, \|(T_{m-1}, T_m))) \\
&\overset{\textbf{Par3, contexte}}{=} \|(T_1, \ldots, T_m).
\end{aligned}
$$

La propriété 2 est une conséquence immédiate de l'axiome **cons1**. ■

9.2.5 Cohérence de A par rapport à $\approx$

Dans cette section nous présentons le théorème 9.6 qui affirme que le système d'axiomes **A** est cohérent par rapport à la bisimulation–ACTC $\approx$.

Notons la relation d'équivalence syntaxique induite par **A** comme

$$= \overset{\text{def}}{=} \{(T, S) \in \mathcal{L} \times \mathcal{L} \mid \mathbf{A} \vdash T = S\}.$$

Théorème 9.6 Théorème de cohérence: *Le système d'axiomes* **A** *est cohérent avec la relation d'équivalence de termes* $\approx$. *Formellement, pour toute paire de termes* $T, S \in \mathcal{L}$,

$$T \equiv S \Rightarrow T \approx S.$$

Preuve: Pour prouver le théorème de cohérence 9.6, nous devons d'abord prouver que les règles d'inférence de la logique équationnelle s'appliquent à la relation $\approx$. Remarquons d'abord que $\mathcal{L}$ est un langage clos, donc il suffit de montrer que $\approx$ a les propriétés d'équivalence et satisfait à la règle du contexte. Formellement, nous devons montrer que

1. pour tous $T_1, T_2, T_3 \in \mathcal{L}$,
   - *réflexivité*: $T_1 \approx T_1$;
   - *symétrie*: $T_1 \approx T_2$ implique $T_2 \approx T_1$;
   - *transitivité*: $T_1 \approx T_2$ et $T_2 \approx T_3$ implique $T_1 \approx T_3$;
2. pour tout $n \geq 1$ et tous $T_1, \ldots, T_n, S_1, \ldots, S_n \in \mathcal{L}$ et tout opérateur $f \in \mathrm{Op}$

$$(\forall i : 1 \leq i \leq n)(T_i \approx S_i), \text{ implique } f[T_1, \ldots, T_n] = f[S_1, \ldots, S_n].$$

Or, selon le théorème 8.1, $\approx$ est une relation d'équivalence de termes et selon la proposition 8.10, $\approx$ est une congruence pour tous les opérateurs de Op. Donc, $\approx$ satisfait aux règles d'inférence de la logique équationnelle.

Il reste à prouver maintenant que tout axiome de **A** reste vrai si on remplace $=$ par $\approx$. Ces preuves seront présentées en annexe, section 12.4.2. ∎

Puisque $T \approx S \Rightarrow T \sim S$, on prouve implicitement que **A** est cohérent avec la relation d'équivalence de traces $\sim$.

Corollaire 9.7 *Le système d'axiomes* **A** *est cohérent avec la relation d'égalité de traces* $\sim \subseteq \mathcal{L} \times \mathcal{L}$. *Formellement, pour toute paire de termes* $T, S \in \mathcal{L}$,

$$T \equiv S \Rightarrow T \sim S.$$

Nous avons donné une axiomatisation de l'algèbre ACTC qui est cohérente avec la relation d'équivalence sémantique $\approx$, mais incomplète. Autrement dit, à partir du système de spécification équationnelle défini par l'axiomatisation **A** et les règles d'inférence de la logique équationnelle R_E définies dans la section 4.3, nous pouvons déduire uniquement des équivalences de termes $T \approx S$, mais nous ne pouvons pas toutes les déduire. Dans le chapitre suivant nous donnons une procédure de décision de l'équivalence $T \approx S$ de termes quelconques du sous-langage de termes feuilles réactifs $\mathcal{L}^r_\beta$.

Chapitre 10

Décidabilité et complétude

Dans ce chapitre nous prouvons la décidabilité des relations d'équivalence $\sim$, $\simeq$, et $\approx$ sur le sous-langage de termes feuilles réactifs, noté $\mathcal{L}^r_\beta$.

Nous commençons, dans la section 10.1, par la présentation d'une méthode de type *model-checking* pour vérifier si un terme feuille quelconque est réactif.

Notons l'ensemble de termes formés à partir des opérateurs de $(\mathrm{Op}_\beta \cup \{+\}) \setminus \{\mathrm{Max}, \mathrm{Min}\}$ par $\mathcal{L}^{\pm}_\beta$. Dans la section 10.2.1 nous prouvons la décidabilité de $\approx$ sur le sous-langage $\mathcal{L}^{\pm}_\beta$. Ensuite, dans la section 10.2.4, nous prouvons que pour tout terme $T \in \mathcal{L}^r_\beta$ il existe un terme noté $\mathcal{FN}(T) \in \mathcal{L}^{\pm}_\beta$ tel que $T \sim \mathcal{FN}(T)$. Finalement, la section 10.2.5 résume la procédure de décision de $\approx$ sur $\mathcal{L}^r_\beta$.

Enfin, dans la section 10.3, nous abordons la question de la complétude de $\mathbf{A}_\beta$. $\mathbf{A}_\beta$ n'est pas complet par rapport à $\approx$, Par contre, en ajoutant à R_E la règle de déduction supplémentaire suivante

$$(\forall T, S \in \mathcal{L}^r_\beta)(\Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S)) \ \wedge \ \alpha(T) = \alpha(S) \ \wedge \ \Gamma(T) = \Gamma(S) \ \Rightarrow \ T = S)$$

nous obtenons un nouveau système de déduction équationnelle, noté $\mathcal{E}^+(\mathcal{L}^r_\beta, \mathbf{A}_\beta, R^+_E)$, qui est alors cohérent et complet par rapport à la relation $\approx$.

10.1 Réactivité

Nous avons défini un terme *réactif* de $\mathcal{L}$ comme un terme tel que aucune trace $\mu \in \mathbb{T}(T)$ ne mêne vers l'état de blocage $\downarrow$ (section 6.1).

Un terme feuille $T \in \mathcal{L}_\beta$ peut se trouver dans une situation de blocage si T est *incohérent* ou bien si T est *non-causal*. Rappelons les définitions de cohérence et de causalité d'une spécification d'interface, en général, données dans la section 3.2:

- *Cohérence*: la cohérence (en anglais *consistency*) [41, 28, 11] est une condition minimale pour la réalisabilité d'une spécification. Une spécification est *cohérente* si le système de contraintes temporelles a une solution, autrement dit s'il existe une affectation des temps d'occurrence des actions qui satisfait à toutes les contraintes de la spécification.

- *Causalité*: une spécification d'interface est *causale* [32] si le temps d'occurrence de toute action de sortie peut être décidé sans tenir compte des actions d'entrée pouvant être exécutées dans le futur par l'environnement.

Dans le cas d'une spécification d'interface donnée par un terme feuille $T \in \mathcal{L}_\beta$ ces définitions se traduisent comme suit:

1. T est *incohérent* si $\Gamma(T) = \emptyset$.

2. un terme T est *causal* si pour toute contrainte réactive $\mathrm{ROp}(o, H, [m, M], \cdot)$ de T, l'action source $o(t_o)$ peut être exécutée n'importe où à l'intérieur de son intervalle réactif

$$t_{\text{source}} + m \leq v \leq t_{\text{source}} + M,$$

sans contredire les contraintes descriptives de T pouvant relier l'action o à des actions d'entrée de T, qui la précèdent. On note ici par t_{source} le temps d'exécution de l'action source de ROp, c'est à dire la première, respectivement la dernière action de H, si $\mathrm{ROp} = \mathrm{Min}$, respectivement $\mathrm{ROp} = \mathrm{Max}$.

Remarquons que l'incohérence d'un terme peut être détectée dès le démarrage du terme. Pour vérifier la causalité d'un terme, par contre, on doit utiliser des techniques de *model-checking*, c'est à dire parcourir tout l'espace d'états du graphe de processus associé au terme en vérifiant si des propriétés spécifiques sont vérifiées à chaque état.

Puisque les termes de $\mathcal{L}$ décrivent des STEI temporisés avec un domaine temporel dense, il est impossible d'énumérer toutes les traces d'un terme. Donc, pour vérifier la causalité nous devons *compacter* le graphe de processus graphe(T) d'un terme $T \in \mathcal{L}_\beta$ quelconque en faisant abstraction des temps d'exécution des actions.

On associe à un terme $T \in \mathcal{L}_\beta$ un modèle, appelé *graphe de processus compacté*, noté graphe$^*(T)$, qui est un STEI dont la relation de transition est étiquetée par des actions (non-temporisées) de A.

Lorsqu'un terme $T \in \mathcal{L}_\beta$ effectue une transition $T \xrightarrow{a(v)} T'$, avec $T' \in \mathcal{L}_\beta$, $a \in A, v \in D$, selon les règles de SOp toute référence au temps t_a dans le terme T' est remplacée dans le terme T' par la constante v. Le STEI graphe$^*(T)$ exécute alors une transition notée

$$T \overset{a}{\mapsto} T'_{t_a^*/v},$$

où $T'_{t_a^*/v}$ désigne le terme T' dont toute référence au temps d'exécution $t_a = v$ dans T' est remplacée par la variable t_a^*, qui désigne le temps d'*occurrence dans le passé* de l'action a. La variable temporelle t_a^* est appelé une *variable temporelle de référence.* Nous notons l'ensemble de *variables temporelles de référence* par

$$T^{A^*} \overset{\text{not.}}{=} \{t_a^* \mid a \in A\}$$

et notons par $\mathcal{L}_\beta^*$ le langage de termes $\mathcal{L}_\beta$ enrichi par T^{A^*}. Puisque les variables de référence désignent le temps d'occurrence d'actions qui se sont produites dans le passé, une contrainte temporelle θ d'un terme $\theta : T \in \mathcal{L}_\beta^*$ peut contenir des variables temporelles qui correspondent aux actions de T:

$$\text{Act}(\text{Var}(\theta) \cap T^A) \subseteq \text{Act}(T)$$

et des variables temporelles de référence pour les actions $a \notin \mathrm{Act}(T)$ exécutées par les prédécesseurs de T:

$$\mathrm{Act}(\mathrm{Var}(\theta) \cap T^{A^*}) \cap \mathrm{Act}(T) = \emptyset;$$

Formellement, le graphe de processus compacté graphe$^*(T)$ est un STEI

$$\text{graphe}^*(T) = (\mathbf{S}, A, \mapsto, T),$$

où:

- A est le même ensemble d'actions de $\mathcal{L}_\beta$;
- $\mathbf{S} \subseteq \mathcal{L}^*_\beta$ est le partie de $\mathcal{L}^*_\beta$ atteignable à partir de T;
- la relation de transition $\mapsto$ est telle que

$$T \overset{a}{\mapsto} T^* \;\Leftrightarrow\; (\exists v \in D)(T \overset{a(v)}{\to} T')$$

et $T^* = T'_{t^*_a/v}$.

Considérons l'exemple suivant tiré de [31]. Soit

$$T \overset{\text{def}}{=} \mathrm{Max}(o, \{a, b\}, [5, 5], \theta : \|(a, b, c, o)),$$
$$\theta \overset{\text{def}}{=} (-4 \leq t_a - t_b \leq 2) \wedge (1 \leq t_c - t_b \leq 8) \wedge (2 \leq t_c - t_a \leq 10) \wedge (2 \leq t_c - t_o \leq 3).$$

Alors

$$\text{graphe}^*(T) = (\{T_{11}, T_{12}, T_{21}, T_{22}\}, A, \mapsto, T),$$

où:

$$\mapsto \overset{\text{def}}{=} \{T \overset{a}{\mapsto} T_{11}, T \overset{b}{\mapsto} T_{21}, T_{11} \overset{b}{\mapsto} T_{12}, T_{12} \overset{a}{\mapsto} T_{22}\};$$
$$T_{11} \overset{\text{def}}{=} \mathrm{Max}(o, \{b\}, [5, 5], \theta_{11} : \|(b, c, o));$$
$$\theta_{11} \overset{\text{def}}{=} (-4 \leq t^*_a - t_b \leq 2) \wedge (1 \leq t_c - t_b \leq 8) \wedge (2 \leq t_c - t^*_a \leq 10) \wedge (2 \leq t_c - t_o \leq 3)$$
$$(t^*_a \leq t_b) \wedge (t^*_a \leq t_c) \wedge (t^*_a \leq t_o);$$

$$T_{12} \stackrel{\text{def}}{=} \theta_{12} : \|(c, o));$$

$$\theta_{12} \stackrel{\text{def}}{=} (-4 \leq t_a^* - t_b^* \leq 2) \wedge (1 \leq t_c - t_b^* \leq 8) \wedge (2 \leq t_c - t_a^* \leq 10) \wedge (2 \leq t_c - t_o \leq 3)$$
$$(t_a^* \leq t_b^*) \wedge (t_b^* \leq t_c) \wedge (t_b^* \leq t_o) \wedge (5 \leq t_o - t_b^* \leq 5);$$

$$T_{21} \stackrel{\text{def}}{=} \text{Max}(o, \{a\}, [5, 5], \theta_{21} : \|(a, c, o));$$

$$\theta_{21} \stackrel{\text{def}}{=} (-4 \leq t_a - t_b^* \leq 2) \wedge (1 \leq t_c - t_b^* \leq 8) \wedge (2 \leq t_c - t_a \leq 10) \wedge (2 \leq t_c - t_o \leq 3)$$
$$(t_b^* \leq t_a) \wedge (t_b^* \leq t_c) \wedge (t_b^* \leq t_o);$$

$$T_{22} \stackrel{\text{def}}{=} \theta_{22} : \|(c, o));$$

$$\theta_{12} \stackrel{\text{def}}{=} (-4 \leq t_a^* - t_b^* \leq 2) \wedge (1 \leq t_c - t_b^* \leq 8) \wedge (2 \leq t_c - t_a^* \leq 10) \wedge (2 \leq t_c - t_o \leq 3)$$
$$(t_b^* \leq t_a^*) \wedge (t_a^* \leq t_c) \wedge (t_a^* \leq t_o) \wedge (5 \leq t_o - t_a^* \leq 5).$$

Cette définition est basée sur le lemme 6.1 et la proposition suivante:

Proposition 10.1 *Un terme quelconque $T \in \mathcal{L}_\beta$ est réactif si et seulement si son graphe de processus compacté* graphe$^*(\mathbf{S}, A, \mapsto, T)$ *est tel que:*

1. *S'il existe un état $S \in \mathbf{S}$ tel que $S \not\mapsto$, alors S ne contient aucun opérateur réactif;*

2. *Pour toute transition $S \stackrel{a}{\mapsto} S'$, avec $S, S' \in \mathbf{S}$, le deux conditions suivantes sont satisfaites:*

   (a) $\Gamma(S') \neq \emptyset$ *et*

   (b) $\text{Prj}(\Gamma(S'), \text{Var}^*(S')) = \text{Prj}(\Gamma(S)_{t_a^*/t_a} \cap \text{Sol}(\wedge_{b \in \text{A}(S)} t_a^* \leq t_b),\ \text{Var}^*(S') \cup \{t_a^*\})$.

Preuve: La procédure de vérification de réactivité d'un terme $T \in \mathcal{L}_\beta$ que nous avons présentée est fortement inspirée de [31]. Dans [31] on donne une méthode de vérification de la réactivité d'un terme feuille quelconque et on prouve l'exactitude de la méthode. L'équivalence des deux méthodes suit du lemme 6.1 et de la remarque suivante: pour tout terme $T \in \mathcal{L}_\beta$ on a que $\text{Act}(T) = \text{Act}(\Gamma(T))$. La preuve de cette remarque suit directement des définitions des fonctions Act et Γ par induction structurelle. ∎

Dans notre exemple nous avons que

$$\mathrm{Prj}(\theta_{12}, \{t_b^*, t_a^*\}) = -3 \leq t_a^* - t_b^* \leq 0 \text{ et} \tag{10.1}$$

$$\mathrm{Prj}((\theta_{11} \wedge t_b \leq t_o)_{t_b^*/t_b}, \{t_b^*, t_a^*\}) = -4 \leq t_a^* - t_b^* \leq 0. \tag{10.2}$$

On déduit que la transition $T_{11} \overset{b}{\mapsto} T_{12}$ ne satisfait pas à la condition (2) et que le terme T n'est, donc, pas réactif. Effectivement, si les actions a et b sont telles que

$$3 \leq t_b - t_a \leq 3,$$

alors il n'existe plus aucun temps d'exécution valide pour l'action o.

10.2 Décidabilité

Les étapes de la preuve de décidabilité de $\approx$ sur $\mathcal{L}_\beta^r$ sont les suivantes:

1. nous prouvons dans la section 10.2.1 que pour tous $T, S \in \mathcal{L}_\beta^\pm$,

$$T \sim S \Leftrightarrow \Gamma(T) = \Gamma(S);$$

2. nous prouvons (théorème 10.11) que pour tout terme $T \in \mathcal{L}_\beta^r$ il existe un terme appelé *forme normale* de T et noté par $\mathcal{FN}(T)$, tel que

$$T \sim \mathcal{FN}(T) \ \wedge \ \mathcal{FN}(T) \in \mathcal{L}_\beta^\pm$$

3. finalement, dans le théorème 10.12 nous montrons que pour tous termes $T, S \in \mathcal{L}_\beta^r$

   (a) $T \sim S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S))$;

   (b) $T \simeq S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S))$;

   (c) $T \approx S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S)) \ \wedge \ \Gamma(T) = \Gamma(S) \ \wedge \ \alpha(T) = \alpha(S)$;

10.2.1 Décidabilité de $\sim$ sur $\mathcal{L}_\beta^\pm$

Dans cette section nous prouvons que la relation d'équivalence de traces $\sim$ est décidable sur $\mathcal{L}_\beta^\pm$. Nous commençons par le lemme suivant, selon lequel l'information temporelle de tout successeur de T, noté T', est plus stricte que celle de T. Autrement dit, à partir de T' on ne peut effectuer aucune transition qui contredit la fonction $\Gamma(T)$:

Lemme 10.2 *Pour tous* $T, T' \in \mathcal{L}_\beta, a \in A \setminus \{\delta\}, v \geq V_0$,

$$T \xrightarrow{a(v)} T' \Rightarrow \Gamma(T') \cap [t_a = v] \subseteq \Gamma(T).$$

Preuve: On utilise la proposition 6.1, selon laquelle

$$\mathrm{Act}(T') = \mathrm{Act}(T) \setminus \{a\}.$$

On peut, donc, appliquer l'égalité 5 de la proposition 5.1 et déduire que

$$\begin{aligned} \Gamma(T') \cap [t_a = v] &\subseteq \Gamma(T) \Leftrightarrow \\ \mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) &\subseteq \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')) \wedge \\ \mathrm{Prj}([t_a = v], \{a\}) &\subseteq \mathrm{Prj}(\Gamma(T), \{a\}). \end{aligned}$$

Or, la dernière de ces inclusions à déjà été prouvée par la proposition 7.10. Donc, il suffit de prouver que pour toute transition $T \xrightarrow{a(v)} T'$,

$$\mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) \subseteq \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')).$$

La preuve sera faite par induction structurelle et est présenté en annexe, section 12.5.1. ∎

La proposition suivante étend la propriété énoncée dans le lemme 10.2 à l'exécution de traces d'actions: le vecteur des valeurs temporelles de toute trace exécutée par un terme feuille T doit satisfaire l'information temporelle $\Gamma(T)$:

Proposition 10.3 *Soient* $T, T' \in \mathcal{L}_\beta$ *et* $\mu = a_1(v_1) \ldots a_n(v_n) \in \mathbb{T}(T)$ *une trace de* T *telle que* $T \xrightarrow{\mu} T'$. *Alors*

$$\Gamma(T') \cap [\bigwedge_{i=1}^{n} t_{a_i} = v_i)] \subseteq \Gamma(T).$$

Preuve: La preuve est faite par induction sur $n = |\mu|$. La base de l'induction est le cas $n = 1$ et suit directement du lemme 10.2.

Prouvons le pas d'induction. Notons par $T_i \in \mathcal{L}_\beta$ les termes intermédiaires, tels que

$$T \overset{a_1(v_1)}{\rightarrow} T_1 \ldots T_{n-1} \overset{a_n(v_n)}{\rightarrow} T'.$$

Alors, selon l'hypothèse d'induction:

$$\Gamma(T_{n-1}) \cap \mathrm{Sol}(\bigwedge_{i=1}^{n-1} t_{a_i} = v_i) \subseteq \Gamma(T). \tag{10.3}$$

On applique le lemme 10.2 à la transition $T_{n-1} \overset{a_n(v_n)}{\rightarrow} T'$ et on obtient que

$$\Gamma(T') \cap \mathrm{Sol}(t_{a_n} = v_n) \subseteq \Gamma(T_{n-1}). \tag{10.4}$$

On combine les équations 10.3 et 10.4 on déduit la relation voulue:

$$\Gamma(T') \cap \mathrm{Sol}(\bigwedge_{i=1}^{n} t_{a_i} = v_i) \subseteq \Gamma(T). \quad \blacksquare$$

En particulier, si

$$\mu = a_1(v_{\sigma(1)}) \ldots a_n(v_{\sigma(n)}) \in \mathbb{T}(T)$$

est une trace *complète*, c'est à dire si $T \overset{\mu}{\rightarrow}\uparrow$, alors

$$\Gamma(\uparrow) \cap \mathrm{Sol}(\bigwedge_{i=1}^{n} t_{a_i} = v_{\sigma(i)}) = [(\bigwedge_{i=1}^{n} t_{a_i} = v_{\sigma(i)}].$$

Donc, selon la proposition 10.3 on obtient que

$$[\bigwedge_{i=1}^{n} t_{a_i} = v_{\sigma(i)}] \in \Gamma(T).$$

Dans la proposition suivante nous prouvons que la réciproque de cette propriété est aussi vraie:

Proposition 10.4 *Soit $T \in \mathcal{L}_\beta$ un terme qui ne contient aucun opérateur réactif. Soit*

$$\vec{v} = (v_1, \ldots, v_n) \in \Gamma(T)$$

tel que $\mathrm{Act}(\vec{v}) = \mathrm{Act}(T)$ *un vecteur d'instants temporels qui satisfait aux contraintes temporelles de* T. *Soit* $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ *une permutation des indices telle que les valeurs temporelles permutées de* $\vec{v}$ *sont en ordre croissant:*

$$v_{\sigma(i)} \leq v_{\sigma(i+1)},$$

pour tout $i : 1 \leq i \leq n-1$. *Notons l'action correspondante à la valeur temporelle* $v_{\sigma(i)}$ *par* $a_{\sigma(i)} \in \mathrm{Act}(T)$. *Alors il existe une trace* $\mu \in \mathbb{T}(T)$ *telle que*

$$\mu = a_{\sigma(1)}(v_{\sigma(1)}) \ldots a_{\sigma(n)}(v_{\sigma(n)}).$$

Preuve: voir section 12.5.2 de l'annexe. ∎

On remarque que dans la proposition précédente, μ est une trace complète de T, c'est à dire qui contient toutes les actions de T. Donc, par la proposition 6.1 on déduit que

$$T \xrightarrow{\mu} \uparrow .$$

Basée sur la proposition 10.4 et le corollaire 10.3, le théorème 10.5 montre qu'une trace complète d'un terme feuille T qui ne contient aucun opérateur réactif est exécutable par le terme T si et seulement si elle satisfait la contrainte $\Gamma(T)$. Remarquons que l'implication à droite $\Rightarrow$ est vraie même pour des termes contenant des opérateurs réactifs (proposition 10.3).

Théorème 10.5 *Soit* $T \in \mathcal{L}_\beta$ *un terme feuille qui ne contient aucun opérateur réactif et soit* $\vec{v} = \langle v_1, \ldots, v_n \rangle \in D^{A^*}$ *un vecteur temporel tel que* $\mathrm{Act}(\vec{v}) = \mathrm{Act}(T)$. *Soit* $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ *une permutation des indices* i *telle que les valeurs temporelles permutées de* $\vec{v}$ *sont en ordre croissant:*

$$v_{\sigma(i)} \leq v_{\sigma(i+1)},$$

pour tout $i : 1 \leq i \leq n-1$. *Alors*

$$\vec{v} \in \Gamma(T)$$

si et seulement si il existe une trace $\mu \in \mathbb{T}(T)$ *telle que*

$$\mu = a_{\sigma(1)}(v_{\sigma(1)}) \ldots a_{\sigma(n)}(v_{\sigma(n)}).$$

Preuve: la preuve suit directement des propositions 10.4 et 10.3. ∎

Le théorème suivant est une conséquence du théorème 10.5. On énonce deux résultats importants: on montre qu'une condition nécessaire et suffisante pour que deux termes feuilles sans opérateurs réactifs soient équivalents du points de vue de leurs traces est que leurs espaces de contraintes temporelles Γ aient exactement les mêmes solutions. Ensuite, on étend cette propriété aux sommations de termes feuilles sans opérateurs réactifs, c'est à dire à tous les termes de $\mathcal{L}_{\beta}^{\pm}$.

Théorème 10.6 *Soient* T *et* S *des termes quelconques de* $\mathcal{L}_{\beta}^{\pm}$. *Alors*

$$T \sim S \Leftrightarrow \Gamma(T) = \Gamma(S).$$

Preuve: Si $T, S \in \mathcal{L}_{\beta}$ sont des termes feuille sans opérateurs réactifs, alors le théorème est un corollaire du théorème 10.5.

Il reste à considérer des paires $(T, S) \in \mathcal{L}_{\beta}^{\pm}$ tels que

$$T = \sum_{i=1}^{m}(T_i) \text{ et } S = \sum_{i=1}^{m}(S_i),$$

avec $T_i, S_i \in \mathcal{L}_{\beta}^{\pm}$. La preuve se fait par induction structurelle: on suppose que tous les sous-termes T_i, S_i ont la propriété énoncée et on prouve qu'alors T et S aussi l'ont. La base de l'induction est donnée par les termes feuilles qui ne contiennent aucun opérateur réactif; le théorème suit alors directement de la proposition 10.5.

Prouvons le pas d'induction. Premièrement, selon la proposition 4.4, on a que

$$\mathbb{T}(T) = \bigcup_{i=1}^{m} \mathbb{T}(T_i), \tag{10.5}$$

$$\mathbb{T}(S) = \bigcup_{i=1}^{m} \mathbb{T}(S_i). \tag{10.6}$$

En plus, par définition de la fonction Γ

$$\begin{aligned}\Gamma(T) &= \bigcup_{i=1}^{m} \Gamma(T_i),\\ \Gamma(S) &= \bigcup_{i=1}^{m} \Gamma(S_i).\end{aligned}$$

Supposons d'abord que $T \sim S$ et soit $\vec{v}$ un vecteur quelconque de valeurs temporelles. Nous prouverons que

$$\vec{v} \in \Gamma(T) \Leftrightarrow \vec{v} \in \Gamma(S).$$

Notons par $\mu(\vec{v})$ la séquence d'actions temporisées correspondante à $\vec{v}$. Alors,

$$\begin{aligned}\vec{v} \in \Gamma(T) &\overset{10.7}{\Leftrightarrow} (\exists i \in \{1,\ldots,m\})(\vec{v} \in \Gamma(T_i))\\ &\overset{\text{hyp.Ind.}}{\Leftrightarrow} \mu(\vec{v}) \in \mathbb{T}(T_i)\\ &\overset{10.5}{\Rightarrow} \mu(\vec{v}) \in \mathbb{T}(T)\\ &\overset{\text{supp.}}{\Leftrightarrow} \mu(\vec{v}) \in \mathbb{T}(S)\\ &\overset{10.6}{\Leftrightarrow} (\exists j \in \{1,\ldots,m\})(\mu(\vec{v}) \in \mathbb{T}(S_j))\\ &\overset{\text{hyp.Ind.}}{\Leftrightarrow} \vec{v} \in \Gamma(S_j)\\ &\overset{10.7}{\Rightarrow} \vec{v} \in \Gamma(S).\end{aligned}$$

Nous avons, donc, prouvé que $T \sim S$ implique que $\Gamma(T) \subseteq \Gamma(S)$. L'implication de l'inclusion symétrique est analogue. Donc,

$$T \sim S \Rightarrow \Gamma(T) = \Gamma(S).$$

Prouvons maintenant que si $\Gamma(T) = \Gamma(S)$, alors toute trace $\mu \in \mathbb{T}(T)$ est aussi une trace de S et vice-versa. Soit $\mu \in \mathbb{T}(T)$ une trace quelconque de T et notons par $\vec{\mu}$ le vecteur de valeurs temporelles correspondant à μ. Alors,

$$\begin{aligned}\mu \in \mathbb{T}(T) &\overset{10.5}{\Leftrightarrow} (\exists j \in \{1,\ldots,m\})(\mu \in \mathbb{T}(T_j))\\ &\overset{\text{hyp.Ind}}{\Leftrightarrow} \vec{\mu} \in \Gamma(T_j)\end{aligned}$$

$$\begin{aligned}
&\overset{10.7}{\Rightarrow} && \vec{\mu} \in \Gamma(T) \\
&\overset{\text{supp.}}{\Leftrightarrow} && \vec{\mu} \in \Gamma(S) \\
&\overset{10.7}{\Leftrightarrow} && (\exists i \in \{1, \ldots, m\})(\vec{\mu}) \in \Gamma(S_i)) \\
&\overset{\text{hyp.Ind}}{\Leftrightarrow} && \vec{\mu} \in \mathbb{T}(S_i) \\
&\overset{10.6}{\Leftrightarrow} && \vec{\mu} \in \mathbb{T}(S),
\end{aligned}$$

Nous pouvons alors déduire que

$$T \sim S \Rightarrow \mathbb{T}(T) \subseteq \mathbb{T}(S).$$

La preuve de l'inclusion inverse est analogue. Ceci conclut la preuve du théorème. ∎

Nous avons ainsi prouvé la décidabilité de $\sim$ sur $\mathcal{L}_\beta^\pm$. Dans la prochaine section nous nous servirons de ce résultat pour prouver la décidabilité de $\approx$ sur le sous-langage de termes feuilles réactifs.

10.2.2 Décidabilité de $\approx$ sur $\mathcal{L}_\beta^r$

Dans cette section nous présentons la procédure de décision de l'équivalence $T \approx S$ de termes réactifs, $T, S \in \mathcal{L}_\beta^r$. L'idée principale de cette procédure est la transformation d'un terme réactif $T \in \mathcal{L}_\beta^r$ quelconque dans un *format normalisé* $\mathcal{FN}(T) \in \mathcal{L}_\beta^\pm$ tel que $T \sim \mathcal{FN}(T)$. La première étape de la transformation de T dans une forme normale est la transformation dans un format intermédiare, noté $\mathcal{FI}(T)$, tel que $T \approx \mathcal{FI}(T)$ et $\mathcal{FI}(T) \in \mathcal{L}_\beta^r$.

10.2.3 Format intermédiaire

Définition 10.1 [Format intermédiaire] *Soit T un terme feuille de $\mathcal{L}_\beta$ qui contient n opérateurs réactifs et dont les actions sont* $\mathrm{Act}(T) = \{a_1, ..., a_m\}$. *$T$ est en format intermédiaire s'il est le terme inactif $T = \uparrow$ ou bien si T est composé comme suit:*

$$T \overset{\text{def}}{=} \nabla_\alpha(\mathrm{ROp}(o_1, H_1, [m_1 M_1], T_1))$$

$$\vdots$$

$$T_n \stackrel{\text{def}}{=} \mathrm{ROp}(o_n, H_n, [m_n M_n], T_{n+1})$$

$$T_{n+1} \stackrel{\text{def}}{=} \theta : \|(a_1, \ldots, a_m),$$

où ∇ *peut être n'importe lequel des opérateurs de décalage* $\trianglelefteq$ *ou* $\triangleleft$ *et* ROp *est un opérateur réactif quelconque.*

Le sous-terme $T_{n+1} = \theta : \|(a_1, \ldots, a_m)$ est appelé le *noyau* de T. Tout terme qui, comme T_{n+1}, est une composition parallèle d'actions, contraint par une seule contrainte θ tout au plus, est dit un *terme simple.*

Pour montrer que tout terme feuille a un format intermédiaire bisimilaire, nous avons besoin d'abord du lemme 10.7. Selon ce lemme on peut remplacer tout port $a_1(t_1).a_2(t_2)\ldots a_m(t_m)$ par la composition parallèle des actions a_i, contrainte par un opérateur θ, qui exige qu'une action a_i se produise avant l'action a_{i+1}.

Lemme 10.7 *Soit* $T = a_1(t_1).a_2(t_2)\ldots a_m(t_m) \in \mathcal{L}_\beta$, *où* $a_i \in A \setminus \{\delta\}$, *avec* $i = 1, \ldots, m$. *Alors,*

$$T \approx \bigwedge_{i=1}^{i=m-1} t_i < t_{i+1} : \|(a_1, a_2, \ldots, a_m).$$

Preuve: L'axiome **Pref1** est cohérent (théorème 9.6). Donc,

$$T \approx \bigwedge_{b \in \mathrm{Act}(T_1)} t_a < t_b : a(t_a)\|S.$$

pour tout terme $S \in \mathcal{L}_\beta$. On suit ensuite un raisonnement inductif sur m, le nombre d'actions de T. Ce raisonnement sera présenté en détail en annexe, section 12.5.3. Notons ici qu'il suffit d'appliquer la propriété de distribution de θ par rapport à la composition parallèle (axiome **cons4** et le théorème de cohérence 9.6), ainsi que la congruence de $\approx$ par rapport aux opérateurs de Op_β (théorème 8.10). ∎

Théorème 10.8 Théorème de la forme intermédiaire *Tout terme feuille* $T = \mathcal{L}_\beta$ *peut être transformé dans une forme intermédiaire* $\mathcal{FI}(T)$ *bisimilaire, c'est à dire telle que* $T \approx \mathcal{FI}(T)$.

Preuve: Si $T = \uparrow$ alors $\mathcal{FI}(T) = T$, donc la preuve est triviale. Soit $T \in \mathcal{L}_\beta \setminus \{\uparrow\}$.

La première étape vers la transformation de T dans une forme intermédiaire consiste à obtenir un terme T_1 équivalent à T, tel que T_1 est une composition parallèle de deux ou plusieurs ports, contrainte par plusieurs ou aucun d'opérateur θ, par zéro ou plus d'opérateurs réactifs ROp, et décalée par aucun ou plusieurs opérateurs $\trianglelefteq$.:

$$\begin{aligned} T_1 &= P \mid \theta : T_1 \mid \mathrm{ROp}(T_1) \mid \trianglelefteq_v(T_1) \\ P &= P' \mid \|(P, P') \\ P' &= a(t_a) \mid a(t_a).P'. \end{aligned}$$

On peut faire cette transformation, parce que nous avons déjà prouvé que les opérateurs θ, $\trianglelefteq$, et des opérateurs réactifs ROp sont distributifs par rapport à la composition parallèle $\|$ (théorème de cohérence 9.6). En plus, selon l'associativité de $\|$ (théorème 9.6), on peut écrire P comme un seul port, ou bien comme la composition de plusieurs ports:

$$P = P' \text{ ou bien } P \approx \|(P_1, \ldots, P_m).$$

On procède alors au regroupement de tous les opérateurs de décalage à l'extérieur de T_1, suivis par les opérateurs réactifs, et finalement suivis par tous les opérateurs θ. On utilise ici le fait que les opérateurs θ, $\trianglelefteq$, et les opérateurs réactifs sont distributifs les uns par rapport aux autres (théorème 9.6, sections 12.4.2.3, 12.4.2.11, et 12.4.2.3). On obtient un terme $T_2 \approx T_1 \approx T$, tel que

$$\begin{aligned} T_2 &= S \mid \trianglelefteq_v(T_2) \\ S &= S' \mid ROp(S) \\ S' &= P \mid \theta : P. \end{aligned}$$

Si T est un terme initial, alors, par définition il ne contient aucun opérateur de décalage. Si T n'est pas un terme initial de $\mathcal{L}_0$, alors, selon la proposition 7.3, il existe une trace d'actions

$$\mu = b_1(t_1) \ldots b_n(t_n)$$

et un terme $T_0 \in \mathcal{L}_0$ tels que

$$T_0 \xrightarrow{\mu} T.$$

On a alors que $\alpha = t_n$. Pour chacune des actions $b_i(t_i)$ de la trace un opérateur de décalage $\trianglelefteq_{t_i}$ est introduit dans T. Selon le théorème de l'avancement du temps (Corollaire 7.8), $t_i \leq t_{i+1} \leq t_n = \alpha(T)$, pour tout opérateur $\trianglelefteq_{t_i}$ de T'. Notons $\alpha = \alpha(T)$. Donc, puisque l'axiome 12.4.2.1 est cohérent, on peut remplacer tous les opérateurs de décalage $\trianglelefteq_{t_i}$ par $\trianglelefteq_\alpha$.

$$T_2 \approx T_3 = \trianglelefteq_\alpha(S).$$

Ensuite, selon la proposition 12.4.2.7, on peut remplacer toutes les contraintes θ de S par leur conjonction, notée $\theta_\wedge$. On obtient alors un terme $T_4 \approx T$, tel que

$$T_4 = \trianglelefteq_\alpha(\mathrm{ROp}_1(\ldots(\mathrm{ROp}_n(\theta_\wedge : P)\ldots)),$$

où $n \geq 0$.

La prochaine étape consiste à remplacer le préfixage d'actions par des compositions parallèles d'actions, contraints par des opérateurs θ. Plus précisément, selon le Lemme 10.7, chaque port P_i peut être remplacé par le terme équivalent suivant:

$$P_i \approx \bigwedge_{j=1}^{i=n_i-1} t_{a^i_j} < t_{a^i_{j+1}} : \|(a^i_1, a^i_2, \ldots, a^i_{n_i}).$$

Notons par θ_i la contrainte temporelle et par P'_i la composition parallèle de ce terme. Étant donné que les ensembles d'actions de ports P_i sont disjoints, que $\mathrm{Act}(\theta_i) \subseteq \mathrm{Act}(T_i)$, on peut appliquer la propriété distributivité restreinte de θ par rapport à l'opérateur $\|$ (voir preuve dans la section 12.4.2.10) et déduire que

$$P \approx \bigwedge_{i=1}^{n} \theta_i : \|(P'_1, \ldots, P'_n).$$

Puisque l'opérateur de composition parallèle $\|$ est associatif (théorème 9.6, section 12.4.2.6), on déduit que

$$P \approx \bigwedge_{i=1}^{n} \theta_i : \|(a^1_1, \ldots, a^1_{n_1}, \ldots a^n_1, \ldots, a^n_{n_n}).$$

Finalement, selon la proposition 12.4.2.7, on note

$$\theta_{FI} = \theta_{\wedge} \bigwedge_i \theta_i,$$

et on obtient que

$$T \approx \mathcal{FI}(T) = \trianglelefteq_{\alpha}(ROp_1(\ldots ROp_n(\theta_{FI} : \|(a_1^1, \ldots, a_{n_1}^1, \ldots a_1^n, \ldots, a_{n_n}^n).$$

Remarquons que, dans cette preuve, nous avons pu remplacer un sous-terme par un autre terme équivalent, parce que la relation $\approx$ est une congruence par rapport aux opérateurs de Op (théorème 8.10). ∎

10.2.4 Forme normale

Dans cette section nous montrons comment remplacer un opérateur réactif par une sommation de termes contraints linéairement. Nous obtenons la *forme normale* d'un terme réactif $T \in \mathcal{L}_{\beta}^{r}$, notée $\mathcal{FN}(T) \in \mathcal{L}_{\beta}^{+} \setminus \{\text{ROp}\}$ et qui est telle que $\mathbb{T}(T) = \mathbb{T}(\mathcal{FN}(T))$.

Définition 10.2 (*Forme normale*) *Un terme de $T \in \mathcal{L}_{\beta}^{+}$ est en format normal s'il est de la forme*

$$\mathcal{FN} = \trianglelefteq_{\alpha}(+(T_1, \ldots, T_n)),$$

où chaque sous-terme est de la forme

$$T_i = \theta^i : \|(a_1^i, \ldots a_{m_i}^i)$$

et a_j^i sont des actions de A_{β}.

La figure 10.1 présente les idées de base de la transformation en forme normale d'un terme qui est en format intermédiaire.

La première étape, la transformation d'un terme T en un format intermédiaire $\mathcal{FI}(T)$, tel que $T \approx \mathcal{FI}(T)$, a été présentée en détail dans la Section 10.2.3. Le Lemme 10.10 montre comment remplacer un opérateur réactif quelconque ROp =

Entrée: deux termes-feuille réactifs T_1, T_2

Transcrire chaque terme T_i dans sa forme normale $\mathcal{NF}_i$ comme suit:

1. Construire la forme intermédiaire $\mathcal{FI}$
2. Remplacer les opérateurs réactifs par des sommations de termes simples
3. Éliminer les branches non-viables

Figure 10.1: Transformation en forme normale

$(o, H, [m, M], \cdot)$ par une sommation de termes contraints par des opérateurs θ: on choisit une action $a \in H$ parmi l'ensemble d'actions source et suppose que c'est la dernière, respectivement la première action source à être exécutée (si ROp est un opérateur Max, respectivement Min). Pour chacun de ces choix d'action source a on définit ensuite une contrainte temporelle, notée $\theta_{\mathrm{ROp},a} \in \Theta$, qui contraint toutes les autres actions sources à se produire avant, respectivement après, l'action a. L'opérateur ROp est alors remplacé par la sommation sur tous les choix d'actions sources possibles, contraints par leur ordonnancement respectif. Parmi ces ordonnancements il peut y en avoir certaines dont l'espace de solution est vide, c'est à dire qui se contredisent réciproquement ou bien qui contredisent l'information temporelle du terme T. Les branches qui correspondent à ses ordonnancements bloquent au temps de démarrage du terme T. Puisque T est réactif, l'ensemble de branches qui ne bloquent pas (appelées branches *viables*), est non-vide. La forme normale de T est la sommation de toutes ces branches viables. La proposition 10.9 nous permet de remplacer la sommation initiale par la sommation des branches viables seulement.

Soit $T = \sum_{i \in I} T_i$ un terme de $\mathcal{L}^+_\beta$. Selon la sémantique opérationnelle de l'opérateur $+$, le terme T peut choisir d'exécuter n'importe quel de ses sous-termes, à l'exception de ceux qui bloquent pendant que d'autres sous-termes peuvent encore exécuter des actions. On dit que T peut exécuter tout sous-terme *viable*. Formellement, une branche T_i de T est dite *viable* si elle ne bloque pas, ou bien si elle bloque lorsqu'aucune

autre branche ne peut encore exécuter des actions. Nous notons par $\mathrm{V}(T)$ l'ensemble des branches viables de T. Donc,

$$\mathrm{V}(T) = \{T_i \mid i \in I, (T_i \overset{\delta}{\not\rightarrow}) \vee (\forall j \neq i)(T_j \overset{\delta(v)}{\rightarrow}\downarrow \ \vee \neg\mathrm{Wait}(v, T_j))\}.$$

Proposition 10.9 *Soit $T = \sum_{i\in I} T_i$ un terme de $\mathcal{L}$. Alors, la sommation des branches viables de T a les mêmes traces que T. Formellement,*

$$T \sim \sum_{T_i \in \mathrm{V}(T)} T_i.$$

Preuve: il suffit de remarquer que, selon les règles de SOp, il existe une transition

$$T \overset{a(v)}{\rightarrow} T'$$

si et seulement si il existe un sous-terme $T_i \in \mathrm{V}(T)$ tel que

$$T_i \overset{a(v)}{\rightarrow} T'. \quad \blacksquare$$

Lemme 10.10 *Soit $\mathrm{ROp}(o, H, [m, M], T) \in \mathcal{L}^r_\beta$ un terme feuille réactif quelconque, avec $\mathrm{ROp} \in \{\mathrm{Max}, \mathrm{Min}\}$. Alors,*

$$\mathrm{ROp}(o, H, [m, M], T) \sim \sum_{a\in H} \theta_{\mathrm{ROp},a} : T,$$

où nous notons par $\theta_{(a,\mathrm{ROp})} \in \Theta$ la contrainte temporelle suivante:

$$\theta_{(a,\mathrm{Max}(o,H,[m,M],\cdot))} \overset{\mathrm{def}}{=} \bigwedge_{b\in H} t_a \geq t_b \ \wedge \ m \leq t_o - t_a \leq M,$$

$$\theta_{(a,\mathrm{Min}(o,H,[m,M],\cdot))} \overset{\mathrm{def}}{=} \bigwedge_{b\in H} t_a \leq t_b \ \wedge \ m \leq t_o - t_a \leq M.$$

Preuve: Selon le théorème 4.8, deux termes quelconques T et S sont équivalents du point de vue de leurs traces si et seulement si les ensembles $\{T\}$ et $\{S\}$ sont bisimilaires, notation $\{T\} \simeq \{S\}$. Rappelons que, par abus de notation, on dit que $\{T\} \simeq \{S\}$ si et seulement si les $\wp$-STEI associés à T et S sont bisimilaires:

$$\{T\} \simeq \{S\} \overset{\mathrm{not.}}{\Leftrightarrow} \wp(\mathrm{graphe}(T)) \simeq \wp(\mathrm{graphe}(S)).$$

Pour prouver que $\{T\} \simeq \{S\}$ on applique le raisonnement présenté dans la section 8.2, c'est à dire on définit une relation $\mathcal{R} \subseteq \wp(\mathcal{L}) \times \wp(\mathcal{L})$ telle qu'elle contient la paire $(\{T\}, \{S\})$ et telle qu'elle soit une bisimulation forte, c'est à dire telle que $\mathcal{R} \subseteq \simeq$. Dans la section 12.5.4 de l'annexe nous allons faire la preuve pour l'opérateur Min seulement, celle pour l'opérateur Max est analogue. ∎

Finalement, le théorème suivant résume la méthode de transformation d'un terme dans une forme normale équivalente du point de vue des traces.

Théorème 10.11 [Théorème de la forme normale]: *Tout terme feuille réactif $T \in \mathcal{L}_\beta$ a une forme normale $\mathcal{FN}(T)$ équivalente du point de vue des traces.*

Preuve: Selon le théorème, tout terme $T \in \mathcal{L}_\beta$ a une forme intermédiaire $\mathcal{FI}(T)$ bisimilaire. Formellement,

$$T \approx \trianglelefteq_\alpha(\mathrm{ROp}_1(\mathrm{ROp}_2, \ldots \mathrm{ROp}_n(\theta : \|(a_1, \ldots, a_m) \ldots))),$$

où $\mathrm{ROp}_i = \mathrm{Min/Max}(o_i, H_i, [m_i, M_i], \cdot)$ sont des opérateurs réactifs.

Selon le Lemme 10.10, et puisque $\sim$ est une congruence par rapport à l'opérateur $\trianglelefteq$ (proposition 8.11), on a alors que

$$T \sim \trianglelefteq_\alpha(\sum_{b_1 \in H_1} \theta_{(b_1, \mathrm{ROp}_1)} : \mathrm{ROp}_2(\ldots \mathrm{ROp}_n(\theta : \|(a_1, \ldots, a_m))).$$

Puisque les opérateurs réactifs et les contraintes θ sont distributifs (théorème 9.6) et étant donné que $\sim$ est une congruence par rapport à $\sum$ et à $\trianglelefteq$ (proposition 8.11), le terme T peut être récrit comme suit:

$$T \sim \trianglelefteq_\alpha(\sum_{b_1 \in H_1} \mathrm{ROp}_2(\ldots \mathrm{ROp}_n(\theta_{(b_1, \mathrm{ROp}_1)} : (\theta : \|(a_1, \ldots, a_m) \ldots))).$$

Ensuite, les deux contraintes $\theta_{(b_1, \mathrm{ROp}_1)}$ et θ peuvent être remplacées par leur conjonction (cohérence de l'axiom **cons1**, théorème 9.6, section 12.4.2.7):

$$T \sim \trianglelefteq_\alpha(\sum_{a \in H_1} \mathrm{ROp}_2(\ldots \mathrm{ROp}_n(\theta_{(b_1, \mathrm{ROp}_1)} \wedge \theta : \|(a_1, \ldots, a_m) \ldots)).$$

Récursivement, puisque $\sim$ est un congruence par rapport aux l'opérateurs $\sum$ et $\trianglelefteq$ (proposition 8.11), on déduit l'équivalence suivante:

$$T \sim \trianglelefteq_\alpha(\sum_{\substack{b_i \in H_i \\ i=1,\ldots,n}} \bigwedge_{i=1}^{n} \theta_{(b_i,\mathrm{ROp}_i)} \wedge \theta : \|(a_1,\ldots,a_m)),$$

Pour simplifier, on peut alors éliminer toutes le branches non-viables de cette sommation, selon la proposition 10.9 et on déduit que T a une forme normale équivalente du point de vue des traces. ∎

10.2.5 Procédure de décision d'équivalence

Nous pouvons maintenant énoncer le théorème de la décidabilité des relations d'équivalence $\sim$, $\simeq$, et $\approx$ dans le sous-langage $\mathcal{L}^r_\beta$.

Théorème 10.12 *Soient $T, S \in \mathcal{L}^r_\beta$ deux termes réactifs et $\mathcal{FN}(T)$ et $\mathcal{FN}(S)$ leurs formes normales respectives. Alors*

1. $T \sim S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S))$;
2. $T \simeq S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S))$;
3. $T \approx S \Leftrightarrow \Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S)) \wedge \Gamma(T) = \Gamma(S) \wedge \alpha(T) = \alpha(S)$;

Preuve: La preuve de l'équivalence 1 suit directement du théorème 10.6, qui affirme que des sommations de termes feuilles ne contenant aucun opérateur réactif sont équivalentes du point de vue de leurs traces si et seulement si elles ont les mêmes solutions de leur espace temporel Γ.

Or, le langage $\mathcal{L}_\beta$ est déterministe (théorème 6.4). Alors, selon la proposition 4.2, pour tous $T, S \in \mathcal{L}_\beta$,

$$T \sim S \;\Leftrightarrow\; T \simeq S.$$

Ceci nous permet de déduire l'équivalence 2.

Finalement, l'équivalence 3 suit directement de la définition de la bisimulation ACTC $\approx$ et de l'équivalence 2. ∎

10.3 Complétude

Dans le chapitre précédent nous avons défini une axiomatisation **A** de l'algèbre ACTC et prouvé qu'elle définit un système de déduction de termes $\mathcal{E} = \langle \mathbf{A}, R_E \rangle$ selon les règles d'inférence, notées R_E, suivantes:

1. les propriétés d'*équivalence* de $=$:
   pour tous $t, s, r \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   - *réflexivité*: $t = t$
   - *symétrie*: $t = s$ implique $s = t$
   - *transitivité*: $t = s$ et $s = r$ implique $t = r$
2. la règle de *substitution*:
   pour tous $t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et toute substitution $\varphi : \mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow T(\mathcal{F}, \mathcal{X})$,
   $$t_1 = t_2 \text{ implique } \varphi(t_1) = \varphi(t_2);$$
3. la règle de *contexte*:
   pour tout $n \geq 1$ et tous $t_1, \ldots, t_n, s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
   $$(\forall i : 1 \leq i \leq n)(t_i = s_i), \text{ implique } f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n).$$

La spécification équationnelle $\mathcal{E}$ est cohérente avec la relation d'équivalence sémantique $\approx$, mais elle n'est pas complète. Par contre, dans la section précédente, nous avons donné une procédure de décision de la relation $\approx$ sur le sous-ensemble de termes $\mathcal{L}^r_\beta$. Ceci nous amène à définir un nouvel ensemble de règles d'inférence, R^+_E qui contient toutes les règles de R_E plus la règle suivante:

$$(\Gamma(\mathcal{FN}(T)) = \Gamma(\mathcal{FN}(S)) \ \wedge \ \alpha(T) = \alpha(S) \ \wedge \ \Gamma(T) = \Gamma(S)) \ \Rightarrow \ T = S,$$

pour tous $T, S \in \mathcal{L}^r_\beta$. Le nouveau système de déduction équationnelle est alors cohérent et complet par rapport à la relation $\approx$.

Ceci résume la définition de l'algèbre de processus temporisée ACTC et des propriétés syntaxiques et sémantique de cette algèbre. Nous présentons les conclusions de notre travail dans le chapitre suivant.

Chapitre 11

Conclusion

Le problème de la spécification et de la vérification d'interfaces matérielles est intrinsèquement difficile pour deux raisons:

- une interface est un système temps-réel, c'est à dire un système qui doit produire ses résultats à l'intérieur d'intervalles de temps spécifiés par des contraintes temporelles;
- le rôle d'une interface matérielle est d'assurer la communication entre des modules séparés qui doivent être interconnectés, sans toutefois disposer d'informations précises sur le fonctionnement des divers modules; l'interface matérielle doit décrire l'ensemble des hypothèses qu'un module fait sur les propriétés de son environnement, ainsi que les réactions du module face aux actions de l'environnement.

Par conséquent, tout formalisme utilisé pour spécifier des interfaces matérielles doit pouvoir exprimer le dualisme des actions de direction entrée et de direction sortie, ainsi que le paradigme des contraintes temporelles de type supposition et de type engagement: un module garantit que ses actions (de direction sortie) ont des propriétés précises, telles que décrites par les contraintes temporelles de type engagement, en autant que les actions de l'environnement (de direction entrée) respectent les con-

traintes temporelles de type supposition. La spécification d'une interface matérielle est correcte lorsque l'interconnexion de tous modules qui satisfont aux contraintes temporelles spécifées est réalisable.

Vu la complexité de ce problème, même la définition de propriétés d'une spécification d'interface matérielle qui soient suffisantes et/ou néccessaires pour que la spécification soit correcte est non-triviale.

Traditionnellement, dans l'industrie ce problème est contourné par "omission": on se contente de donner des spécifications incomplètes, donnant lieu à des ambiguïtés, et de vérifier l'exactitude par une simulation non-exhaustive. Cerny et al. [32, 54] ont proposé une approche modulaire et hiérarchique, les HAAD, (de l'anglais *Hierarchical Annotated Action Diagrams*) pour simplifier ce problème. Les HAAD sont un formalisme graphique qui permet de représenter des interfaces matérielles par des modules de base, appelés *chronogrammes de base* ou *feuilles*, qui sont ensuite composés avec des opérateurs de composition hiérarchiques.

Les tentatives de formaliser les HAAD en utilisant des méthodes formelles existantes ([20, 21]) se sont heurtées à l'impossibilité d'exprimer le dualisme entrée/sortie et engagement/supposition. Nous avons alors tenté de définir un langage formel de spécification algébrique, spécifiquement conçu pour répondre à ces besoins [22, 31]. Ces efforts ont finalement abouti à la définition de l'algèbre de processus ACTC.

Dans ce document nous donnons pour la première fois la définition complète de l'algèbre de processus temporisée ACTC pour la spécification et la vérification d'interfaces matérielles avec le langage de spécification $\mathcal{L}$, dont la sémantique est complètement définie par un ensemble de règles de sémantique opérationnelle et munie d'une axiomatisation cohérente.

Les particularités du langage $\mathcal{L}$ sont les suivantes:

- contraintes temporelles linéaires de type conjonctif, min, et max de type supposition et/ou engagement;
- support du paradigme supposition/engagements dans la sémantique des divers

opérateurs de composition hiérarchiques;

- la définition d'opérateurs temporisés d'arité arbitraire pour la composition parallèle avec communication et le choix retardé;
- la définition d'opérateurs temporisés d'exception et de composition récursive.

Les principaux résultats présentés dans ce document sont les suivants:

- la preuve de la *congruence* de la relation d'équivalence sémantique de termes $\approx \subseteq \mathcal{L} \times \mathcal{L}$;
- la preuve de la *monotonie temporelle* des séquences d'actions exécutées par un terme;
- la définition d'une axiomatisation $\mathbf{A}$ pour le langage $\mathcal{L}$ et la preuve de la cohérence de $\mathbf{A}$ sur $\mathcal{L}$;
- la preuve de la décidabilité de la relation $\approx$ sur le sous-ensemble de termes du langage $\mathcal{L}^r_\beta$.

Étant donné la complexité du langage $\mathcal{L}$, nous n'avons pas réussi à définir une axiomatisation complète de ce langage. L'axiomatisation $\mathbf{A}$ permet néanmoins d'établir des propriétés importantes des divers opérateurs de composition et permet de décider l'équivalence $\approx$ sur un sous-langage particulier $\mathcal{L}^r_\beta \subset \mathcal{L}$.

Comme travail futur nous proposons d'élaborer davantage l'axiomatisation de l'algèbre ACTC pour pouvoir combiner des techniques de vérification de type *model-checking* avec des méthodes de vérification de type équationnelle.

Pour valider l'applicabilité et l'utilité de l'algèbre ACTC il serait intéressant de développer un outil de traduction automatique d'un HAAD vers le terme du langage $\mathcal{L}$ correspondant et de vérification automatique de diverses propriétés de la spécification, telles que la réactivité ou la cohérence. Un tel outil combinerait la méthode de spécification intuitive et claire des HAAD avec des techniques de vérification formelle

puissantes, propres aux spécifications algébriques. En expérimentant sur des exemples concrets d'interfaces matérielles, on pourrait alors identifier d'éventuelles limitations sémantiques du langage $\mathcal{L}$. Une extension sémantique intéressante du langage de spécification $\mathcal{L}$ pourrait être, par exemple, de définir des contraintes temporelles hiérarchiques, puisque dans $\mathcal{L}$ les contraintes temporelles sont définies à l'intérieur des feuilles seulement.

Finalement, on peut envisager la combinaison des méthodes de vérification algébriques avec des techniques de simulation d'interfaces matérielles: le modèle pourrait être une spécification algébrique, qui peut formellement décrire le comportement du système à simuler dans toute situation possible, par la sémantique opérationnelle du langage algébrique de spécification.

Chapitre 12

Annexe

Nous présentons dans cette annexe les preuves des propositions, lemmes, et théorèmes qui ont été énoncés sans être prouvés dans les chapitres précédents.

12.1 Preuves du chapitre 6

Dans cette section nous prouvons le lemme 6.1:

Soit $T \in \mathcal{LA}$ *un terme quelconque qui effectue une transition quelconque* $T \stackrel{a(v)}{\rightarrow} T'$, *avec* $a \in A$ *et* $v \geq V_0$. *Alors*

1. *si* $a = \delta$ *alors* $v = \alpha(T)$ *et* $T' = \downarrow$;
2. *si* $T \in \mathcal{LA}_\beta$ *est un terme feuille alors* $\mathrm{Act}(T') = \mathrm{Act}(T) \setminus \{a\}$;
3. *si* $T \in \mathcal{LA}_\mathcal{H}$ *est un terme hiérarchique alors* $\mathrm{Act}(T') \subseteq \mathrm{Act}(T)$.

Preuve: La preuve de la première propriété est triviale: il suffit de remarquer que chaque règle de SOp qui décrit un blocage a la propriété énoncée.

Prouvons les propriétés 2 et 3. La preuve est faite par induction structurelle. La base de l'induction consiste à prouver que la proposition est vérifiée par les actions temporisées $T = a(t_a) \in \mathcal{LA}$. La seule transition possible de T est $T \stackrel{a(v)}{\rightarrow} \uparrow$. Or, par

définition $\mathrm{Act}(T) = \{a\}$ et $\mathrm{Act}(\uparrow) = \emptyset$. Donc, $\mathrm{Act}(T) = \{a\} \cup \mathrm{Act}(\uparrow)$, ce qu'il fallait prouver.

Prouvons maintenant le pas d'induction: nous montrerons que pour tout terme $f[T_1, \ldots, T_n] \in \mathcal{LA}$ avec $n \geq 1$, si les sous-termes $T_1, \ldots, T_n$ ont la propriété énoncée dans la proposition, alors T aussi satisfait aux conditions de la proposition.

Soit $T = a(t_a).T_1 \in \mathcal{LA}$ un terme préfixé quelconque. Par définition

$$\mathrm{Act}(T) \stackrel{\mathrm{def}}{=} \{a\} \cup \mathrm{Act}(T_1).$$

Seule la règle $\mathbf{P}_2$ s'applique à cette situation. Selon cette règle T peut effectuer seulement la transition $T \stackrel{a(v)}{\rightarrow} \trianglelefteq_v(T_1)$, pour tout $v \in D$. Or, par définition $\mathrm{Act}(T) = \mathrm{Act}(T_1) \cup \{a\}$ et $\mathrm{Act}(\trianglelefteq_v(T_1)) = \mathrm{Act}(T_1)$, ce qu'il fallait prouver.

Les preuves pour les deux opérateurs de décalage sont analogues. Supposons alors que

$$T = \triangleleft_v(T_1).$$

Alors T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$ seulement s'il existe une transition $T_1 \stackrel{a(v)}{\rightarrow} T'$. On applique alors l'hypothèse d'induction et la définition de A:

$$\mathrm{Act}(T) = \mathrm{Act}(T_1) = \{a\} \cup \mathrm{Act}(T').$$

Considérons maintenant le cas où $T = \|(T_1, \ldots, T_n)$. T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$ si et seulement si un de ses sous-termes T_i effectue une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et les autres sous-termes T_j peuvent attendre jusqu'au temps v. Supposons, sans perte de généralité, que $i = 1$. Alors, il y a deux cas à considérer

1. $T_1' \neq \surd$, et $T' = \|(T_1', \trianglelefteq_v(T_2) \ldots, \trianglelefteq_v(T_n))$;

2. $T_1' = \uparrow$ et $T' = \|(\trianglelefteq_{T_2}, \ldots, \trianglelefteq_v(T_n))$.

Dans le premier cas on déduit les égalités suivantes:

$$\mathrm{Act}(T) \quad \stackrel{\mathrm{def}}{=} \quad \mathrm{Act}(T_1) \bigcup_{j=2}^{n} \mathrm{Act}(T_j)$$

$$\begin{aligned} &\overset{\text{hyp. d'ind.}}{=} && \{a\} \cup \mathrm{Act}(T_1') \bigcup_{j=2}^{n} \mathrm{Act}(T_j) \\ &\overset{\text{def}}{=} && \{a\} \cup \mathrm{Act}(T'), \end{aligned}$$

ce qu'il fallait prouver.

Dans le cas (2) il suffit de remarquer que $\mathrm{Act}(T_1) = \{a\}$, puisque, selon l'hypothèse d'induction, $\mathrm{Act}(T_1) = \{a\} \cup \mathrm{Act}(\uparrow)$ et $\mathrm{Act}(\uparrow) = \emptyset$. Alors

$$\mathrm{Act}(T) = \{a\} \bigcup_{j=2}^{n} \mathrm{Act}(T_j) = \{a\} \cup \mathrm{Act}(T').$$

Supposons maintenant que $T = \theta : S$. Alors pour toute transition $T \overset{a(v)}{\rightarrow} T'$ avec $a \neq \delta$ il existe une transition $S \overset{a(v)}{\rightarrow} S'$. Il y a deux possibilités: soit (1) $S' \neq \uparrow$ et $T' = \theta_{v/t_a} : S'$, ou bien (2) $S' = T' = \uparrow$. Dans les deux cas $\mathrm{Act}(T) = \mathrm{Act}(S)$ et $\mathrm{Act}(T') = \mathrm{Act}(S')$ (par définition). Alors, la proposition suit directement de l'hypothèse d'induction.

Les preuves pour les opérateurs réactifs Max et Min sont analogues. Supposons alors que $T = \mathrm{Max}(o, H, [m, M], S)$. Si T effectue une transition $T \overset{a(v)}{\rightarrow} T'$ avec $a \neq \delta$ il existe une transition $S \overset{a(v)}{\rightarrow} S'$. Dans chacune de ces transitions (qui correspondent aux règles $\mathbf{Max}_{1,2,3}$ de sémantique opérationnelle de Max) on remarque que $\mathrm{Act}(T) = \mathrm{Act}(S)$ et $\mathrm{Act}(T') = \mathrm{Act}(S')$. La proposition suit alors directement de l'hypothèse d'induction.

Considérons maintenant les termes hiérarchiques. Soit $T = +(T_1, \ldots, T_n) \in \mathcal{LA}_{\mathcal{H}}$ un terme hiérarchique quelconque. Alors pour toute transition $T \overset{a(v)}{\rightarrow} T'$ il doit exister une transition $T_i \overset{a(v)}{\rightarrow} T_i'$ et $T' = T_i'$. Donc, par hypothèse d'induction

$$\mathrm{Act}(T_i') \subseteq \mathrm{Act}(T_i).$$

Or, par définition

$$\mathrm{Act}(T) = \bigcup_{i=1}^{n} \mathrm{Act}(T_i).$$

Donc, $\mathrm{Act}(T') \subseteq \mathrm{Act}(T)$, ce qu'il fallait prouver.

Considérons le cas où $T = \text{Loop}(S) \in \mathcal{L}_{\mathcal{H}}$. Alors pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$ et une des cinq situations suivantes sont possibles:

1. $T' = \uparrow$,
2. $T' = \downarrow$,
3. $T' = \lhd_v(\text{Loop}(S))$,
4. $T' = \text{Seq}(S', \text{Loop}(S))$,
5. $T' = S'$.

Le premier cas n'est possible que si S effectue une transition $S \stackrel{a(v)}{\rightarrow} \uparrow$ et que la transition de T suit la règle $\mathbf{B}_4$. Or, selon l'hypothèse d'induction, $\{a\} \subseteq \text{Act}(S)$. Or, par définition, $\text{Act}(T) \stackrel{\text{def}}{=} \text{Act}(S)$. Donc, la proposition est vérifiée.

Dans le cas 2 on remarque que $T \stackrel{\delta(v)}{\rightarrow} \downarrow$ si et seulement si $S \stackrel{\delta(v)}{\rightarrow} \downarrow$. La propriété énoncée suit alors directement de l'hypothèse d'induction.

Si $T' = \lhd_v(\text{Loop}(S))$, alors $\text{Act}(T') = \text{Act}(\lhd_v(\text{Loop}(S)) = \text{Act}(T)$, et la propriété énoncée est vérifée trivivalement.

Dans le quatrième cas on applique l'hypothèse d'induction et on déduit que

$$\text{Act}(T') = \text{Act}(S') \ \cup \ \text{Act}(\text{Loop}(S)) = \text{Act}(S) = \text{Act}(T),$$

ce qu'il fallait montrer. Finalement, si $T' = S'$ la proposition est triviale.

Supposons maintenant que

$$T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m)$$

est un terme quelconque de $\mathcal{L}$ qui effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$. Par définition

$$\text{Act}([\mathcal{C}, \mathcal{E}](T_1, \ldots, T_m)) = \mathcal{C} \ \cup \ \bigcup_{i=1}^{m} \text{Act}(T_i) \setminus \bigcup_{c \in \mathcal{C}} \epsilon(c).$$

Si $T' \in \{ \uparrow, \downarrow\}$ la proposition est triviale. Sinon, alors

$$T' = [\mathcal{C}', \epsilon']_{\substack{i \in I(a) \\ j \in J(a)}} (T'_i, \trianglelefteq_v(T_j)),$$

où

$$\begin{aligned} I(a) &= \{i \mid \mathrm{Act}(T_i) \cap \epsilon(a) \neq \emptyset\} \\ J(a) &= \{1, \ldots m\} \setminus I(a) \\ a_i &= \epsilon(a, T_i), i \in I(a). \end{aligned}$$

et $\forall c \in \mathcal{C}$

$$\epsilon'(c) = \epsilon(c) \cap (\bigcup_{i \in I(a)} \mathrm{Act}(T_i') \bigcup_{j \in J(a)} \mathrm{Act}(T_j))$$

et

$$\mathcal{C}' = \{c \in \mathcal{C} \mid \epsilon'(c) \neq \emptyset\}.$$

Selon l'hypothèse d'induction on a que

$$\mathrm{Act}(T_i)' \subseteq \mathrm{Act}(T_i), \forall i \in I(a).$$

De plus, par définition

$$\mathrm{Act}(\trianglelefteq_v(T_j)) = \mathrm{Act}(\trianglelefteq_v(T_j)), \forall j \in J(a)$$

et $\mathcal{C}' \subseteq \mathcal{C}$. Donc, il suffit maintenant de remarquer que

$$\bigcup_{i \in I(a)} \mathrm{Act}(T_i') \bigcup_{j \in J(a)} \mathrm{Act}(\trianglelefteq_v(T_j)) \setminus \bigcup_{c \in C'} \epsilon'(c) = \tag{12.1}$$

$$\bigcup_{i \in I(a)} \mathrm{Act}(T_i') \bigcup_{j \in J(a)} \mathrm{Act}(T_j) \setminus \bigcup_{c \in C} \epsilon'(c) \subseteq \tag{12.2}$$

$$\bigcup_{i \in I(a)} \mathrm{Act}(T_i) \bigcup_{j \in J(a)} \mathrm{Act}(T_j) \setminus \bigcup_{c \in C} \epsilon'(c). \tag{12.3}$$

Donc, $\mathrm{Act}(T') \subseteq \mathrm{Act}(T)$, ce qu'il fallait prouver.

Supposons maintenant que

$$T = \oplus(T_1, \ldots, T_n) \in \mathcal{LA}$$

et considérons une transition quelconque $T \stackrel{a(v)}{\rightarrow} T'$. Alors une des situations suivantes peut se produire:

- $T' \in \{\uparrow, \downarrow\}$ et $\mathrm{Act}(T') = \emptyset \subseteq \mathrm{Act}(T)$;
- $T' = T'_i$ si T_i est le seul terme qui peut exécuter l'action $a(v)$ et $T_i \stackrel{a(v)}{\rightarrow} T'_i$. On applique alors l'hypothèse d'induction et on déduit que

$$\mathrm{Act}(T') = \mathrm{Act}(T'_i) \subseteq \mathrm{Act}(T_i) \subset \mathrm{Act}(T);$$

- $T' = \oplus_{i \in I}(T'_i)$, où $I \subseteq \{1, \ldots, n\}$ est l'ensemble d'indices des sous-termes qui peuvent effectuer une transition $T_i \stackrel{a(v)}{\rightarrow} T'_i$. Alors selon l'hypothèse d'induction $\mathrm{Act}(T'_i) \subseteq \mathrm{Act}(T_i)$ et

$$\mathrm{Act}(T') = \bigcup_{i \in I} \mathrm{Act}(T'_i) \subseteq \mathrm{Act}(T).$$

Donc, dans toutes ces situations la proposition est vérifiée.

Finalement, considérons le cas où

$$T = \mathrm{Ex}(B, C, E) \in \mathcal{LA}$$

est un terme de $\mathcal{LA}$ qui effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$. Les situations suivantes sont alors possibles:

- $T' \in \{\uparrow, \downarrow\}$, où la proposition est trivialement satisfaite;
- $T' = \mathrm{Ex}(B', C', E)$, $B \stackrel{a(v)}{\rightarrow} B'$, $C \stackrel{a(v)}{\rightarrow} C'$ et $C', B' \notin \{\uparrow, \downarrow\}$. Il suit alors directement de l'hypothèse d'induction que $\mathrm{Act}(T') \subseteq \mathrm{Act}(T)$;
- $T' = \trianglelefteq_v(E)$; ce cas est trivial, puisque

$$\mathrm{Act}(T') = \mathrm{Act}(E) \subset \mathrm{Act}(T).$$

Donc, on peut conclure que le lemme est vrai. ∎

12.2 Preuves du chapitre 7

Dans cette section nous prouvons les proposition 7.9 et 7.11.

12.2.1 Preuve de la proposition 7.9

L'énoncé de la proposition est comme suit:

Pour tous $T, T' \in \mathcal{L}$, $a \in A \setminus \{\delta\}$, *et* $v \geq V_0$,

$$T \stackrel{a(v)}{\rightarrow} T' \Rightarrow \mathrm{First}(a(v), T).$$

Preuve:

La proposition sera prouvée par induction structurelle. Nous remarquons d'abord que la proposition est vraie pour des termes préfixés. Ensuite, pour chaque opérateur de composition de Op, nous montrons que si la proposition est vérifiée par les sous-termes, alors elle est aussi vérifiée par le terme composé.

La base de l'induction est le cas où $T = a(t_a) \in \mathcal{L}$ est une action temporisée quelconque. Alors, le prédicat $\mathrm{First}(a(v), T)$ est trivialement vrai pour tout $v \in D$ et selon les règles de SOp, T peut effectuer la transition $T \stackrel{a(v)}{\rightarrow} \uparrow$ également pour tout $v \in D$. Donc la proposition est satisfaite.

Supposons alors que

$$T = a(t_a).S.$$

Le terme T peut exécuter l'action préfixée a à tout instant temporel $v \geq V_0$. Par définition,

$$\mathrm{First}(a(v), T) \stackrel{\mathrm{def.}}{\Leftrightarrow} \Gamma^{\geq v}(T) \cap \mathrm{Sol}(t_a = v) \neq \emptyset.$$

Selon la syntaxe de $\mathcal{L}$ $S \in \mathcal{L}$ doit être un *port*, c'est à dire il existe des actions $a_1, \ldots, a_n \in A$, avec $n \geq 1$, telles que $a_i \neq a_j$, pour tous $i, j : 1 \leq i, j \leq n$ et telles que $a \neq a_i$ pour tout $i : 1 \leq i \leq n$ et $S = a_1(t_{a_1}) \ldots a_n(t_{a_n})$. Donc, selon la définition de Γ,

$$\Gamma(T) = t_a \leq t_{a_1} \bigwedge_{i=1}^{n-1} t_{a_i} \leq t_{a_{i+1}}.$$

On remarque alors qu'il y a une infinité d'affectations des variables $t_a, t_{a_1}, \ldots, t_{a_n}$ qui satisfont cette formule, donc le prédicat First est vrai.

Considérons maintenant le cas où

$$T = \mathrm{ROp}(S) \in \mathcal{L}_\beta$$

est un terme quelconque de $\mathcal{L}$ avec où ROp est un opérateur réactif. Lorsque T exécute une action $a(v)$, avec $a \neq \delta$, son sous-terme S aussi doit exécuter la même action. Selon l'hypothèse d'induction alors la formule $\text{First}(a(v), S)$ est vrai. Or, $\Gamma(\text{ROp}(S)) = \Gamma(S)$ et $\text{Act}(\text{ROp}(S)) = \text{Act}(T)$. Donc,

$$\text{First}(a(v), \text{ROp}(S)) = \text{First}(a(v), S) = \text{Vrai}.$$

Soit maintenant $T = \theta : S \in \mathcal{L}$ un terme quelconque, où $\theta \in \Theta$ est une contrainte temporelle. Pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ avec $a \neq \delta$, deux conditions doivent être satisfaites: le sous-terme S doit exécuter $a(v)$ et le prédicat $\text{First}(a(v), \theta : T)$ doit être vrai. La proposition est, donc, satisfaite trivialement.

Considérons les termes

$$T = \|(T_1, \ldots, T_n) \in \mathcal{L}.$$

Toute transition $T \stackrel{a(v)}{\rightarrow} T'$ a lieu si et seulement si un des sous-termes T_i effectue une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et les autres sous-termes peuvent attendre jusqu'au temps v. Supposons, sans perte de généralité, que $i = 1$. Alors, selon l'hypothèse d'induction, le prédicat $\text{First}(a(v), T_1)$ doit être vrai. En plus, puisque $\text{Wait}(v, T_j)$ est vrai, pour tout $j \geq 2$, il existe au moins une transition

$$T_j \stackrel{b_j(v_j)}{\rightarrow} T_j'$$

telle que $v' \geq v$ pour tout $j \geq 2$. En appliquant encore une fois l'hypothèse d'induction on déduit que $\text{First}(b_j(v_j), T_j)$ est vrai aussi. Selon les règles de composition de termes, les ensembles d'actions des sous-termes T_i sont disjoints. Donc, puisque $a \in \text{exec}(T_1)$ on déduit que $a \notin \text{Act}(T_j)$, pour tout $j \geq 2$. En tenant compte de ceci et du fait que $v' \geq v$ on déduit que

$$\text{First}(b_j(v_j), T_j) \Rightarrow \text{First}(a(v), T_j), \tag{12.4}$$

pour tout $j \geq 2$. De plus, puisque

$$\begin{aligned} \Gamma(\|(T_1, \ldots, T_n)) &= \bigcap_{i=1}^{n} \Gamma(T_i) \text{ et} \\ \text{Act}(\|(T_1, \ldots, T_n)) &= \bigcup_{i=1}^{n} \text{Act}(T_i), \end{aligned}$$

on a que

$$\begin{aligned}\text{First}(a(v), \|(T_1, \ldots, T_n)) &\Leftrightarrow (\bigcap_{i=1}^{n} \Gamma(T_i))^{\geq v} \cap \text{Sol}(t_a = v) \neq \emptyset \\ &\Leftrightarrow \bigcap_{i=1}^{n} \Gamma^{\geq v}(T_i) \cap \text{Sol}(t_a = v) \neq \emptyset \\ &\Leftrightarrow \bigwedge_{i=1}^{n} \text{First}(a(v), T_i).\end{aligned}$$

Or le prédicat $\text{First}(a(v), T_1)$ est vrai et puisque $First(b_j(v_j), T_j)$ est vrai aussi et selon l'équation 12.4, on déduit que $\text{First}(a(v), \|(T_1, \ldots, T_n)$ est vrai.

Il reste à considérer les cas où le terme T est composé à l'aide des opérateurs de décalage. Nous allons prouver la proposition pour l'opérateur $\lhd$ seulement; la preuve pour $\trianglelefteq$ est analogue. Soient

$$T = \lhd_{v'}(S)$$

et $S \in \mathcal{L}_\beta$. À toute transition $T \xrightarrow{a(v)} T'$, où $a \neq \delta$ et $v > v'$, correspond une transition $S \xrightarrow{a(v)} S'$. Selon l'hypothèse d'induction alors, le prédicat $\text{First}(a(v), S)$ est vrai. Remarquons que

$$\Gamma(T) = \Gamma^{>v'}(S) \tag{12.5}$$

Donc,

$$\begin{aligned}\text{First}(a(v), T) &\overset{\text{def.}}{\Leftrightarrow} \Gamma^{\geq v}(T) \cap \text{Sol}(t_a = v) \neq \emptyset \\ &\overset{(1)}{\Leftrightarrow} (\Gamma^{>v'}(S))^{>v} \cap \text{Sol}(t_a = v) \neq \emptyset.\end{aligned}$$

Mais, puisque $v > v'$

$$(\Gamma^{>v'}(S))^{>v} = \Gamma^{>v}(S).$$

Donc,

$$\text{First}(a(v), T) \Leftrightarrow \text{First}(a(v), S)$$

et le prédicat est vrai $\text{First}(a(v), T)$ est vrai, ce qu'il fallait prouver.

Considérons maintenant les termes hiérarchiques. Soit

$$T = \sum_{j=1}^{n} T_j,$$

où $T_i \in \mathcal{L}$ et supposons que $a(v) \in \text{exec}(T)$. Alors au moins un des sous-termes T_i doit être tel que $a(v) \in \text{exec}(T_i)$. Selon l'hypothèse d'induction on a que

$$\text{First}(a(v), T_i) \Leftrightarrow \Gamma^{\geq v}(T_i) \cap \text{Sol}(t_a = v) \neq \emptyset = \text{Vrai}.$$

Or, par définition,

$$\Gamma^{\geq v}(T) = \bigcup_{j=1}^{n} \Gamma^{\geq v}(T_j).$$

Donc,

$$\text{First}(a(v), T_i) \Rightarrow \text{First}(a(v), T).$$

Selon l'hypothèse d'induction, le prédicat $\text{First}(a(v), T_i)$ est vrai. Donc, on peut conclure que

$$\text{First}(a(v), T) = \text{Vrai},$$

ce qu'il fallait montrer.

Considérons maintenant le cas où

$$T = [\mathcal{C}, \epsilon](T_1, \ldots, T_n)$$

et $a(v) \in \text{exec}(T)$ avec $a \neq \delta$. Par définition on a que

$$\text{Act}([\mathcal{C}, \mathcal{E}](T_1, \ldots, T_n)) = \mathcal{C} \cup \bigcup_{i=1}^{m} \text{Act}(T_i) \setminus \bigcup_{c \in \mathcal{C}}^{m} \epsilon(c)$$

et

$$\Gamma([\mathcal{C}, \mathcal{E}](T_1, \ldots, T_m)) = (\bigcap_{i=1}^{m} \Gamma(T_i))_{t_c/t_a}, \forall c \in \mathcal{C}, \forall a \in \epsilon(c).$$

Notons par $I(a) \subseteq \{1, \ldots, m\}$ le sous-ensemble d'indexes des sous-termes qui communiquent via l'action $a \in \text{Act}(T)$:

$$I(a) = \{i \in \{1, \ldots, m\} \mid \text{Act}(T_i \cap \epsilon(a) \neq \emptyset\}$$

Rappelons que si $a \in \mathcal{C}$, alors l'ensemble $\epsilon(a)$ contient toutes les projections $a_i = \epsilon(a, T_i)$ de a sur les sous-termes T_i qui communiquent via a

$$\epsilon(a) \stackrel{\text{def}}{=} \{\epsilon(a, T_i) \mid i \in I(a)\}$$

et si $a \notin \mathcal{C}$, alors

$$\epsilon(a) \stackrel{\text{def}}{=} \{a\}.$$

Supposons d'abord que $T \stackrel{a(v)}{\rightarrow} T'$ et $a \in \text{Act}(T) \setminus \mathcal{C}$. Alors il existe un (unique) sous-terme T_i tel que $T_i \stackrel{a(v)}{\rightarrow} T_i'$. Selon l'hypothèse d'induction, le prédicat $\text{First}(a(v), T_i)$ est vrai. De plus, il faut que le prédicat $\text{Wait}(v, T_j)$ soit vrai pour tous les autres sous-termes T_j, avec $j \in \{1, \ldots, m\} \setminus \{i\}$. Par définition, alors

$$(\forall j \in \{1, \ldots, m\} \setminus \{i\})(\exists v^j \geq v_i, a_j, T_j')(T_j \stackrel{a_j(v_j)}{\rightarrow} T_j').$$

Or ceci implique, par hypothèse d'induction, que pour tout $j \neq i$ il existe au moins un vecteur

$$\vec{v_j} \in \Gamma(T_j),$$

tel que toutes les valeurs temporelles de $\vec{v_j}$ sont supérieures ou égales à v^j et $V^j \geq v$. Si on tient compte aussi du fait que

$$\text{Act}(T_j) \cap \text{Act}(T_k) = \emptyset,$$

pour tous $j \neq k$, alors on peut déduire que le prédicat $\text{First}(a(v), T)$ doit être vrai.

Supposons maintenant que $a \in \mathcal{C}$. Le raisonnement est similaire à celui présenté plus haut pour $a \notin \mathcal{C}$. Il suffit de remarquer en plus que lorsque T exécute $a(v)$, tous les sous-termes T_i qui communiquent via l'action a exécutent leur action interne respective $a_i = \epsilon(a, T_i)$. Selon l'hypothèse d'induction on déduit que

$$\text{First}(a_i(v), T_i)$$

pour tout i tel que T_i est un sous-terme qui communique via a.

Considérons maintenant le cas où

$$T = \text{Seq}(T_1, T_2) \in \mathcal{L}.$$

La proposition suit alors directement du fait que $\text{exec}(T) = \text{exec}(T_1)$ et que $\Gamma(T) \stackrel{\text{def}}{=} \Gamma(T_1)$.

Si

$$T = \oplus(T_1, \ldots, T_m),$$

alors

$$\text{exec}(T) \subseteq \bigcup_{i=1}^{m} \text{exec}(T_i)$$

et

$$\Gamma^{\geq v}(\oplus(T_1, \ldots, T_m)) = \bigcup_{i=1}^{m} \Gamma^{\geq i}(T_i).$$

Il suffit alors d'appliquer l'hypothèse d'induction pour déduire que l'égalité voulue.

Le cas où $T = \text{Loop}(T_1)$ est trivial puisque $\text{exec}(T) = \text{exec}(T_1)$ et $\Gamma(T) = \Gamma(T_1)$.

■

12.2.2 Preuve de la proposition 7.11

L'enoncé de la proposition est le suivant:

Soit $T \in \mathcal{L} \setminus \{\uparrow, \downarrow\}$ *un terme quelconque. Alors* $\text{exec}(T) \neq \emptyset$.

Preuve:

La preuve sera faite par induction structurelle. Le pas d'induction consiste à prouver que si tous les sous-termes $S_i \in \mathcal{L}$, avec $1 \leq i \leq n$ et $n \geq 1$, d'un terme quelconque $T = f[S_1, \ldots, S_n] \in \mathcal{L}$ peuvent exécuter au moins une action, alors ceci est vrai pour T aussi.

La base de l'induction sont des termes $a(t_a)$, avec $a \in \mathbf{A}_\beta$. Un terme $a(t_a)$ peut exécuter l'action a à tout instant $v \geq V_0$, donc la condition est satisfaite trivialement.

À titre d'exemple nous présentons ici le cas où $T = \|(T_1, \ldots, T_n)$, avec $T_i \in \mathcal{L}_\beta$, $i = 1, \ldots, n$. Les autres cas sont soit analogues ou bien triviales.

Selon l'hypothèse d'induction, pour chaque sous-terme T_i on a que

$$\text{exec}(T_i) \neq \emptyset.$$

Notons par $a_0(v_0)$ l'action temporisée telle que

$$(\exists i_0 \in \{1, \ldots, n\})(a_0(v_0) \in \text{exec}(T_{i_0})),$$

et telle que

$$(\forall i \in \{1, \ldots, n\})(\forall a(v) \in \text{exec}(T_i))(v \geq v_0).$$

Autrement dit, $a_0(v_0)$ est la première action temporisée parmi les actions des sous-termes $T_1, \ldots, T_n$ qui peut être exécutée. Donc, il existe une transition

$$T_{i_0} \overset{a_{i_0}(v_{i_0})}{\rightarrow} T'_{i_0}$$

et pour tout $i \neq i_0$ il existe une transition

$$T_i \overset{a_i(v_i)}{\rightarrow} T'_i$$

avec $v_i \geq v$. Or ceci implique que pour tout $i \neq i_0$ le prédicat $\text{Wait}(v_{i_0}, T_i)$ est vrai.

Il y a deux cas à considérer: soit (1) $a_0 \neq \delta$ out bien (2) $a_0 = \delta$.

Dans le premier cas, où $a \neq \delta$, il suit directement des règles $\|_1$–$\|_3$ que T peut exécuter $a_j(v_j)$.

Si $a_j = \delta$, il suffit de remarquer qu'aucun des sous-termes T_i avec $i \neq i_0$ ne peut exécuter aucune action avant le temps v_0. Donc, $\text{exec}^{<v_0}(T_i) = \emptyset$, pour tout $i \neq i_0$. Selon la règle $\|_4$ alors, T exécute l'action de blocage $\delta(v_{i_0})$. ∎

12.3 Preuves du chapitre 8

12.3.1 Preuve de la proposition 8.6

L'enoncé de la proposition est le suivant:

Soient $T, T_i, i = 1, 2, 3$ *des termes quelconques de* $\mathcal{L}^+_\beta$. *Alors* $\forall v \geq V_0$, $\forall a \in A$, *et* $\forall v' \geq V_0$,

1. $\text{Wait}(v, a(t_a).T) \Leftrightarrow v \geq V_0$,

2. (a) $v' \geq v \Rightarrow \text{Wait}(v, \lhd_{v'}(T))$,

   (b) $v' < v \Rightarrow (\text{Wait}(v, \lhd_{v'}(T)) \Leftrightarrow \text{Wait}(v, T))$,

3. (a) $v' \geq v \Rightarrow \text{Wait}(v, \trianglelefteq_{v'}(T))$,

   (b) $v' < v \Rightarrow (\text{Wait}(v, \trianglelefteq_{v'}(T)) \Leftrightarrow \text{Wait}(v, T))$,

4. $\text{Wait}(v, \theta : T) \Rightarrow \text{Wait}(v, T)$,

5. $\text{Wait}(\text{Max}(o, H, [m, M], T)) \Rightarrow \text{Wait}(v, T)$,

6. $\text{Wait}(\text{Min}(o, H, [m, M], T)) \Rightarrow \text{Wait}(v, T)$,

7. $\text{Wait}(v, \|(T_1, \ldots, T_m)) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v, T_1)$,

8. $\text{Wait}(v, +(T_1, \ldots, T_m)) \Leftrightarrow \bigvee_{i=1}^{m} \text{Wait}(v, T_i)$,

9. $\text{Wait}(v, \text{Seq}(T_1, T_2)) \Leftrightarrow \text{Wait}(v, T_1)$,

10. $\text{Wait}(v, [\mathcal{C}, \epsilon](T_1, \ldots, T_m)) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v, T_i)$,

11. $\text{Wait}(v, \oplus(T_1, \ldots, T_m)) \Rightarrow \bigvee_{i=1}^{m} \text{Wait}(v, T_i)$,

12. $\text{Wait}(v, \text{Ex}(T_n, T_c, T_e)) \Leftrightarrow \text{Wait}(v, T_n) \wedge \text{Wait}(v, T_c)$,

13. $\text{Wait}(v, \text{Loop}(T)) \Leftrightarrow \text{Wait}(v, T)$.

La première égalité est triviale. Les égalités 2. et 3. sont analogues. L'égalité 2. a été prouvée dans le chapitre 8. Nous prouvons ici les points 4 – 13.

12.3.1.1 Preuve des implications 4, 5, et 6

Les preuves des trois implications sont analogues: il suffit de remarquer que, exceptant les règles $\mathbf{supp}_4$, $\mathbf{Max}_4$, et $\mathbf{Min}_4$, respectivement, pour toute transition du terme composé il y a une transition du sous-terme T. Dans le cas des exception mentionnées ci-haut, les contraintes temporelles sont contradictoires et le terme composé bloque à son temps de démarrage α. Or, le temps de démarrage du sous-terme T est aussi égale à α (proposition 7.2). T peut, donc, trivialement attendre jusqu'au temps α. ∎

12.3.1.2 Preuve de l'équivalence 7

Prouvons l'implication droite d'abord. Si le terme $T = \|(T_1, \ldots, T_m)$ peut attendre jusqu'au temps v alors il existe une transition $T \stackrel{a(v')}{\rightarrow} T'$ telle que $v' \geq v$. Selon la sémantique opérationnelle de $\|$, ceci est possible seulement si un des sous-termes T_i peut exécuter $a(v')$ et tous les autres sous-termes T_j, avec $j \neq i$, peuvent attendre jusqu'au temps v'. Selon la définition de Wait, on a que $\text{Wait}(v', T_i)$ est vrai. Or, $v' \geq v$, donc $\text{Wait}(v, T_i)$ et $\text{Wait}(v, T_j)$ sont vrai aussi, ce qu'il fallait prouver.

Prouvons l'implication gauche maintenant. Supposons que $\text{Wait}(v, T_i)$ est vrai pour tous les sous-termes T_i. Alors, chaque sous-terme peut effectuer une transition

$$T_i \stackrel{a_i(v_i)}{\rightarrow} T_i',$$

avec $v_i \geq v$, pour tout i. Soit k l'indice du sous-terme T_i qui effectue cette transition en premier. Formellement,

$$k = \text{argmin}_{i=1}^{m} v_i.$$

Alors, T_k peut exécuter l'action $a_k(v_k)$ et pour tout i le prédicat $\text{Wait}(v_k, T_i)$ est vrai, puisque $v_k \leq v_i$. Selon les règles la sémantique opérationnelle de l'opérateur $\|$, le terme T peut alors exécuter l'action $a_k(v_k)$. Ceci implique que le prédicat $\text{Wait}(v_k, T)$ est vrai. Mais $v_k \geq v$, donc, le prédicat $\text{Wait}(v, T)$ est vrai aussi, ce qu'il fallait prouver. ∎

12.3.1.3 Preuve de l'équivalence 8

La preuve est triviale: il suffit de remarquer que $+(T_1, \ldots, T_m)$ exécute une action quelconque $a(v)$ si et seulement si au moins un des sous-termes T_i exécute $a(v)$. ∎

12.3.1.4 Preuve de l'équivalence 9

Il est trivial de montrer que:

$$\text{Wait}(v, \text{Seq}(T_1, T_2)) \Leftrightarrow \text{Wait}(v, T_1) :$$

il suffit de remarquer que $\text{exec}(\text{Seq}(T_1, T_2) = \text{exec}(T_1)$. ∎

12.3.1.5 Preuve de l'équivalence 10

Notons par T le terme composé $[\mathcal{C}, \epsilon](T_1, \ldots, T_m)$. L'implication à droite est évidente et suit directement des règles de sémantique opérationnelle: si T exécute une action quelconque $a(v)$, alors tous les sous-termes T_i soit exécutent une action $b_i(v)$ ou bien sont tels que le prédicat $\text{Wait}(v, T_i)$ est vrai.

Prouvons l'implication à gauche:

$$(\forall i \in \{1, \ldots, m\})\text{Wait}(v, T_i) \Rightarrow \text{Wait}(v, T).$$

Supposons que le prédicat $\text{Wait}(v, T_i)$ est vrai pour tous les sous-termes T_i et un instant temporel quelconque v. Alors il existe des transitions $T_i \overset{a_i(v_i)}{\rightarrow} T_i'$, avec $a_i \in A, T_i' \in \mathcal{L}$, et $v_i \geq v$. Soit k l'indice de la première de ces transitions. Formellement,

$$k = \text{argmin}_{i=1}^{m}(v_i).$$

Alors, $T_k \overset{a_k(v_k)}{\rightarrow} T_k'$ et, puisque $v_k \leq v_i$, le prédicat $\text{Wait}(v_k, T_i)$ est vrai pour tout $i \neq k$. Nous devons considérer les trois cas :(1) a_k est une action interne de T, (2) a_k est une action non-interne $a_k \in \text{Act}(T)$, et (3) $a_k = \delta$.

Supposons d'abord que a_k est une action interne de T, c'est à dire qu'il existe une action de communication $c \in \mathcal{C}$ telle que $a_k \in \epsilon(c)$. Si la communication des sous-termes peut s'établir, T va exécuter l'action de communication $c(v_k)$ qui correspond à $a_k(v_k)$. Sinon T exécute $\delta(v_k)$. Dans les deux situations, le prédicat $\text{Wait}(v_k, T)$ est vrai. Comme $v_k \geq v$ ceci implique que $\text{Wait}(v, T)$ est vrai aussi, ce qu'il fallait prouver.

Dans le deuxième cas T exécute l'action $a_k(v_k)$ et le prédicat $\text{Wait}(v, T)$ est, donc, validé.

Dans le cas (3) T exécute $\delta(v_k)$, donc le prédicat $\text{Wait}(v, T)$ est vrai. ∎

12.3.1.6 Preuve de l'implication 11

L'implication est immédiate: il suffit de remarquer que $T = \oplus(T_1, \ldots, T_m)$ effectue une transition quelconque à un temps v seulement si au moins un de ses sous-termes

le fait aussi. ∎

12.3.1.7 Preuve de l'équivalence 12

Le prédicat $\text{Wait}_v(\text{Ex}(T_n, T_c, T_e))$ est vrai si et seulement s'il existe une transition $\text{Ex}(T_n, T_c, T_e) \stackrel{a(v')}{\rightarrow} T$, où $a \in A$ (donc possiblement l'action de blocage) et $v' \geq v$. On distingue alors deux cas possibles: soit (1) T_n et T_c peuvent tous les deux exécuter $a(v)$ (règle $\mathbf{Ex}_1$), ou bien (2) un des sous-termes T_n, T_c peut exécuter $a(v)$ et l'autre sous-terme peut attendre jusqu'au temps v. Dans les deux cas, les prédicats $\text{Wait}_v(T_n)$ et $\text{Wait}_v(T_c)$ sont vrais, ce qui conclut cette preuve. ∎

12.3.1.8 Preuve de l'équivalence 13

La preuve est immédiate: on remarque que, pour toute action $a \in A$ et tout $v \geq V_0$, $\text{Loop}(T)$ exécute $a(v)$ si et seulement si T exécute $a(v)$. ∎

12.3.2 Preuve de la proposition 8.7

Rappelons l'énoncé de la proposition:

Soit $T \in \mathcal{L}_\beta$ un terme feuille quelconque. Si $\delta(v) \in \text{exec}(T)$, alors $\neg\text{Wait}(v', T)$, pour tout $v' > v$. En plus, pour toute action $a(v) \in \text{exec}(T)$, telle que $a \neq \delta$, toutes les traces $\mu = a(v)... \in \mathbb{T}(T)$, doivent finir dans un blocage, autrement dit $\mu = a(v)...\delta(v)$.

Preuve:

La preuve sera faite par induction structurelle. La base de l'induction consiste à prouver que la proposition est vraie dans des situations de *blocage non-hérité* de SOp. Rappelons qu'une transition quelconque

$$T = f[T_1, \ldots, T_n] \stackrel{a(v)}{\rightarrow} T'$$

d'un terme quelconque $T \in \mathcal{LA}$ est *héritée*, si et seulement s'il existe un sous-terme T_i de T et une transition

$$T_i \stackrel{a(v)}{\rightarrow} T_i'.$$

Dans le cas des blocages hérités nous appliquons le pas d'induction: nous prouvons que si les sous-termes de T ont la propriété énoncée dans la proposition, alors T aussi a cette propriété.

Prouvons d'abord la base de l'induction: les règles $\trianglelefteq_2$, $\triangleleft_2$, $\mathbf{supp}_4$, $\mathbf{Max}_4$, et $\mathbf{Min}_4$ définissent toutes les situations de blocage non-hérité du sous-langage $\mathcal{L}_\beta$.

Soit $T = \trianglelefteq_v(T_0)$ et supposons que c'est la règle de sémantique opérationnelle $\trianglelefteq_2$ qui s'applique. Alors, par définition, T ne peut exécuter aucune action au temps v ou plus tard et exécute $\delta(v)$. L'énoncé est, donc, satisfait trivialement. Le raisonnement pour la règle $\triangleleft_2$ est analogue.

Supposons que $T = \theta : T_0$ et que T bloque selon la règle $\mathbf{supp}_4$. On remarque qu'aucune autre règle de transition ne peut s'appliquer à T et, donc,

$$\mathrm{exec}(T) = \{\delta(\alpha(T))\}$$

et la proposition est vérifiée.

Lorsqu'on applique la règle $\mathbf{Max}_4$, respectivement et $\mathbf{Min}_4$, à un terme

$$T = \mathrm{ROp}(\{a\}, o, [m, M], T),$$

on a que $\mathrm{exec}(T) = \{a\}$ et par conséquent $\mathrm{exec}(T) = \{\delta(\alpha(T))$ et la proposition et vérifiée.

Prouvons le pas d'induction. Il ne reste qu'a considérer les blocages hérités, notamment ceux définis par les règle $\|_4$ et $\mathbf{supp}_3$ du tableau 6.3, ainsi que par les règle $\mathbf{Max}_3$ et $\mathbf{Min}_3$ du tableau 6.4.

Si (règle SOp $\|_4$) $T = \|(T_1, \ldots, T_m)$ et un sous-terme T_i effectue une transition $T_i \stackrel{\delta(v)}{\rightarrow} \downarrow$ lorsque pour tous les autres sous-termes T_j le prédicat $\mathrm{Wait}(v, T_j)$ est vrai, alors le terme T exécute $\delta(v)$ et transite vers $\downarrow$. Selon l'hypothèse d'induction,

$$\neg\mathrm{Wait}(v', T_i), \forall v' > v.$$

Or, selon le théorème 8.6,

$$\mathrm{Wait}(v', T) \Leftrightarrow \bigwedge_{j=1}^{m} \mathrm{Wait}(v', T_j).$$

On déduit que $\neg \text{Wait}(v', T), \forall v' > v$. Il reste à prouver que pour toute action $a(v) \in \text{exec}(T)$ avec $a \neq \delta$ toutes les traces $\mu = a(v) \ldots \in \mathbb{T}(T)$ finissent éventuellement par l'action $\delta(v)$. On remarque que les hypothèses de la règle $\|_4$ peuvent être satisfaites simultanément avec les hypothèses des règle $\|_1$, $\|_2$, et $\|_3$, notamment lorsqu'il existe un sous-terme T_j qui peut effectuer une transition $T_j \stackrel{a(v)}{\rightarrow} T'_j$, avec $a \neq \delta$. Supposons qu'on peut appliquer la loi $\|_1$, les autres situations sont analogues.

Donc, on a que $T \stackrel{a(v)}{\rightarrow} T'$ et

$$T' = \|(\trianglelefteq_v(T_1, \ldots, \trianglelefteq_v(T_i), \ldots, T'_j, \ldots, \trianglelefteq_v(T_m)).$$

On remarque que $\delta(v) \in \text{exec}(\trianglelefteq_v(T_i)$ et que les prédicats $\text{Wait}(v, \trianglelefteq_v(T_k)$ $\text{Wait}_v(v, T'_j)$ sont vrais. Donc, $\delta(v) \in \text{exec}(T')$. On peut réitérer ce raisonnement tant qu'il existe des actions $a'(v) \in \text{exec}(T')$, respectivement dans un des successeurs de T', noté T'', telles que $a' \neq \delta$. Donc, à partir de T il existe seulement des traces:

$$T \stackrel{a(v)}{\rightarrow} T' \stackrel{a'(v)}{\rightarrow} \ldots \stackrel{a''(v)}{\rightarrow} T''.$$

Plusieures situations différentes sont possibles:

- ou bien t $T'' = \trianglelefteq_v(\ldots(\trianglelefteq_v(T'_i)\ldots))$,
- ou encore $T'' = \|(T''_i, \ldots, \trianglelefteq_v(\ldots(\trianglelefteq_v(T'_i)\ldots)), \ldots, T''_n)$, et le prédicat $\text{Wait}(v, T''_k)$ est vrai pour $k \neq i$ et est tel que

$$\nexists a(v) \in \text{exec}(T''), a \neq \delta.$$

Or, $\delta(v) \in \text{exec}(\trianglelefteq_v(\ldots(\trianglelefteq_v(T'_i)\ldots)))$. Donc, le terme T'' doit exécuter $\delta(v)$ et transiter vers $\downarrow$.

Dans le cas où $T = \theta : T_1$ et l'on applique la règle $\mathbf{supp}_3$, il suffit de remarquer que $\mathbb{T}(T) \subseteq \mathbb{T}(T_1)$ et on peut appliquer directement l'hypothèse d'induction.

La même remarque est suffisante pour prouver les cas $T = \text{ROp}(T_1)$, où ROp= Max ou Min et l'on applique soit la règle $\mathbf{Max}_3$ ou la règle $\mathbf{Min}_3$. ∎

12.3.3 Preuve de la proposition 8.8

L'énoncé de la proposition est le suivant:

Soit $T \in \mathcal{L}_\beta$ un terme feuille quelconque. Alors, pour tout $v \geq V_0$, le prédicat $\text{Wait}(v, T)$ *est vrai si et seulement si toutes les actions de* $\text{exec}(T)$ *peuvent encore être exécutées au temps v ou plus tard. Formellement,*

$$\text{Wait}(v, T) \Leftrightarrow \text{Wait}_\beta(v, T).$$

Preuve: La preuve est faite par induction structurelle. La base, qui consiste des termes préfixés, et le pas d'induction pour les termes composés en parallèle ont été prouvés dans la section 8.3.1. Ici nous prouvons le pas d'induction pour les opérateurs temporels.

Soit $T = \theta : S$, où $\theta \in \Theta$ est une contrainte descriptive et $S \in \mathcal{L}_\beta$ est un sous-terme qui satisfait l'énoncé du théorème. Soit $v \geq V_0$ tel que le prédicat $\text{Wait}(v, T)$ est vrai. Par définition, il existe alors une transition $T \stackrel{a(v_a)}{\rightarrow} T'$, avec $a \in A$ et $v_a \geq v$. Si toutes les actions de $\text{exec}(T)$ on lieu au temps v ou plus tard, il n'y a rien à prouver. Dans le cas contraire, soit $b(v_b) \in \text{exec}(T)$, $b \neq a$, $v_b < v$ une autre action quelconque de $\text{exec}(T)$. Nous voulons prouver qu'il existe un instant temporel $v'_b \geq v$ tel que $b(v'_b) \in \text{exec}(T)$.

Considérons d'abord le cas où $a \neq \delta$ et $b \neq \delta$. Alors, selon les règles $\mathbf{supp}_1$ et $\mathbf{supp}_2$, il faut que les prédicats $\text{First}(a(v_a), \theta : T)$ et $\text{First}(b(v_b), \theta : T)$ soient vrais et que $a(v_a), b(v_b) \in \text{exec}(S)$. Le prédicat $\text{Wait}(v_a, S)$ est, donc, vrai aussi. Selon l'hypothèse d'induction appliquée au sous-terme S, il existe alors un temps $v_b \geq v_a \geq v$ tel que $b(v'_b) \in \text{exec}(S)$.

Il reste à prouver que le prédicat $\text{First}(b(v'_b, \theta : T)$ est vrai pour conclure que $b(v''_b) \in \text{exec}(T)$. Cette preuve est assez lourde dans le cas général, où T a un nombre $m \geq 2$ arbitraire d'actions. Nous montrons ici que la propriété est vrai si T ne contient que les deux actions a et b.

T est une terme feuille, donc la fonction $\Gamma(T)$ décrit un système d'inéquations

linéaires de la forme:

$$m_{a,b} \leq t_a - t_b \leq M_{a,b} \wedge \tag{12.6}$$

$$m_a \leq t_a \leq M_a \wedge \tag{12.7}$$

$$m_b \leq t_b \leq M_b \wedge \tag{12.8}$$

Les solutions d'un tel système d'inéquations décrivent un espace convexe. Puisque $\text{First}(a(v_a), \theta : T)$ est vrai, il existe un point $A = (v_a, y_b) \in D^*$ tel que

$$v \leq v_a \leq y_b.$$

De manière similaire, parce que $\text{First}(a(v_b), \theta : T)$ est vrai, il existe un point $B = (x_a, v_b) \in D^*$ tel que

$$v_b \leq v \wedge v_b \leq x_a.$$

Alors, puisque Γ et convexe, $\Gamma(T)$ doit nécessairement contenir aussi au moins un point $C = (v'_a, v'_b)$ tel que

$$v \leq v'_b \leq v_a.$$

Puisque $a, b \in \text{exec}(T)$, alors il ne peut y avoir aucun opérateur réactif contraignant les temps d'occurrence des actions a et b. Selon le théorème 10.5, dont la preuve ne dépend pas du résultat présent, T peut exécuter la trace $\mu = b(v'_b).a(v'_a)$, ce qu'il fallait prouver.

Le raisonnement pour les deux opérateurs réactifs Max et Min est analogue. Nous faisons ici la preuve correspondant à un terme composé avec l'opérateur Max seulement. Soit $T = \text{Max}(o, H, [m, M], S) \in \mathcal{L}_\beta$ tel que le sous-terme $S \in \mathcal{L}_\beta$ satisfait la propriété énoncée dans ce théorème et tel que le prédicat $\text{Wait}(v, T)$ est vrai pour un instant temporel $v \in D$ quelconque. Supposons que $\text{exec}(S) \neq \{o\}$. Dans ce cas on a que

$$\text{exec}(T) = \text{exec}(S) \setminus \{o\}.$$

Donc, si l'énoncé est vrai pour S, il l'est trivialement aussi pour T. Si $\text{exec}(S) = \{o\}$ (règle $\mathbf{Max}_4$), alors

$$\text{exec}(T) = \{\delta(\alpha(T))\}$$

et les prédicats $\text{Wait}(v, T)$ et $\text{Wait}_\beta(v, T)$ sont équivalents. ∎

12.3.4 Preuve de la propriété 8.3

Nour voulons prouver que pour tous $T_1, \ldots, T_m \in \mathcal{L}$ et tout $v \geq 0$

$$\text{Wait}(v, \oplus(T_1, \ldots, T_m)) \Leftarrow \bigvee_{i=1}^{m} \text{Wait}_\beta(v, T_i). \tag{12.9}$$

Preuve: Supposons que pour au moins un des sous-termes T_i le prédicat $\text{Wait}_\beta(v, T_i)$ est vrai. Il existe, donc, une transition $T_i \stackrel{a(v')}{\rightarrow} T_i'$ avec $v' \geq v$ et $a \neq \delta$.

Supposons d'abord que $a = ic$. Dans ce cas T va soit exécuter $ic(v)$ ou bien $\delta(v)$, tout dépendant si (i) tous les sous-termes, (ii) certains sous-termes , ou (iii) aucun sous-terme ne transitent vers $\uparrow$ en exécutant l'action $ic(v)$. Dans les cas (i) et (iii) T exécute $ic(v)$ et dans le cas (ii) T exécute $\delta(v)$. Donc, le prédicat $\text{Wait}(v, T)$ est vrai.

Sinon, alors $a = oc$ et il suit directement de la règle $\oplus_4$ que T exécute $\delta(v)$. ∎

12.3.5 Preuve du théorème 8.10

Prouvons ici le théorème de congruence:

*La relation de bisimulation–*ACTC $\approx$ *est une congruence pour tous les opérateurs de* Op *sur le langage* $\mathcal{L}$. *Plus précisément:*

1. *pour tout* $v \geq V_0$ *et tous* $T, S \in \mathcal{L}$

$$T \approx S \Rightarrow \trianglelefteq_v(T) \approx \trianglelefteq_v(S)$$
$$T \approx S \Rightarrow \triangleleft_v(T) \approx \triangleleft_v(S)$$

2. *pour tout* $a \in A$ *et* $T, S \in \mathcal{L}_\beta$,

$$T \approx S \Rightarrow a(t_a).T \approx a(t_a).S$$

3. *pour tous* $T_1, \ldots, T_m, S_1, \ldots, S_m \in \mathcal{L}_\beta$,

$$(\forall i = 1, \ldots, m)(T_i \approx S_i) \Rightarrow \|(T_1, \ldots, T_m) \approx \|(S_1, \ldots, S_m);$$

4. *pour tous* $T_1, S_1 \in \mathcal{L}_\beta$, *et tout* $\theta \in \Theta$ *tel que* $\mathrm{Act}(\theta) \subseteq \mathrm{Act}(T_1)$ *et* $\mathrm{Act}(\theta) \subseteq \mathrm{Act}(S_1)$,

$$T_1 \approx S_1 \Rightarrow \theta : T_1 \approx \theta : S_1;$$

5. *pour tout opérateur réactif* $\mathrm{ROp} = (o, H, [m, M], \cdot)$ *et tous* $T, S \in \mathcal{L}_\beta$, *tels que* $\{o\} \cup H \subseteq \mathrm{Act}(T)$ *et* $\{o\} \cup H \subseteq \mathrm{Act}(S)$,

$$T \approx S \Rightarrow \mathrm{ROp}(T) \approx \mathrm{ROp}(S)$$

6. *pour tous* $n \geq 2$ *et* $T_1, \ldots T_n, S_1, \ldots, S_n \in \mathcal{L}$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow \sum_{i=1}^{n} T_i \approx \sum_{i=1}^{n} (S_i)$$

7. *pour tous* $T_1, T_2, S_1, S_2 \in \mathcal{L}$

$$(\forall i = 1, 2)(T_i \approx S_i) \Rightarrow \mathrm{Seq}(T_1, T_2) \approx \mathrm{Seq}(S_1, S_2)$$

8. *pour tous* $T, S \in \mathcal{L}$,

$$T \approx S \Rightarrow \mathrm{Loop}(T) \approx \mathrm{Loop}(S).$$

9. *pour tous* $n \geq 2$ *et* $T_1, \ldots, T_n, S_1, \ldots, S_n \in \mathcal{L}$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow \oplus(T_1, \ldots, T_n) \approx \oplus(S_1, \ldots, S_n)$$

10. *pour tous* $n \geq 2$, *tous* $T_1, \ldots, T_n, S_1, \ldots, S_n \in \mathcal{L}$, *et tout opérateur de composition parallèle avec communication* $[\mathcal{C}, \epsilon]$,

$$(\forall i = 1, \ldots, n)(T_i \approx S_i) \Rightarrow [\mathcal{C}, \epsilon](T_1, \ldots, T_n) \approx [\mathcal{C}, \epsilon](S_1, \ldots, S_n)$$

11. *pour tous* $T_{n1}, T_{c1}, T_{e1}, T_{n2}, T_{c2}, T_{e2} \in \mathcal{L}$,

$$(T_{n1} \approx T_{n2}) \wedge (T_{c1} \approx T_{c2}) \wedge (T_{e1} \approx T_{e2}) \Rightarrow \mathrm{Ex}(T_{n1}, T_{c1}, T_{e1}) \approx \mathrm{Ex}(T_{n2}, T_{c2}, T_{e2})$$

12.3.5.1 Congruence de $\approx$ par rapport aux opérateurs $\trianglelefteq$ et $\lhd$

Nous ne considérons que l'opérateur de décalage stricte $\lhd$, la preuve pour l'opérateur $\trianglelefteq$ est analogue.

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\lhd_v(S), \lhd_v(T)) | (S,T) \in \approx, v \in D\} \cup \approx .$$

Nous prouverons que $\mathcal{R}$ est une bisimulation-ACTC.

Selon le raisonnement présenté dans la section 8.2, il suffit de considérer les paires $(T_1, T_2) \in \mathcal{R}$ telles que $T_1 = \lhd_v(S)$ et $T_2 = \lhd_v(T)$, pour un certain temps v et telles que $S \approx T$.

La première propriété de $\mathcal{R}$ que nous devons vérifier est que $\alpha(T_1) = \alpha(T_2)$. Par définition, l'opérateur de décalage met le temps de démarrage $\alpha(\lhd_v(\cdot))$ à v. Donc, cette condition est satisfaite trivialement.

Deuxièmement, il faut que $\Gamma(\lhd_v(S)) = \Gamma(\lhd_v(T))$. Or, par définition,

$$\Gamma(\lhd_v(S)) = \Gamma^{\geq v}(S) \text{ et } \Gamma(\lhd_v(T)) = \Gamma^{\geq v}(T).$$

Puisque $S \approx T$ par définition on a que $\Gamma(S) = \Gamma(T)$. Donc,

$$\Gamma(\lhd_v(S)) = \Gamma(\lhd_v(T)).$$

Il reste à vérifier que $\mathcal{R}$ est une bisimulation forte, c'est à dire que $\lhd_v(T)$ et $\lhd_v(S)$ peuvent exécuter les mêmes actions aux mêmes instants temporels, et que leurs successeurs respectifs sont dans $\mathcal{R}$. Il y a deux transition à analyser, qui correspondent aux règles de sémantique opérationnelle $\lhd_1$ et $\lhd_2$.

La première transition $\lhd_v(S) \xrightarrow{a(v')} S'$ a lieu lorsque $S \xrightarrow{a(v')} S'$ et $v' > v$. Puisque $S \approx T$, le terme $\lhd_v(T)$ aussi peut exécuter $a(v')$ et transite alors vers T', où

$$(S', T') \in \approx \subseteq \mathcal{R}.$$

Selon la deuxième règle, le terme $\lhd_v(S)$ exécute $\delta(v)$ parce que S ne peut exécuter aucune action après le temps v. Par conséquent, T non-plus ne peut exécuter aucune

action après le temps v et $\lhd_v(S)$ signale un blocage au temps v. Or,

$$(S', T') = (\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}.$$

Nous pouvons conclure que $\mathcal{R}$ est une bisimulation–ACTC et, donc, que $\approx$ est une congruence par rapport à $\lhd$. ∎

12.3.5.2 Congruence de $\approx$ par rapport à l'opérateur de préfixage

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation définie par

$$\mathcal{R} = \{(a(t_a).S, a(t_a).T) \mid (S, T) \in \approx, a \in A\} \cup \approx .$$

Nous prouverons que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire nous devons vérifier que les termes $(a(t_a).S, a(t_a).T)$ de $\mathcal{R}$ satisfont les conditions 1–3 de la définition d'une relation de bisimulation–ACTC.

D'abord

$$\alpha(a(t_a).S) = \alpha(S) = \alpha(T) = \alpha(a(t_a).T) = V_0,$$

parce que les termes préfixés et leurs sous-termes sont des termes initiaux de $\mathcal{L}_0$ et ont, donc, un temps de démarrage égale à V_0.

Ensuite, selon la syntaxe de $\mathcal{L}$, T et S doivent être des ports, c'est à dire des séquences d'actions composées par préfixage. Donc, si $T \approx S$, forcément ils doivent avoir les mêmes actions. Donc,

$$\begin{aligned} \Gamma(a(t_a).S) &= \mathrm{Sol}(\bigwedge_{b \in \mathrm{Act}(S)} t_b > t_a)) \cap \Gamma(S) \\ &= \mathrm{Sol}(\bigwedge_{b \in \mathrm{Act}(T)} t_b > t_a)) \cap \Gamma(T) \\ &= \Gamma(a(t_a).T). \end{aligned}$$

Il reste à prouver que $\mathcal{R}$ est une bisimulation forte. Il suffit de considérer les paires de termes $(a(t_a).S, a(t_a).T)$, avec $S \approx T$. Le terme $a(t_a).S$ peut exécuter l'action a à tout instant $v \geq \alpha(S)$ et transite vers $\trianglelefteq_v(S)$. Puisque $S \approx T$, par définition,

$\alpha(S) = \alpha(T)$ et, étant donné que nous avons déjà prouvé la congruence de $\approx$ par rapport à l'opérateur $\trianglelefteq$ (section 12.3.5.1), on sait que $\trianglelefteq_v(S) \approx \trianglelefteq_v(T)$. Donc, $\mathcal{R}$ est une bisimulation et nous pouvons conclure que $\approx$ est une congruence par rapport au préfixage d'actions. ∎

12.3.5.3 Congruence de $\approx$ par rapport à la composition parallèle $\|$

Soient $m \geq 2$ un entier quelconque et $T_i, \ldots, T_m, S_1, \ldots, S_m \in \mathcal{L}_\beta$ des termes tels que

$$\mathrm{Act}(T_i) \cap \mathrm{Act}(T_j) = \mathrm{Act}(S_i) \cap \mathrm{Act}(S_j) = \emptyset, \forall i \neq j.$$

Nous voulons prouver que

$$(\forall i = 1, \ldots, m)(T_i \approx S_i) \Rightarrow \|(T_1, \ldots, T_m) \approx \|(S_1, \ldots, S_m)$$

Pour ce faire nous définissons la relation $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ suivante:

$$\mathcal{R} = \{(\|(T_1, \ldots, T_m), \|(S_1, \ldots S_m)) \mid (T_i, S_i) \in \approx, 1 \leq i \leq m, m \geq 2\} \cup \approx,$$

et montrons que $\mathcal{R}$ est une bisimulation–ACTC.

Il suffit de considérer des termes $S = \|(S_1, \ldots, S_m)$, et $T = \|(T_1, \ldots, T_m))$ avec $S_i \approx T_i$ pour tout i et prouver que la paire (T, S) satisfait aux conditions énoncées dans la définition 8.1 (voir raisonnement de la section 8.2).

Selon le lemme 8.9 on déduit que $\alpha(T) = \alpha(S)$. Comme $S_i \approx T_i$, il suit que $\Gamma(S_i) = \Gamma(T_i)$, pour tout i. Donc,

$$\begin{aligned} \Gamma(T) &\stackrel{\text{def}}{=} \bigcap_{i=1}^{m} \Gamma(T_i) \\ &\stackrel{\text{def}\approx}{=} \bigcap_{i=1}^{m} \Gamma(S_i) \\ &\stackrel{\text{def}}{=} \Gamma(S). \end{aligned}$$

Prouvons maintenant que $\mathcal{R}$ est une bisimulation forte. On va considérer d'abord les règles de sémantique opérationnelle $\mathbf{Par}_{1,2,3}$. On suppose, qu'un des sous-termes

de S exécute une transition $S_i \stackrel{a(v)}{\rightarrow} S_i'$, avec $a \neq \delta$, et que tous les autres sous-termes S_j peuvent attendre jusqu'au moment v. Puisque $T_i \approx S_i$, il existe une transition correspondante $T_i \stackrel{a(v)}{\rightarrow} T_i'$, telle que $S_i' \approx T_i'$. De plus, par le Corollaire 8.3, on a que $\text{Wait}(v, S_j) \Leftrightarrow \text{Wait}(v, T_j)$, pour tout j.

Donc, pour toute transition de $S \stackrel{a(v)}{\rightarrow} S'$ avec $a \neq \delta$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$. On doit montrer maintenant, que lors de chacune de ces transitions $(S', T') \in \mathcal{R}$.

Si $S_i' \neq \uparrow$, et par conséquent $T_i' \neq \uparrow$, alors

$$\begin{aligned} S' &= \|(\trianglelefteq_v S_1, \ldots, S_i', \ldots, \trianglelefteq_v(S_m)), \\ T' &= \|(\trianglelefteq_v T_1, \ldots, T_i', \ldots, \trianglelefteq_v(T_m)). \end{aligned}$$

Puisque nous avons déjà prouvé que $\approx$ est une congruence par rapport à l'opérateur $\trianglelefteq$ on déduit que $\trianglelefteq_v(T_j) \approx \trianglelefteq_v(S_j)$ et que, par conséquent, $(T', S') \in \mathcal{R}$.

Sinon, c'est à dire si $S_i' = \uparrow$, alors il y a deux cas à considérer. Premièrement, si $m = 2$ on applique la règle $\|_3$ et déduit que $S' = \trianglelefteq_v(S_j)$ et $T' = \trianglelefteq_v(T_j)$. Or, puisque $S_j \approx T_j$, ceci implique que $(T', S') \in \approx \subseteq \mathcal{R}$. Deuxièmement, si $m \geq 3$ on déduit selon la règle $\|_2$ que

$$\begin{aligned} S' &= \|(\trianglelefteq_v S_1, \ldots, \trianglelefteq_v(S_{i-1}), \trianglelefteq_v(S_{i+1}), \ldots, \trianglelefteq_v(S_m)), \\ T' &= \|(\trianglelefteq_v T_1, \ldots, \trianglelefteq_v(T_{i-1}), \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_m)). \end{aligned}$$

Donc, puisque $\approx$ est une congruence par rapport à l'opérateur $\trianglelefteq$, $(T', S') \in \mathcal{R}$.

Il reste à considérer les situations correspondantes à la règle $\|_4$. Si $T_i \stackrel{\delta(v)}{\rightarrow} \downarrow$ alors il existe la transition $S_i \stackrel{\delta(v)}{\rightarrow} \downarrow$. De plus, on remarque que $\text{exec}^{\leq v}(T_j) = \text{exec}^{\leq v}(S_j)$, pour tout j. Donc, le terme S va exécuter $\delta(v)$ et transiter vers $\downarrow$ et $(T', S') = (\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$. ∎

12.3.5.4 Congruence de $\approx$ par rapport aux opérateurs réactifs

Les preuves correspondantes aux deux opérateurs Min et Max sont analogues. Nous n'allons montrer la congruence de $\approx$ que par rapport à Max.

Nous voulons prouver que $\text{Max}(o, H, [m, M], S) \approx \text{Max}(o, H, [m, M], T)$ pour toute paire de termes $(S, T) \in \approx$, toute action de sortie $o \in O$, tout ensemble d'actions source $H \subseteq A$, et tout intervalle réactif $[m, M]$.

Nous allons prouver que la relation $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ est une bisimulation–ACTC, où

$$\begin{aligned}\mathcal{R} &= \{(\text{Max}(o, H, [m, M], S), \text{Max}(o, H, [m, M], T)) \mid (S, T) \in \approx, o \in O, H \subseteq A, \\ &\quad [m, M] \text{ intervalle réactif de} D\} \cup \approx .\end{aligned}$$

Il suffit de considérer une paire $(S, T) \in \mathcal{R}$ telle que

$$\begin{aligned}S &= \text{Max}(o, H, [m, M], S_1) \in \mathcal{L} \text{ et} \\ T &= \text{Max}(o, H, [m, M], T_1) \in \mathcal{L}\end{aligned}$$

et telle que $S_1 \approx T_1$. Les deux premières conditions, $\alpha(S) = \alpha(T)$ et $\Gamma(T) = \Gamma(S)$ sont trivialement satisfaites selon le lemme 8.9 et par la définition de la fonction Γ.

Il reste à prouver que $\mathcal{R}$ est une bisimulation forte. Il suffit de montrer que le T peut exécuter les mêmes actions aux mêmes instants temporels que S et que leurs successeurs respectifs, T' et S', sont tels que $(S', T') \in \mathcal{R}$.

On remarque que les hypothèses des quatre différentes transitions de S, correspondantes aux règle $\mathbf{Max}_{1-4}$, sont satisfaites pour S si et seulement si elles sont satisfaites pour T. Donc les termes S et T peuvent exécuter les mêmes actions en même temps. Dans le cas des transitions qui correspondent à la règle $\mathbf{Max}_4$ ceci est assuré par le fait que $\alpha(S) = \alpha(T)$.

Notons par S_1' et T_1' les successeurs des sous-termes S_1 et T_1 lors des ces transitions. Puisque $S_1 \approx T_1$ il il existe des successeurs T_1' et S_1' tels que $S_1' \approx T_1'$. Les sucesseurs S' et T' de S et T sont alors

$$\begin{aligned}S' &= \text{Max}(o, H \setminus \{a\}, [m, M], S_1') \\ T' &= \text{Max}(o, H \setminus \{a\}, [m, M], T_1'),\end{aligned}$$

pour la règle $\mathbf{Max}_1$,

$$S' = m \leq t_o - v \leq M : S_1'$$

$$T' \;=\; m \leq t_o - v \leq M : T',$$

pour la $\mathbf{Max}_2$,

$$S' = T' = \uparrow ,$$

pour la règle $\mathbf{Max}_3$, et

$$S' = T' = \downarrow ,$$

pour la règle $\mathbf{Max}_4$. Or, dans tous ces cas, $(S', T') \in \mathcal{R}$: dans le premier cas selon la définition de $\mathcal{R}$, dans le deuxième cas selon la congruence de l'opérateurs θ, déjà prouvée dans la section 8.3.2), et dans les autres cas parce que $\approx$ est réflexive (théorème 8.1).

On conclut que $\approx$ est un congruence par rapport à l'opérateur réactif Max. ∎

12.3.5.5 Congruence de $\approx$ par rapport à l'opérateur +

Nous voulons montrer que

$$(\forall i = 1, \ldots, n) T_i \;\approx\; S_i \Rightarrow \sum_{i=1}^{n} T_i \;\approx\; \sum_{i=1}^{n} (S_i).$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation de termes suivante:

$$\mathcal{R} = \{(+(S_1, \ldots, S_m), +(T_1, \ldots, T_m)) \mid (S_i, T_i) \in \approx , 1 \leq i \leq m, m \geq 2\}$$

et vérifions que $\mathcal{R}$ est une bisimulation–ACTC.

Il suffit de considérer une paire quelconque $(S, T) \in \mathcal{R}$ telle que $S = +(S_1, \ldots, S_m)$, et $T = +(T_1, \ldots, T_m)$ avec $(S_i, T_i) \in \approx$ pour tout i, et de prouver que (S, T) satisfait aux conditions enoncées dans la définition 8.1.

Premièrement, les termes S et T ont les mêmes temps de démarrage, puisque leurs sous-termes ont les mêmes temps de démarrage (lemme 8.9).

Ensuite, on a que

$$\begin{aligned} \Gamma(S) &= \bigcup_{i=1}^{m} \Gamma(S_i) \\ &= \bigcup_{i=1}^{m} \Gamma(T_i) \\ &= \mathrm{Sol}(\Gamma(T)). \end{aligned}$$

Prouvons maintenant que $\mathcal{R}$ est une bisimulation forte, c'est à dire que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$, telle que $(S', T') \in \mathcal{R}$.

La preuve de cette propriété est triviale: il suffit de remarquer que toute transition $S \stackrel{a(v)}{\rightarrow} S'$ correspond à une transition $S_i \stackrel{a(v)}{\rightarrow} S'$, pour un i quelconque, $1 \leq i \leq m$. Donc, puisque $S_i \approx T_i$, pour tout i, $1 \leq i \leq m$, il existe une transition $T_i \stackrel{a(v)}{\rightarrow} T'$ telle que $(S', T') \in \approx \subseteq \mathcal{R}$. Pour la règle de sémantique opérationnelle $\mathbf{C}_4$, on remarque en plus que $\text{Wait}(v, T_i) \Leftrightarrow \text{Wait}(v, S_i)$ (Corollaire 8.3). ∎

12.3.5.6 Congruence de $\approx$ par rapport à la composition séquentielle

Montrons que pour tous $T_1, T_2, S_1, S_2 \in \mathcal{L}$

$$(\forall i = 1, 2)(T_i \approx S_i) \Rightarrow \text{Seq}(T_1, T_2) \approx \text{Seq}(S_1, S_2)$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\text{Seq}(S_1, S_2), \text{Seq}(T_1, T_2)) \mid S_i \approx T_i, i = 1, 2\} \cup \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Puisque $\mathcal{R}$ est par définition symétrique, il suffit de considérer des paires $(S, T) \in \mathcal{R}$ telles que

$$S = \text{Seq}(S_1, S_2) \text{ et } T = \text{Seq}(T_1, T_2),$$

et telles que $S_i \approx T_i$ avec $i = 1, 2$ (voir le raisonnement de la section 8.2).

D'abord, par le lemme 8.9, $\alpha(S) = \alpha(T)$. Ensuite, c'est trivial de vérifier que $\Gamma(T) = \Gamma(S)$, puisque $T_1 \approx S_1$ et que, par définition de la relation $\approx$, $\Gamma(T_1) = \Gamma(S_1)$. Or, par définition

$$\Gamma(\text{Seq}(T_1, T_2)) = \Gamma(T_1) \text{ et } \Gamma(\text{Seq}(S_1, S_2)) = \Gamma(S_1).$$

Montrons maintenant que $\mathcal{R}$ est une bisimulation forte. On remarque d'abord que, selon la sémantique opérationnelle de Seq, $\text{exec}(S) = \text{exec}(S_1)$ et $\text{exec}(T) = \text{exec}(T_1)$. Or, comme $S_1 \approx T_1$ on a que $\text{exec}(S_1) = \text{exec}(T_1)$. Donc, S et T peuvent exécuter

les mêmes actions temporisées. Il reste à montrer que leurs successeurs respectifs, S' et T', lors de l'exécution d'une même action temporisée $a(v)$ quelconque, forment une paire $(S', T') \in \mathcal{R}$. On a alors que $S_1 \stackrel{a(v)}{\rightarrow} S'_1$ et $T_1 \stackrel{a(v)}{\rightarrow} T'_1$.

Si $a = \delta$, alors $S' = T' = \downarrow$ et $(S', T') \in \approx \subseteq \mathcal{R}$ (réflexivité de la relation $\approx$ – théorème 8.1).

Sinon, puisque $S_1 \approx T_1$, et par conséquent $S'_1 \approx T'_1$, alors $S'_1 = \uparrow$ si et seulement si $T'_1 = \uparrow$. Donc, si $S'_1 = \uparrow$ alors $S' = \lhd_v(S_2)$ et $T' = \lhd_v(T_2)$. Or, $S_2 \approx T_2$ et $\approx$ est une congruence pour l'opérateur de décalage $\lhd$ (théorème 8.10, équation 1). Ceci implique que $(S', T') \in \approx \subseteq \mathcal{R}$.

Si $S'_1 \neq \uparrow$, alors $S' = \text{Seq}(S'_1, S_2)$ et $T' = \text{Seq}(T'_1, T_2)$. Par définition de $\mathcal{R}$ alors on obtient que $(S', T') \in \mathcal{R}$. On peut conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

12.3.5.7 Congruence de $\approx$ par rapport à l'opérateur Loop

On va prouver que pour tous $T, S \in \mathcal{L}$,

$$T \approx S \Rightarrow \text{Loop}(T) \approx \text{Loop}(S).$$

Soient $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_{\text{synt}}, \mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ les relations définies comme suit:

$$\begin{aligned}
\mathcal{R}_1 &= \{(\text{Loop}(S_k), \text{Loop}(T_k)) \mid (S_k, T_k) \in \approx\} \\
\mathcal{R}_2 &= \{(\text{Seq}(S'_k, \text{Loop}(S_k)), \text{Seq}(T'_k, \text{Loop}(T_k))) \mid (S_k, T_k) \in \approx, (S'_k, T'_k) \in \approx, v \geq V_0\} \\
\mathcal{R}_3 &= \{(\lhd_v(\text{Loop}(S_k)), \lhd_v(\text{Loop}(T_k)) \mid \\
&\quad S_k \approx T_k, v_0 \geq V_0\} \\
\mathcal{R}_{\text{synt}} &= \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 \\
\mathcal{R} &= \{(S, T) \mid (\exists (S_0, T_0) \in \mathcal{R}_{\text{synt}})(S \approx S_0 \wedge T \approx T_0)\} \cup \approx .
\end{aligned}$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Selon le raisonnement présenté dans la section 8.2, il suffit de considérer juste des paires quelconques $(S, T) \in \mathcal{R}_{\text{synt}}$ dans notre preuve. On va d'abord supposer que $(S, T) \in \mathcal{R}_1$, ensuite $(S, T) \in \mathcal{R}_2$, et finalement que $(S, T) \in \mathcal{R}_3$.

Soit (S, T) une paire quelconque de $\mathcal{R}_1$. Il existe, donc, des termes S_k et T_k tels que $S = \text{Loop}(S_k)$ et $T = \text{Loop}(T_k)$ et $(S_k, T_k) \in \approx$. D'abord $\alpha(S) = \alpha(S_k)$ et $\alpha(T) = \alpha(T_k)$ (selon la syntaxe de $\mathcal{L}$). Comme $S_k \approx T_k$, par définition $\alpha(S_k) = \alpha(T_k)$. Donc, $\alpha(T) = \alpha(S)$.

De manière similaire, on prouve que $\Gamma(T) = \Gamma(S)$.

Montrons maintenant que $\text{exec}(S) = \text{exec}(T)$ et que leurs successeurs respectifs, S' et T', lors de l'exécution d'une même action $a(v)$ sont tels que $(S', T') \in \mathcal{R}$. Selon la sémantique opérationnelle de l'opérateur Loop, définie par les règles $\mathbf{B}_{1-5}$, on déduit que $\text{exec}(S) = \text{exec}(S_k)$ et $\text{exec}(T) = \text{exec}(T_k)$. Puisque $S_k \approx T_k$, alors $\text{exec}(S_k) = \text{exec}(T_k)$. Donc, S peut exécuter une action quelconque $a(v)$ si et seulement si T peut l'exécuter. Notons par S', T', S'_k, et T'_k, respectivement, les successeur de S, T, S_k, et respectivement T_k lors de l'exécution d'une action $a(v)$, tels que $S'_k \approx T'_k$. Il faut prouver que $(S', T') \in \mathcal{R}$.

Selon la règle $\mathbf{B}_1$, on a que

$$\begin{aligned} T' &= \text{Seq}(T'_k, \text{Loop}(T_k)), \\ S' &= \text{Seq}(S'_k, \text{Loop}(S_k)). \end{aligned}$$

Donc, $(S', T') \in \mathcal{R}_2 \subseteq \mathcal{R}$.

Considérons la règle $\mathbf{B}_2$: on a que $T' = T'_k$ et $S' = S'_k$. Or, puisque $S_k \approx T_k$ ils ont des successeurs S'_k et T'_k tels que $S'_K \approx T'_k$. Donc, par transitivité, $(S', T') \in \approx \subseteq \mathcal{R}$.

Suivant la règle $\mathbf{B}_3$ on obtient que

$$\begin{aligned} T' &= \lhd_v(\text{Loop}(T_k)), \\ S' &= \lhd_v(\text{Loop}(S_k)). \end{aligned}$$

Alors $(S', T') \in \mathcal{R}_3 \subseteq \mathcal{R}$.

Finalement, selon les règles $\mathbf{B}_4$, respectivment $\mathbf{B}_5$, $S' = T' = \uparrow$, respectivement $\downarrow$. $(S', T') \in \approx \subseteq \mathcal{R}$.

Supposons maintenant que $(S, T) \in \mathcal{R}_2$. Donc, il existe des paires de termes telles

$(S_k, T_k), (S'_k, T'_k) \in \approx$, tels que

$$\begin{aligned} S &= \text{Seq}(S'_k, \text{Loop}(S_k)) \\ T &= \text{Seq}(T'_k, \text{Loop}(T_k)). \end{aligned}$$

Alors on remarque que, selon les règles $\mathbf{Seq}_{1--3}$, $\text{exec}(S) = \text{exec}(S'_k)$ et $\text{exec}(T) = \text{exec}(T'_k)$. Or, $\text{exec}(S'_k) = \text{exec}(T'_k)$, puisque $S'_k \approx T'_k$. Donc, S peut exécuter une action quelconque $a(v)$ si et seulement si T peut l'exécuter, et si et seulement si S'_k et T'_k peuvent l'exécuter. Notons leurs successeurs lors de cette exécution par S', T', S''_k, et T''_k, respectivement. Notons que $S''_k \approx T''_k$. Prouvons maintenant que $(S', T') \in \mathcal{R}$.

Selon la règle $\mathbf{Seq}_1$, les successeurs de S et T sont

$$S' = \text{Seq}(S''_k, \text{Loop}(S_k)) \tag{12.10}$$

$$T' = \text{Seq}(T''_k, \text{Loop}(T_k)). \tag{12.11}$$

Donc, $(S', T') \in \mathcal{R}_2 \subseteq \mathcal{R}$.

Si S et T suivent la règle $\mathbf{Seq}_2$, alors

$$S' = \lhd_v(\text{Loop}(S_k)) \tag{12.12}$$

$$T' = \lhd_v(\text{Loop}(T_k)). \tag{12.13}$$

Or, $(\lhd_v(\text{Loop}(S_k), \lhd_v(\text{Loop}(T_k)) \in \mathcal{R}_3$. Donc, $(S', T') \in \mathcal{R}$.

Enfin, selon la règle $\mathbf{Seq}_3$, on a que $S' = T' = \downarrow$. Donc, $(S', T') \in \approx \subseteq \mathcal{R}$.

Il reste à considérer une paire quelconque $(S, T) \in \mathcal{R}_3$. Soient

$$\begin{aligned} S &= \lhd_v(\text{Loop}(S_k)) \text{ et} \\ T &= \lhd_v(\text{Loop}(T_k)), \end{aligned}$$

avec $S_k \approx T_k$ et $v \geq 0$. D'abord $\alpha(S) = \alpha(T) = v$. Ensuite, par définition, nous avons que

$$\Gamma(\text{Loop}(T)) = \Gamma(T) \text{ et } \Gamma(\text{Loop}(S)) = \Gamma(S),$$

et, puisque $T_k \approx S_k$,

$$\Gamma(T_k) = \Gamma(S_k).$$

Donc,

$$\begin{aligned} \Gamma(S) &\stackrel{\text{def}}{=} \Gamma^{>v}(\text{Loop}(S_k) \\ &= \Gamma^{>v}(\text{Loop}(T_k) \\ &\stackrel{\text{def}}{=} \Gamma(T). \end{aligned}$$

Prouvons maintenant que $\text{exec}(S) = \text{exec}(T)$. On remarque d'abord que

$$\text{exec}(\text{Loop}(T)) = \text{exec}(T),$$

pour tout terme $T \in \mathcal{L}$. Rappelons que pour un terme quelconque nous notons $\text{exec}^{>v}(T)$ l'ensemble

$$\text{exec}^{>v}(T) = \{a(v') \in \text{exec}(T) \mid v' > v\}.$$

On remarque alors que si $\text{exec}_v(T) \neq \emptyset$, alors $\text{exec}(\lhd_v(T)) = \text{exec}^{>v}(T)$. Sinon, $\text{exec}(\lhd_v(T)) = \{\delta(v)\}$. En plus, étant donné que $S_k \approx T_k$, on a que $\text{exec}(S_k) = \text{exec}(T_k)$. On peut donc déduire que $\text{exec}(S) = \text{exec}(T)$.

Supposons que $\text{exec}^{>v}(S_k) \neq \emptyset$. Alors S exécute une action $a(v')$ avec $v' \geq v$ si et seulement si $\text{Loop}(S_k)$ l'exécute, et donc, si et seulement si $\text{Loop}(T_k)$ et T peuvent l'exécuter. Notons par S', S'', T', et T'' les successeurs de $S, \text{Loop}(S_k), T$, et $\text{Loop}(T_k)$, respectivement, lors d'une telle exécution. Alors, selon la règle $\lhd_{v1}$, $S' = S''$ et $T' = T''$. Or la paire $(\text{Loop}(S_k), \text{Loop}(T_k))$ est dans $\mathcal{R}_1$ et nous avons déjà montré que la paire de successeurs qui lui correspond (S'', T'') est dans $\mathcal{R}$. On déduit, donc, que $(S', T') \in \mathcal{R}$.

Si $\text{exec}^{>v}(S_k) = \text{exec}^{>v}(T_k) = \emptyset$, alors $\text{exec}(S) = \text{exec}(T) = \{\delta(v)\}$ et $\text{succ}(S) = \text{succ}(T) = \{\downarrow\}$. Or, $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$.

On peut finalement conclure que $\mathcal{R}$ est une bisimulation–ACTC et, donc, que $\approx$ est une congruence par rapport à l'opérateur Loop. ∎

12.3.5.8 Congruence de $\approx$ par rapport à l'opérateur $\oplus$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\oplus(T_1, \ldots, T_m), \oplus(S_1, \ldots, S_m)) \mid T_i, S_i \in \mathcal{L}, T_i \approx S_i, 1 \leq i \leq m\} \cup \approx .$$

Nous prouverons que $\mathcal{R}$ est une bisimulation–ACTC. Selon le raisonnement présenté dans la section 8.2 il suffit de prouver qu'une paire quelconque $(T, S) \in \mathcal{R}$, telle que

$$T = \oplus(T_1, \ldots, T_m) \text{ et}$$
$$S = \oplus(S_1, \ldots, S_m),$$

et telle que $T_i \approx S_i$ pour tout i, a les trois propriétés énoncés dans la définition 8.1.

Il suit directement du lemme 8.9 que

$$\alpha(T) = \alpha(S).$$

La preuve du fait que $\Gamma(T) = \Gamma(S)$ est identique à celle présentée dans la section 12.3.5.5 pour l'opérateur $+$.

Prouvons maintenant que (S, T) satisfait aux conditions données dans la définition de la bisimulation forte.

On va commencer par montrer que $\text{exec}(S) = \text{exec}(T)$. Dans le cas des transitions suivant les règles $\mathbf{CR}_1, \ldots, \mathbf{CR}_6$ il suffit de remarquer que, puisque $T_i \approx S_i$ pour tout i, $\text{exec}(T_i) = \text{exec}(S_i)$. Donc, pour tout j, tel que $1 \leq j \leq 5$ l'hypothèse $\mathbf{Hyp}_j(T)$ est satisfaite si et seulement si $\mathbf{Hyp}_j(S)$ et satisfaite. Donc, il est évident que $\text{exec}(T) = \text{exec}(S)$.

Considérons la règle $\mathbf{CR}_7$. Remarquons d'abord que

$$J = \{j \mid T_j \overset{\delta}{\not\rightarrow}\} = \{j \mid S_j \overset{\delta}{\not\rightarrow}\}.$$

De plus, $T_i \overset{\delta(v)}{\rightarrow}$ si et seulement si $S_i \overset{\delta(v)}{\rightarrow}$. Donc,

$$v = \max_{i \notin J}\{v_i \mid T_i \overset{\delta(v_i)}{\rightarrow} \downarrow\} = \max_{i \notin J}\{v_i \mid S_i \overset{\delta(v_i)}{\rightarrow} \downarrow\}.$$

Il reste à prouver que

$$\mathbf{Hyp}_k(\bigoplus_{j\in J} \trianglelefteq_v(T_j)) \Leftrightarrow \mathbf{Hyp}_k(\bigoplus_{j\in J} \trianglelefteq_v(S_j)) \tag{12.14}$$

pour tout k, $1 \leq k \leq 6$. Nous avons déjà prouvé que $\approx$ est une congruence par rapport à l'opérateur de décalage $\trianglelefteq_v$. Donc, $\trianglelefteq_v(T_j) \approx \trianglelefteq_v(S_j)$. On peut alors suivre le même raisonnement fait pour T et S pour déduire que l'équivalence 12.14 est vraie pour tout k, $1 \leq k \leq 6$.

Suivant les règles de sémantique opérationnelle de l'opérateur $\oplus$ on déduit alors que $\text{exec}(T) = \text{exec}(S)$. Il reste à prouver que $\mathcal{R}$ est fermée aux transitions.

Notons par T' et S' les successeurs de T et S, respectivement, après l'exécution d'une action quelconque $a(v)$. Puisque le cas d'un blocage est trivial, nous supposons que $a \neq \delta$. Ces transitions de T et S sont définies par les règle de sémantique opérationnelle $\mathbf{CR}_1\mathbf{CR}_3, \mathbf{CR}_6$, et $\mathbf{CR}_7$

Considérons d'abord la règle $\mathbf{CR}_1$. Selon cette règle, a est une action d'entrée ic. On a alors que

$$\begin{aligned} T' &= \bigoplus_{i\in I} T'_i \text{ et} \\ S' &= \bigoplus_{i\in I} S'_i, \end{aligned}$$

où

$$\begin{aligned} I &= \{i \mid (\exists T'_i)(T_i \overset{ic(v)}{\rightarrow} T'_i)\} \\ &= \{i \mid (\exists S'_i)(S_i \overset{ic(v)}{\rightarrow} S'_i)\}, \end{aligned}$$

et $T'_i \approx S'_i$. Donc, par définition de la relation $\mathcal{R}$, $(T', S') \in \mathcal{R}$.

La preuve pour la règle $\mathbf{CR}_3$ est triviale: on a que $T' = \mathcal{S}' = \uparrow$, donc $(T', S') \in \approx \subseteq \mathcal{R}$.

Dans le cas de la règle $\mathbf{CR}_6$, tous les sous-termes T_j, $j \in \{1, \ldots, m\} \setminus \{i\}$, sauf un sous-termes T_i, bloquent au temps v et T_i peut exécuter une action $a(v)$, différente de δ. De même pour les sous-termes de S aussi, puisque $S_i \approx T_i$ et $S_j \approx T_j$

pour tout $j \in \{1, \ldots, m\} \setminus \{i\}$. Donc, $\text{succ}(a(v), T) = T'_i$ et $\text{succ}(a(v), S) = S'_i$. Or, puisque $T_i \approx S_i$, il existe des termes $T'_i = \text{succ}(a(v), T_i)$ et $S'_i = \text{succ}(a(v), S_i)$ tels que $S'_i \approx T'_i$. Donc, $(T', S') \in \approx \subseteq \mathcal{R}$, ce qu'il fallait montrer.

Finalement, considérons la règle $\mathbf{CR}_7$. Selon cette règle de sémantique opérationnelle, au moins un et tout au plus $m-2$ sous-termes T_i de T exécutent l'action de blocage $\delta(v_i)$, où $v_i \in D$. Notons par

$$v^\delta \stackrel{\text{not}}{=} \max_i \{v_i\}$$

le temps du dernier de ces blocages. Puisque $S_i \approx T_i$ pour tout $i \in \{1, \ldots, m\}$, le même raisonnement et les mêmes notations s'appliquent pour S aussi.

Introduisons les notations suivantes:

$$T^\delta \stackrel{\text{not}}{=} \bigoplus_{j \in J^\delta} \trianglelefteq_{v^\delta}(T_j)$$
$$S^\delta \stackrel{\text{not}}{=} \bigoplus_{j \in J^\delta} \trianglelefteq_{v^\delta}(S_j),$$

où J^δ et le sous-ensemble d'indices des sous-termes T_j de T, respectivement S_j de S qui ne bloquent pas:

$$J^\delta \stackrel{\text{not}}{=} \{j \in \{1, \ldots, m\} \mid \delta \notin \text{exec}(T_j)\}.$$

Alors, puisque $T_j \approx S_j$ et que $\approx$ est une congruence pour l'opérateur $\trianglelefteq$, on déduit que $(T^\delta, S^\delta) \in \mathcal{R}$. Or, nous avons déjà prouvé que pour toute règle de sémantique opérationnelle $\mathbf{CR}_k$ avec $1 \leq k \leq 6$, et pour toute paire de termes $(T, S) \in \mathcal{R}$ l'hypothèse $\mathbf{Hyp}_k(T)$ et vraie si et seulement si $\mathbf{Hyp}_k(S)$ est vraie et que les conclusions respectives $\mathbf{Cons}_k$ décrivent des transitions $T \xrightarrow{a(v)} T'$ et $S \xrightarrow{a(v)} S'$ telles que $(T', S') \in \mathcal{R}$. Donc, ceci est vrai pour la paire (T^δ, S^δ) aussi et nous pouvons conclure que $(T', S') \in \mathcal{R}$ et que $\mathcal{R}$ est une bisimulation ACTC. ∎

12.3.5.9 Congruence de $\approx$ par rapport à l'opérateur []

Soit $\mathcal{R}_0 \subseteq \mathcal{L} \times \mathcal{L}$ la relation contenant toutes les paires de termes $(S, T) \in \mathcal{L} \times \mathcal{L}$ et uniquement celles telles que:

$$\begin{aligned} S &= [\mathcal{C}, \epsilon](S_1, \ldots, S_m) \\ T &= [\mathcal{C}, \epsilon](T_1, \ldots, T_m), \end{aligned}$$

avec $m \geq 2$ et

$$S_i \approx T_i, \forall i : 1 \leq i \leq m.$$

On remarque que S et T ont le même ensemble d'actions de communication $\mathcal{C}$, ainsi que la même fonction de communication ϵ. Donc, pour tout $c \in \mathcal{C}$, les instances de c dans $\epsilon(c)$ sont en même temps des actions des sous-termes T_i et des sous-termes S_i. Formellement,

$$\epsilon(c) \subseteq \bigcup_{i=1}^{m} \mathrm{Act}(T_i) \text{ et } \epsilon(c) \subseteq \bigcup_{i=1}^{m} \mathrm{Act}(S_i).$$

De plus, puisque les ensembles d'actions des sous-termes de toute composition parallèle sont disjoints par définition, pour toute action $a_i \in \epsilon(c)$ il existe un sous-terme unique T_i tel que $a_i \in \mathrm{Act}(T_i)$, notation

$$a_i = \epsilon(c, T_i) \stackrel{\text{not.}}{=} \epsilon(c) \cap \mathrm{Act}(T_i).$$

Notons par $\epsilon^*(c, T)$ l'ensemble de sous-termes de T qui communiquent via l'action de communication c:

$$\epsilon^*(c, T) = \{T_i \in \mathcal{L} \mid T_i = \mathrm{sous-terme} T, \mathrm{Act}(T_i) \cap \epsilon(c) \neq \emptyset\}$$

On peut alors appliquer le même raisonnement aux sous-termes de S. Puisque $T_i \approx S_i$ nous pouvons déduire que pour toute action $c \in \mathcal{C}$ et pour tout i,

$$T_i \in \epsilon^*(c, T) \Leftrightarrow S_i \in \epsilon^*(c, S).$$

Finalement, nous pouvons déduire que les ensembles d'indices $I(a)$ et $J(a)$ définies dans le tableau 6.6 sont identiques pour T et pour S:

$$\begin{aligned} I(a) &= \{i \mid T_i \in \epsilon^*(a,T)\} = \{i \mid S_i \in \epsilon^*(a,T)\} \\ J(a) &= \{j \mid T_j \notin \epsilon^*(a,T)\} = \{j \mid S_j \notin \epsilon^*(a,T)\}. \end{aligned}$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation définie comme suit

$$\mathcal{R} = \mathcal{R}_0 \cup \approx .$$

Nous prouverons que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire il suffit de considérer une paire quelconque $(S,T) \in \mathcal{R}_0$ et de prouver (S,T) satisfait aux trois conditions énoncées dans la définition 8.1 (voir le raisonnement présenté dans la section 8.2).

Les termes S et T ont, évidemment, le même temps de démarrage:

$$\alpha(S) \overset{\text{Lemme8.9}}{=} \alpha(T).$$

Le fait que $\Gamma(T) = \Gamma(S)$ suit directement de la définition de la fonction Γ et parce que, selon la définition de $\approx$, $\Gamma(T_i) = \Gamma(S_i)$ pour tout i.

Montrons maintenant que pour toute transition $T \overset{a(v)}{\rightarrow} T'$ il existe une transition $S \overset{a(v)}{\rightarrow} S'$ avec $(S',T') \in \mathcal{R}$. Ce n'est pas nécessaire de prouver la réciproque de cette propriété puisque $\mathcal{R}$ est, par définition, commutative.

Supposons d'abord que $a \neq \delta$. Alors, la seule règle de sémantique opérationnelle qui s'applique est $\mathbf{Com}_1$. Deux conditions doivent être satisfaites pour que T puisse exécuter l'action temporisée $a(v)$. Premièrement, tous les sous-termes T_i avec $i \in I(a)$ doivent effectuer une transition $T_i \overset{a_i(v)}{\rightarrow} T_i'$, où $a_i = \epsilon(a,T_i)$ est l'action interne de T_i qui correspond à a. Deuxièmement, tous les sous-termes T_j avec $j \in J(a)$ doivent pouvoir attendre jusqu'au temps v. Puisque $S_i \approx T_i$ pour tout i, $1 \leq i \leq n$, on déduit directement à partir du théorème 8.6 et de la définition de la bisimulation forte que l'hypothèse de $\mathbf{Par}_1$ est vraie pour T si et seulement si elle est vraie pour S.

Donc, T exécute $a(v)$, si et seulement si S exécute $a(v)$. De plus, pour tout $i \in I(a)$ il existe un successeur $S'_i = \text{succ}(a_i, S_i)$ tel que $S'_i \approx T'_i$.

Il reste à montrer que les successeurs respectifs S' et T' après l'exécution de $a(v)$ sont tels que $(S', T') \in \mathcal{R}$. Or, selon $\mathbf{Par}_1$ T' et S' sont:

$$\begin{aligned} T' &= [\mathcal{C}, \epsilon]_{\substack{i \in I(a) \\ j \in J(a)}} (T'_i, \trianglelefteq_v(T_j)), \\ S' &= [\mathcal{C}, \epsilon]_{\substack{i \in I(a) \\ j \in J(a)}} (S'_i, \trianglelefteq_v(S_j)). \end{aligned}$$

On remarque que $S'_i \approx T'_i$ et $\trianglelefteq_v(S_j) \approx \trianglelefteq_v(T_j)$ (congruence de $\approx$ par rapport à $\trianglelefteq$). Donc, par définition, $(S', T') \in \mathcal{R}$.

Les cas où $a = \delta$ sont similaires. Nous concluons, donc, que $\approx$ est une congruence par rapport à l'opérateur [ ]. ∎

12.3.6 Preuve du théorème 8.11

Dans les sous-sections suivantes nous prouverons que:

1. $(\forall v \geq V_0)(\forall T, S \in \mathcal{L})(T \sim S \Rightarrow \trianglelefteq_v(T) \sim \trianglelefteq_v(S))$ et $(T \simeq S \Rightarrow \trianglelefteq_v(T) \simeq \trianglelefteq_v(S))$;
2. $(\forall v \geq V_0)(\forall T, S \in \mathcal{L})(T \sim S \Rightarrow \triangleleft_v(T) \sim \triangleleft_v(S))$ et $(T \simeq S \Rightarrow \triangleleft_v(T) \simeq \triangleleft_v(S))$;
3. $\sum_{i \in I} T_i \sim \sum_{i \in I} S_i$,

12.3.6.1 Congruence de $\sim$ par rapport à $\trianglelefteq$

Soient $T, S \in \mathcal{L}, i \in I$ des termes quelconque. Nous voulons prouver que pour tout $v \in D$

$$T \sim S \Rightarrow \trianglelefteq_v(T) \sim \trianglelefteq_v(S).$$

Selon le Théorème fondamentale d'équivalence 4.8, pour tous termes $T, S \in \mathcal{L}$

$$T \sim S \Leftrightarrow \{T\} \simeq \{S\}.$$

Donc, pour prouver le théorème il suffit de définir relation $\mathcal{R} \subseteq 2^{\mathcal{L}} \times 2^{\mathcal{L}}$, telle que

$$\mathcal{R} = \{(\{\trianglelefteq_v(T_1)\}, \{\trianglelefteq_v(S_1)\}) \mid (T_1, S_1) \in \sim, v \in D\} \cup \simeq .$$

et de prouver que $\mathcal{R}$ est une bisimulation forte.

Soit $(\{T\}, \{S\}) \in \mathcal{R}$ une paire quelconque de $\mathcal{R}$ telle qu'il existe une paire de termes $(T_1, S_1) \in \sim$ et

$$\begin{aligned} T &= \trianglelefteq_v(T_1) \text{ et} \\ S &= \trianglelefteq_v(S_1). \end{aligned}$$

D'abord, par définition,

$$\text{exec}(\{T\}) = \text{exec}(T) \ \text{exec}(\{S\}) = \text{exec}(S).$$

La sémantique de l'opérateur $\trianglelefteq_v$ et telle que

$$\begin{aligned} \text{exec}(T) &= \text{exec}^{\geq v}(T_1), \text{ si } \text{exec}^{\geq v}(T_1) \neq \emptyset, \text{ et} \\ \text{exec}(T) &= \{\delta(v)\}, \text{ si } \text{exec}^{\geq v}(T_1) = \emptyset. \end{aligned}$$

De même pour S:

$$\begin{aligned} \text{exec}(S) &= \text{exec}^{\geq v}(S_1), \text{ si } \text{exec}^{\geq v}(S_1) \neq \emptyset, \text{ et} \\ \text{exec}(S) &= \{\delta(v)\}, \text{ si } \text{exec}^{\geq v}(S_1) = \emptyset. \end{aligned}$$

Puisque $T_1 \sim S_1$ c'est évident que

$$\text{exec}(T_1) = \text{exec}(S_1) \ \ \text{exec}(T) = \text{exec}(S).$$

Donc,

$$\begin{aligned} \text{exec}^{\geq v}(S) = \emptyset &\Leftrightarrow \text{exec}^{\geq v}(T) = \emptyset \\ &\Leftrightarrow \text{exec}(\{S\}) = \{\delta(v)\} \\ &\Leftrightarrow \text{exec}(\{T\}) = \{\delta(v)\}. \end{aligned}$$

De plus, lorsque $\{T\}$ et $\{S\}$ bloquent, ils transitent vers $\{\downarrow\}$. Puisque la relation $\simeq$ est réflexive, on déduit que $(\{\downarrow\}, \{\downarrow\}) \in \simeq$. Donc, les situations de blocage satisfont aux conditions données dans la définition d'une bisimulation forte.

Supposons maintenant que $\text{exec}^{\geq v} \neq \emptyset$. On a alors que

$$\text{exec}(\{T\}) = \text{exec}(T) = \text{exec}(T_1) = \text{exec}(S_1) = \text{exec}(S). = \text{exec}(\{S\}).$$

De plus,

$$\begin{aligned} \text{exec}(\{T\}) = \text{exec}(T) \quad &\text{et} \quad \text{exec}(T_1) = \text{exec}(\{T_1\}) \\ \text{exec}(\{S\}) = \text{exec}(S) \quad &\text{et} \quad \text{exec}(S_1) = \text{exec}(\{S_1\}) \end{aligned}$$

Donc, pour toute transition

$$\{T\} \stackrel{a(w)}{\rightarrow} \mathcal{T}$$

avec $w \geq v$ il existe une transition

$$\{S\} \stackrel{a(w)}{\rightarrow} \mathcal{S}$$

Il reste à prouver que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$. Par définition, pour tout $w \geq v$ on a que

$$\begin{aligned} \mathcal{T} &= \{T' \mid T \stackrel{a(w)}{\rightarrow} T'\} \\ \mathcal{S} &= \{S' \mid S \stackrel{a(w)}{\rightarrow} S'\}. \end{aligned}$$

Or, selon les règles de SOp, pour toute transition

$$T \stackrel{a(w)}{\rightarrow} T' \text{ respectivement } S \stackrel{a(w)}{\rightarrow} S',$$

avec $w \geq v$ il doit exister une transition

$$T_1 \stackrel{a(w)}{\rightarrow} T_1' \text{ respectivement } S_1 \stackrel{a(w)}{\rightarrow} S_1'$$

et, de plus, $T' = T_1'$ et $S' = S_1'$. Donc,

$$\mathcal{T} = \{T' \mid T \stackrel{a(w)}{\rightarrow} T'\}$$

$$\begin{aligned} &= \{T' \mid T_1 \stackrel{a(w)}{\rightarrow} T'\} \text{ et} \\ \mathcal{S} &= \{S' \mid T \stackrel{a(w)}{\rightarrow} T'\} \\ &= \{S' \mid T_1 \stackrel{a(w)}{\rightarrow} T'\} \end{aligned}$$

Lorsqu'on considère alors les transitions par parties de $\{T_1\}$ et $\{S_1\}$

$$\{T_1\} \stackrel{a(w)}{\rightarrow} \mathcal{T}_1 \text{ et } \{S_1\} \stackrel{a(w)}{\rightarrow} \mathcal{S}_1,$$

on remarque alors que $\mathcal{T} = \mathcal{T}_1$ et que $\mathcal{S} = \mathcal{S}_1$. Or, $T_1 \sim S_1$. Selon le théorème 4.8, on a alors que

$$\{T_1\} \simeq \{S_1\}.$$

Par définition d'une bisimulation forte et puisque tout système de transitions par parties est déterministe, on déduit que

$$\mathcal{T}_1 \simeq \mathcal{S}_1.$$

Donc,

$$(\mathcal{T}, \mathcal{S}) \in \simeq \subseteq \mathcal{R},$$

ce qu'il fallait montrer. ∎

12.3.6.2 Congruence de $\sim$ par rapport à $\sum$

Selon le Théorème fondamentale d'équivalence 4.8, pour tous termes $T, S \in \mathcal{L}$

$$T \sim S \Leftrightarrow \{T\} \simeq \{S\}.$$

Soient $T_i, S_i \in \mathcal{L}, i \in I$ des termes quelconque. Nous voulons prouver que

$$(\forall i \in I)(\{T_i\} \simeq \{S_i\}) \Rightarrow \{\sum_{i \in I} T_i\} \simeq \{\sum_{i \in I} S_i\}.$$

Pour ce faire nous définissons la relation $\mathcal{R} \subseteq 2^{\mathcal{L}} \times 2^{\mathcal{L}}$, telle que

$$\mathcal{R} = \{(\{\sum_{i \in I} T_i\}, \{\sum_{i \in I} S_i\}) \mid (T_i, S_i) \in \sim, i \in I\} \cup \simeq$$

et montrons que $\mathcal{R}$ est une bisimulation forte. Considérons d'abord les transitions

$$\{\sum_{i\in I} T_i\} \stackrel{a(v)}{\rightarrow} \mathcal{T},$$

où $a \neq \delta$. Alors

$$\begin{aligned}
\mathcal{T} &= \{T' \mid \sum_{i\in I} T_i \stackrel{a(v)}{\rightarrow} T'\} \\
&= \{T' \mid (\exists i \in I)(T_i \stackrel{a(v)}{\rightarrow} T')\} \\
&= \bigcup_{i\in I} \{T_i' \mid T_i \stackrel{a(v)}{\rightarrow} T_i'\} \\
&\stackrel{\text{not.}}{=} \bigcup_{i\in I} \mathcal{T}_i,
\end{aligned}$$

où $\mathcal{T}_i$ est le successeur de $\{T_i\}$ après l'exécution de $a(v)$.

Or, $T_i \sim S_i$ et, donc, $\{T_i\} \simeq \{S_i\}$. Donc, $\{T_i\}$ peut exécuter $a(v)$ si et seulement si $\{S_i\}$ le peut aussi et leurs successeurs respectifs $\mathcal{T}_i$ et $\mathcal{S}_i$ sont bisimilaires (ils ont un unique successeur, puisque tout système de transitions par parties est déterministe). Donc, il existe une transition

$$\{\sum_{i\in I} S_i\} \stackrel{a(v)}{\rightarrow} \mathcal{S},$$

où

$$\mathcal{S} = \bigcup_{i\in I} \mathcal{S}_i.$$

Puisque des unions d'ensembles bisimilaires de termes sont bisimilaires (proposition 4.6), on déduit que

$$(\{\sum_{i\in I} T_i\}, \{\sum_{i\in I} S_i\}) \in \simeq \subseteq \mathcal{R},$$

ce qu'il fallait montrer.

Il reste maintenant à considérer les situations de blocage. $\{\sum_{i\in I} T_i\}$ exécute $\delta(v)$ et transite ensuite vers $\{\downarrow\}$ si et seulement s'il existe au moins un sous-terme T_j bloque au temps v et tous les autres sous-termes soit bloquent aussi au temps v, ou bien ne peuvent pas attendre jusqu'au temps v. Mais, pour tout $i \in I$, $T_i \sim S_i$. Donc, si T_i exécute $\delta(v)$, alors S_i aussi peut l'exécuter et, selon la proposition 8.2, les prédicats

Wait(v, T_i) et Wait(v, S_i) sont équivalents. Par conséquent $\{\sum_{i\in I} S_i\}$ aussi exécute $\delta(v)$ et transite ensuite vers $\{\delta\}$. Or, puisque $\simeq$ est réflexive, $(\downarrow, \downarrow) \in \simeq \subseteq \mathcal{R}$, ce qu'il fallait prouver. ∎.

12.4 Preuves du chapitre 9

Nour prouverons ici la proposition 9.3 et le théorème de cohérence de **A**.

12.4.1 Preuve de la propsition 9.3

Rappelons l'énoncé de la proposition:

Soient $\times, + : E^* \longrightarrow E$ *deux opérateurs d'arité arbitraire quelconques. Si* $\times$ *et* $+$ *sont associatifs alors*

1. $\times$ *est commutatif* $\Leftrightarrow$ $\times^2$ *est commutatif;*
2. u *est un élément identité pour* $\times$ $\Leftrightarrow$ u *est un élément identité pour* $\times^2$*;*
3. v *est un élément absorbant pour* $\times$ $\Leftrightarrow$ v *est un élément absorbant pour* $\times^2$*;*
4. $\times$ *est distributif par rapport à* $+$ $\Leftrightarrow$ $\times^2$ *est distributif par rapport à* $+^2$*;*
5. $\times$ *est réductible* $\Leftrightarrow$ $\times^2$ *est réductible;*
6. $\times$ *est idempotent* $\Leftrightarrow$ $\times^2$ *est idempotent.*

Preuve de l'égalité 1:

Pour tout $1 \leq i < m$ on a que

$$\begin{aligned} \times(a_1, \ldots, a_i, a_{i+1}, \ldots, a_m) &\overset{\text{assoc.}}{=} \times(a_1, \ldots, \times(a_i, a_{i+1}), \ldots, a_m) \\ &\overset{2-\text{comm.}}{=} \times(a_1, \ldots, \times(a_{i+1}, a_i), \ldots, a_m) \\ &\overset{\text{assoc.}}{=} \times(a_1, \ldots, a_{i+1}, a_i, \ldots, a_m). \end{aligned}$$

On utilise ensuite un résultat connu: toute permutation $\sigma \in \mathcal{P}_m$ peut être écrite comme une composition d'inversions $\sigma_{(i,i+1)} \in \mathcal{P}_m$. ∎

Preuve des l'égalités 2 et 3:

Les preuves des deux égalités sont analogues. Considérons juste l'égalité 2 pour l'opérateur $\times$. Pour tout $1 \leq i < m$ on a que

$$\begin{aligned} \times(a_1, \ldots, a_i, v, a_{i+1}, \ldots, a_m) &\overset{\text{assoc}}{=} \times(a_1, \ldots, \times(a_i, v), a_{i+1}, \ldots, a_m) \\ &\overset{\text{idem2}}{=} \times(a_1, \ldots, a_i, a_{i+1}, \ldots, a_m). \end{aligned}$$

Les cas $i = m$ et $\times(v, a_1, \ldots, a_m)$ sont analogues. ∎

Preuve de l'égalité 4:

Prouvons d'abord par induction sur n que

$$P(n) : \times(a, +(b_1, \ldots, b_n)) = +(\times(a, b_1), \ldots, \times(a, b_n)),$$

pour tout $n \geq 2$. La base d'induction est le cas $P(2)$ et est satisfaite trivialement par hypothèse (2–distributivité). Montrons maintenant que $P(n)$ implique $P(n+1)$:

$$\begin{aligned} \times(a, +(b_1, \ldots, b_n, b_{n+1})) &\overset{\text{assoc. } +}{=} \times(a, +(b_1, \ldots, b_{n-1} + (b_n, b_{n+1}))) \\ &\overset{P(n)}{=} +(\times(a, b_1), \ldots, \times(a, b_{n-1}), \times(a, +(b_n, b_{n+1}))) \\ &\overset{P(2)}{=} +(\times(a, b_1), \ldots, \times(a, b_{n-1}), +(\times(a, b_n), \times(a, b_{n+1}))) \\ &\overset{\text{assoc. } +}{=} +(\times(a, b_1), \ldots, \times(a, b_{n+1})), \end{aligned}$$

ce qu'il fallait montrer. On déduit alors directement de la propriété d'associativité de l'opérateur $\times$ que $\times$ est distributif par rapport à $+$. ∎

Preuve de l'égalité 5:

Nous voulons prouver que si $\times$ est 2-réductible alors

$$\times(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_m) = \times(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_m) \;\Rightarrow\; a_i = b_i, \tag{12.15}$$

pour tout $m \geq 2$, tout $i : 1 \leq i \leq m$, et tous $a_1, \ldots, a_m, b_i \in E$. La preuve se fait par induction sur m. Notons l'implication 12.15 par $P(m)$. La base de l'induction

est l'implication $P(2)$ qui est trivialement satisfaite par l'énoncé de la proposition. L'hypothèse d'induction est que $P(m)$ et vrai pour tout $m \leq k$, où $k \geq 2$ est un entier quelconque. Prouvons alors que $P(k+1)$ est vrai aussi.

$$\begin{aligned} \times(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_{k+1}) &= \times(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_{k+1}) \overset{\text{assoc}}{\Rightarrow} \\ \times(\times(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_k), a_{k+1}) &= \times(\times(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_k), a_{k+1}) \overset{\text{hyp. d'ind.}}{\Rightarrow} \\ \times(a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_k) &= \times(a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_k) \overset{\text{hyp. d'ind}}{\Rightarrow} \\ a_i &= b_i. \end{aligned}$$

Ces équations s'appliquent si $i < k+1$. Dans le cas où $i = k+1$ on utilise le fait que l'opération binaire $\times$ est droite-réductible et on suit le même raisonnement. ∎

Preuve de l'égalité 6:

La preuve est triviale et suit directement des définitions. ∎

12.4.2 Cohérence de A

Rappelons l'énoncé du théorème:

Le système d'axiomes **A** *est cohérent avec la relation d'équivalence de termes* $\approx$. *Formellement, pour toute paire de termes* $T, S \in \mathcal{L}$,

$$\mathbf{A} \vdash T = S \Rightarrow T \approx S.$$

Dans la section 9.2.5 nous avons prouvé que les règles d'inférence de la logique équationnelle s'applique à la relation $\approx$. Dans les sous-sections suivantes nous montrerons que chaque axiome de **A** reste vrai si on remplace $=$ par $\approx$.

12.4.2.1 Cohérence de Dec1

Nous allons prouver que pour tout terme $T \in \mathcal{L}_\beta$ $v \geq v'$ des instants temporels, si $v \geq v' \geq V_0$, alors

1. $\lhd_v(\lhd_{v'}(T)) \approx \lhd_v(T)$;

2. $\trianglelefteq_v(\trianglelefteq_{v'}(T)) \approx \trianglelefteq_v(T)$.

Les preuves des deux équivalences sont analogues. Nous ne présenterons que celle concernant l'opérateur $\lhd$.

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\lhd_v(\lhd_{v'}(T)), \lhd_v(T)) \mid T \in \mathcal{L}, V_0 \leq v' \leq v\} \cup \approx .$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation–ACTC. Selon le raisonnement présenté dans la section 8.2, il suffit de considérer une paire quelconque $(T_1, T_2) \in \mathcal{R}$ telle que

$$T_1 = \lhd_v(\lhd_{v'}(T)) \text{ et } T_2 = \lhd_v(T)$$

avec $T \in \mathcal{L}, V_0 \leq v' \leq v$ et de prouver que (T_1, T_2) satisfait aux conditions imposées dans la définition d'une bisimulation–ACTC.

Premièrement, par définition $\alpha(T_1) = \alpha(T_2) = v$. Deuxièmement, puisque $v' \leq v$ et que $\text{Act}(\lhd_v(T)) = \text{Act}(T)$,

$$\begin{aligned} \Gamma(T_1) &= (\Gamma^{>v'}(T))^{>v} \\ &= \Gamma^{>v}(T) \\ &= \Gamma(T_2). \end{aligned}$$

Il reste à prouver que $\mathcal{R}$ est une bisimulation forte. Nous allons d'abord montrer que $\text{exec}(T_1) = \text{exec}(T_2)$. On remarque alors que, selon les règles de SOp,

$$\begin{aligned} \text{exec}(\lhd_v(T)) &= \text{exec}^{>v}(T), \text{ si } \text{exec}^{>v}(T) \neq \emptyset, \\ &= \{\delta(v)\}, \text{ sinon.} \end{aligned}$$

Il y a trois cas à considérer: soit (1) $\text{exec}^{>v'}(T) = \emptyset$, ou bien (2) $\text{exec}^{>v'}(T) \neq \emptyset$ et $\text{exec}^{>v}(T) = \emptyset$, ou encore (3) $\text{exec}^{>v} \neq \emptyset$.

Dans le premier cas on a que $\text{exec}(\lhd_{v'}(T)) = \{\delta(v')\}$, donc $\text{exec}(T_1) = \{\delta(v)\}$. En même temps $\text{exec}(T_2) = \{\delta(v)\}$ et $\text{succ}(T_1) = \text{succ}(T_2) = \{\downarrow\}$.

Dans le cas (2) on déduit que $\text{exec}(\lhd_{v'}(T) = \text{exec}^{>v'}(T)$. Donc, puisque $\text{exec}^{>v}(T) = \emptyset$ et que $v' \leq v$, alors $\text{exec}(T_1) = \{\delta(v)\}$. Par définition, alors $\text{exec}(T_2) = \{\delta(v)\}$ et $\text{succ}(T_1) = \text{succ}(T_2)\{\downarrow\}$.

Finalement, dans la situation (3) on a que $\text{exec}(T_1) = \text{exec}(T_2) = \text{exec}^{>v}(T)$. Or, à toute transition $T_1 \stackrel{a(v'')}{\rightarrow} T'$ avec $v'' > v$ correspond une transition $T \stackrel{a(v'')}{\rightarrow} T'$, et la même chose est donc valide pour T_2 aussi. Donc, après l'exécution d'une même action temporisée, les termes T_1 et T_2 transitent vers le même terme. Or toute paire $(T', T') \in \approx \subseteq \mathcal{R}$. ∎.

12.4.2.2 Cohérence de Dec2

Le but de cette section est de prouver la cohérence de l'axiome **Dec2**. Pour ce faire nous avons besoin du résultat intermédiaire suivant:

Lemme 12.1 *Pour tout terme $T \in \mathcal{L}$ on a que*

$$\Gamma^{\geq \alpha(T)}(T) = \Gamma(T).$$

Preuve: L'idée de base de la preuve est la suivante: on va prouver que pour tout vecteur $\vec{v} \in \Gamma(T)$ les variables temporelles des actions de T prennent uniquement des valeurs supérieures ou égales à $\alpha(T)$.

La preuve sera faite par induction structurelle. Tous les termes de $\mathcal{L}$ sont composés à partir d'actions temporisées $a(t_a)$, avec $a \in A_\beta$. Or,

$$\Gamma(a(t_a)) \stackrel{\text{def}}{=} \text{Sol}(t_a \geq V_0).$$

Donc, toutes les solutions $(t_a = v) \in \Gamma(a(t_a))$ sont telles que $v \geq \alpha(a(t_a)) = V_0$.

Le pas d'induction consiste à prouver que si la proposition est vrai pour tous les sous-termes d'un terme T, alors elle est vrai pour T aussi.

Si

$$T = a(t_a).S,$$

avec $S \neq \surd$, alors $\alpha(a(t_a).S) = \alpha(S) = V_0$ et

$$\Gamma(a(t_a).S) \stackrel{\text{def}}{=} \Gamma(S) \cap \text{Sol}(t_a \geq V_0 \bigwedge_{b \in \text{Act}(S)} t_b \geq t_a).$$

Évidemment alors $\Gamma(a(t_a).S)$ n'admet que des valeurs supérieures ou égales à V_0.

Soit

$$T = \|(S_1, \ldots, S_m).$$

Alors, par définition,

$$\Gamma(T) \stackrel{\text{def}}{=} \bigcap_{i=1}^{m} \Gamma(S_i).$$

et selon la définition de α,

$$V_0 \leq \alpha(T) \leq \alpha(S_i)$$

pour tout $i \in \{1, \ldots, m\}$. Selon l'hypothèse d'induction: toutes les solutions de $(\Gamma(S_i)$ sont supérieures ou égales à $\alpha(S_i)$, donc implicitemenet supérieures ou égales à $\alpha(T)$, ce qu'il fallait prouver.

Si

$$T = \trianglelefteq_v(S),$$

alors $\alpha(T) = v$ et $\Gamma(T) = \Gamma(S)^{\geq v}$. Forcement alors, toutes les valeurs temporelles v_i de toute solution de $\Gamma(T)$ sont telles que $v_i \geq v = \alpha(T)$. La preuve pour $T = \trianglelefteq_v(S)$ est analogue à celle pour $T = \triangleleft_v(T)$.

La preuve pour les opérateurs réactifs ROp, où ROp = Max ou ROp = Min, est triviale: par définition, $\Gamma(\text{ROp}(S)) = \Gamma(S)$ et $\text{exec}(\text{ROp}(S)) \subseteq \text{exec}(T)$. Donc, la proposition est une conséquence directe de l'hypothèse d'induction.

Dans le cas où

$$T = \theta : S$$

on a que $\alpha(T) = \alpha(S)$ (par définition) et

$$\Gamma(T) \stackrel{\text{def}}{=} \text{Sol}(\theta) \cap \text{Sol}(S).$$

Donc, toute solution de $\Gamma(\theta : S)$ est aussi une solution de $\Gamma(S)$, qui, selon l'hypothèse d'induction, n'admet que des valeurs $v \geq \alpha(S) = \alpha(T)$.

Considérons les termes hiérarchiques maintenant. Soit

$$T = +(T_1, \ldots, T_m).$$

Par définition

$$\begin{aligned} \Gamma(T) &\stackrel{\text{def}}{=} \bigcup_{i=1}^{m} \Gamma(T_i) \text{ et} \\ \Gamma^{\geq\alpha(T)}(T) &\stackrel{\text{def}}{=} \bigcup_{i=1}^{m} \Gamma^{\geq\alpha(T)}(T_i). \end{aligned}$$

Selon l'hypothèse d'induction, pour tout $i \in \{1, \ldots, m\}$

$$\Gamma(T_i) = \Gamma^{\geq\alpha(T_i)}(T_i).$$

De plus, par définition, $\alpha(T) \leq \alpha(T_i)$. Donc,

$$\Gamma^{\geq\alpha(T_i)}(T_i) = \Gamma^{\geq\alpha(T)}(T_i).$$

On déduit que

$$\Gamma^{\geq\alpha(T)} = \Gamma(T),$$

ce qu'il fallait montrer.

Supposons maintenant que

$$T = \text{Seq}(T_1, T_2).$$

Alors, par définition,

$$\alpha(T) = \alpha(T_1) \stackrel{\text{not.}}{=} \alpha.$$

De plus, par définition,

$$\Gamma(\text{Seq}(T_1, T_2)) = \Gamma(T_1).$$

Mais, selon l'hypothèse d'induction, tout vecteur $\vec{v_1} \in \Gamma(T_1)$ des valeurs $v_i \geq \alpha(T_1) = \alpha(T)$, ce qu'il fallait montrer.

Le cas ou

$$T = \text{Loop}(S)$$

est trivial, puisque $\alpha(T) = \alpha(S)$ et, par définition, $\Gamma(T) = \Gamma(S)$.

Si

$$T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m),$$

alors

$$\Gamma(T) = (\bigcup_{i=1}^{m} \Gamma(T_i))_{t_c/t_a}, \forall c \in \mathcal{C}, \forall a \in \epsilon(c).$$

Donc, toute solution de $\Gamma(T)$ doit satisfaire les systèmes $\Gamma(T_i)$ aussi. Or, selon l'hypothèse d'induction, pour tout sous-terme T_i et tout $\vec{v} = (v_1, \ldots, v_n) \in \Gamma(T_i)$ et tout j $v_j \geq \alpha(T_j)$. De plus $\alpha(T_j) \geq \alpha(T) = \alpha$. Donc,

$$\Gamma^{\geq \alpha(T)}(T) = \Gamma(T).$$

La preuve du cas

$$T = \oplus(T_1, \ldots, T_m)$$

est identique au cas $T = +(T_1, \ldots, T_m)$.

Finalement, soit

$$T = \text{Ex}(T_n, T_c, T_e)$$

Alors,

$$\Gamma(\text{Ex}(T_n, T_c, T_e)) = \Gamma(T_n) \cap \Gamma(T_c),$$

et

$$\alpha(T) = \min(\alpha(T_n), \alpha(T_c)).$$

Donc, on peut appliquer le même raisonnement qu'á la composition parallèle non-communicante $\|$. ∎

Nous pouvons prouver maintenant que l'axiome **Dec2** est consistent, c'est á dire que

$$\trianglelefteq_{\alpha(T)}(T) \approx T.$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(T, \trianglelefteq_{\alpha(T)}(T)) \mid T \in \mathcal{L}\} \cup \approx .$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire il suffit de considérer une paire quelconque $(T, S) \in \mathcal{R}$ telle que $T \in \mathcal{L}$ et $S = \trianglelefteq_{\alpha(T)}(T)$ et prouver que (1) $\alpha(T) = \alpha(S)$, que (2) $\Gamma(T) = \Gamma(S)$, que (3) $\text{exec}(T) = \text{exec}(S)$, et, finalement, que lorsque T et S exécutent une même action $a(v)$ quelconque, leur successeurs respectifs, T' et S', forment une paire $(T', S') \in \mathcal{R}$.

La première condition $\alpha(T) = \alpha(S)$ est trivialement satisfaite. La deuxième est une conséquence immédiate du lemme 12.1. Nous avons déjà montré (proposition 7.7) que pour tout $T \in \mathcal{L}$

$$\text{exec}^{\geq \alpha(T)}(T) = \text{exec}(T).$$

Donc, $\text{exec}(T) = \text{exec}(S)$. Finalement on remarque que pour toute action $a(v) \in \text{exec}(T)$

$$\text{succ}(a(v), T) = \text{succ}(a(v), S).$$

Donc, nous pouvons déduire que $\mathcal{R}$ est une bisimulation ACTC, ce qu'il fallait prouver. ∎

12.4.2.3 Cohérence de Dec3

Nous devons considérer, un par un, tous les opérateurs $f \in \text{Op}$ et prouver que

1. $\triangleleft_v(f(T_1, \ldots, T_m)) \approx f(\triangleleft_v(T_1), \ldots, \triangleleft_v(T_m))$, et
2. $\trianglelefteq_v(f(T_1, \ldots, T_m)) \approx f(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_m))$,

pour tout $v \geq V_0$, tout $m \geq 1$, $T_1, \ldots, T_m \in \mathcal{L}$, et tout opérateur $f \in \text{Op}$ $m \geq 2$.

Nous ferons la preuve pour l'opérateur $\trianglelefteq$ seulement, car celle pour l'opérateur $\triangleleft$ est analogue.

Distributivité de $\trianglelefteq$ par rapport à l'opérateur $\|$

Nous prouvons la proposition seulement pour le cas $n = 2$. Le cas générale $n \geq 2$ est analogue.

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(\trianglelefteq_v(\|(T_1, T_2)), \|(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2))) \mid T_{1,2} \in \mathcal{L}_\beta, v \geq V_0\}.$$

Nous allons montrer que $\mathcal{R}$ est une bisimulation–ACTC.

Considérons une paire $(T, S) \in \mathcal{R}$, telle que

$$\begin{aligned} T &= \trianglelefteq_v(\|(T_1, T_2)) \\ S &= \|(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2)) \end{aligned}$$

Les égalités $\alpha(T) = \alpha(S)$ et $\Gamma(T) = \Gamma(S)$ sont triviales. Montrons que

1. $T \stackrel{a(v)}{\rightarrow} T' \Rightarrow (\exists S')(S \stackrel{a(v)}{\rightarrow} S' \wedge (T', S') \in \mathcal{R})$;
2. $S \stackrel{a(v)}{\rightarrow} S' \Rightarrow (\exists T')(T \stackrel{a(v)}{\rightarrow} T' \wedge (T', S') \in \mathcal{R})$.

Nous ne prouverons que la première propriété; la deuxième est analogue.

Supposons d'abord que T exécute $\delta(v)$ parce que $\|(T_1, T_2)$ ne peut effectuer aucune transition au temps v ou plus tard. Ceci n'est possible que si au moins un des sous-termes T_i ne peut exécuter aucune action à un temps supérieur ou égale à v. Par conséquent, le terme $\trianglelefteq_v(T_i)$ signale un blocage $\delta(v)$. Si l'autre sous-terme T_j ne peut exécuter aucune action $a(v')$ avec $v' \geq v$, alors $\trianglelefteq_v(T_j)$ aussi bloque au temps v et le terme S exécute $\delta(v)$. Autrement, si T_j peut effectuer au moins une transition au temps v ou plus tard, le prédicat $\text{Wait}_v(T_j)$ est vrai. Par conséquent, le prédicat $\text{Wait}(v, \trianglelefteq_v(T_j))$ est vrai aussi. Donc, le terme S bloque au temps v (règle $\|_4$).

Supposons maintenant qu'il existe une transition $\|(T_1, T_2) \stackrel{a(v')}{\rightarrow} T$ avec $v' \geq v$ et $a \in A$. Le terme T exécute alors $a(v')$ et transite vers T'. Ceci implique qu'un des sous-termes T_i effectue une transition $T_i \stackrel{a(v')}{\rightarrow} T_i'$ et que le prédicat $\text{Wait}_{v'}(T_j)$ est vrai, où T_j est l'autre sous-terme. Supposons, sans perte de généralité, que $i = 1$ et

$j = 2$. Puisque $v' \geq v$, le prédicat $\text{Wait}_{v'}(\trianglelefteq_v(T_2))$ est vrai aussi. En plus, $\trianglelefteq_v(T_i)$ peut exécuter $a(v')$. Donc, le terme S peut exécuter $a(v')$ et transite ensuite vers un terme S'.

Il reste à prouver que $(T', S') \in \mathcal{R}$. Il y a trois situations possibles: soit (1) $T_1' \neq \surd$, ou (2) $T_1' = \uparrow$ ou bien (3) $T_1' = \downarrow$.

Dans le premier cas $T' = \|(T_1', \trianglelefteq_{v'}(T_2))$ et $T' = \|(T_1', \trianglelefteq_{v'}(\trianglelefteq_v(T_2)))$. Mais, $v' \geq v$ (théorème 7.8), donc selon la Proposition 12.4.2.1, $\trianglelefteq_{v'}(\trianglelefteq_v(T_2)) \approx \trianglelefteq_{v'}(T_2)$. Selon la congruence de $\approx$ par rapport à $\|$ on conclut que $(T', S') \in \approx \subseteq \mathcal{R}$.

Dans le deuxième cas, $T' = \trianglelefteq_{v'}(T_2)$ et $S' = \trianglelefteq_{v'}(\trianglelefteq_v(T_2))$. Or, comme dans le cas précédent, ceci implique que $(T', S') \in \approx \subseteq \mathcal{R}$.

La cas (3) est trivial: on a alors que $T_1' = T = T' = \downarrow$, donc $(T, T') \in \approx \subseteq \mathcal{R}$ (théorème 8.1). Ceci résume la preuve de la proposition. ∎

Distributivité de $\trianglelefteq$ par rapport aux opérateurs réactifs

Nous considérons juste l'opérateur $\trianglelefteq$, puisque la preuve pour $\triangleleft$ est analogue. Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\begin{aligned} \mathcal{R} \;=\; & \{(\trianglelefteq_v(\text{ROp}(T_1)), \text{ROp}(\trianglelefteq_v(T_1))) \mid \\ & T_1 \in \mathcal{L}_\beta, \text{ROp} = \text{Min/Max}(o, H, [m, M], \cdot), v \geq V_0\} \cup \approx . \end{aligned}$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation–ACTC. Soient $S, T \in \mathcal{L}_\beta$ des termes telles que

$$\begin{aligned} S &= \trianglelefteq_v(\text{ROp}(o, H, [m, M], T_1)), \\ T &= \text{ROp}(o, H, [m, M], \trianglelefteq_v(T_1)). \end{aligned}$$

où $T_1 \in \mathcal{L}_\beta$ et $v \geq V_0$. Selon le raisonnement de la section 8.2, il suffit de montrer que la paire (S, T) a les trois propriétés énoncées dans la définition 8.1 d'une bisimulation–ACTC.

Premièrement, il suit directement des définitions que

$$\alpha(T) = \alpha(S) = \alpha(T_1)$$

et que

$$\Gamma(T) = \Gamma(S).$$

Il reste à prouver que la paire (T, S) satisfait aux deux conditions de la définition 4.2 d'une bisimulation forte.

Considérons une transition quelconque $T \stackrel{a(v')}{\rightarrow} T'$.

Si $a \neq \delta$, alors $v' \geq v$, $a \neq o$, et $a(v') \in \text{exec}(T_1)$. Mais alors S aussi peut exécuter $a(v')$. Notons par T' et S' les successeurs de T et S, respectivement, lors de l'exécution de $a(v')$. Il suffit alors de remarquer que $T' = S'$ et que, par conséquent, $(S', T') \in \approx \subseteq \mathcal{R}$ ($\approx$ est réflexive selon le théorème 8.1).

Si $a = \delta(v')$, avec $v' \geq v$, le blocage peut être causé par plusieurs scénarios différents. Premièrement, il peut être du au fait que le sous-terme T_1 bloque lui même au temps v'. Dans ce cas, puisque $v' \geq v$, on a que $\trianglelefteq_v(T_1)$ et S aussi bloquent au temps v'.

Deuxièmement, le blocage peut être du au fait que T ne peut exécuter aucune action au temps v ou plus tard. Notons par

$$\text{exec}^{\geq v}(T_1) = \{a(v') \mid a(v') \in \text{exec}(T_1), v' \geq v\}.$$

Si $\text{exec}^{\geq v}(T_1) = \emptyset$, alors d'une part

$$\text{exec}^{\geq v}(\text{ROp}(T_1)) = \text{exec}^{\geq v}(T_1) = \emptyset$$

et, par conséquent, T exécute $\delta(v)$. D'autre part, $\trianglelefteq_v(T_1)$, et par conséquent S aussi exécutent $\delta(v)$.

Troisièmement, supposons que $\text{exec}(T_1) = \{o\}$. Dans ce cas $\text{ROp}(o, H, [m, M], T_1)$ exécute $\delta(\alpha)$, où $\alpha = \alpha(\text{ROp}(o, H, [m, M], T_1)$. Selon le théorème de la monotonie temporelle 7.8, $\alpha \leq v$, donc S exécute $\delta(v)$. Il y a, alors, deux situations à considérer: soit (1) T_1 ne peut exécuter o au temps v ou plus tard, autrement dit $\text{exec}^{\geq v}(T_1) = \emptyset$, ou bien (2) il existe des instants $v' \geq v$ tels que $o(v') \in \text{exec}(T_1)$. La situation (1) a déjà été traitée. La situation (2) implique que $\trianglelefteq_v(T_1)$ ne bloque pas, mais que la

seule action qu'il peut exécuter est o. Par conséquent S bloque au temps $\alpha(S)$. Or, selon la proposition 7.2, $\alpha(S) = \alpha(\trianglelefteq_v(T_1)) = v$. Donc, S exécute $\delta(v)$.

Puisque nous avons couvert toutes les possibilités de blocage de T et S et avons prouvé que

$$\delta(v) \in \text{exec}(T) \Leftrightarrow \delta(v) \in \text{exec}(S),$$

et que lors d'un blocage $(S', T') = (\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$ (théorème 8.1), on peut déduire que (S, T) satisfait aux conditions d'une bisimulation forte et donc, que $\mathcal{R}$ est une bisimulation–ACTC. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur θ

Nous considérons juste l'opérateur $\triangleleft$ car la preuve pour $\trianglelefteq$ est analogue. Soient $T \in \mathcal{L}_\beta$ un terme, $\theta \in \Theta$ une contrainte, et $v \geq V_0$ un instant temporel quelconques. On va prouver que

$$\triangleleft_v(\theta : T) \;\approx\; \theta : \triangleleft_v(T).$$

Nous prouverons ici l'équivalence (1) concernant l'opérateur $\triangleleft$ seulement; la preuve de la deuxième équivalence est analogue.

Soi $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(\triangleleft_v(\theta : T), \theta : \triangleleft_v(T)) \mid v \geq V_0, \theta \in \Theta, T \in \mathcal{L}_\beta\}.$$

On va montrer que $\mathcal{R}$ est une bisimulation –ACTC.

Premièrement,

$$\begin{aligned} \alpha(\triangleleft_v(\theta : T)) &\stackrel{\text{def}}{=} v \\ \alpha(\theta : \triangleleft_v(T)) &\stackrel{\text{def}}{=} \alpha(\triangleleft_v(T)) \\ &\stackrel{\text{def}}{=} v. \end{aligned}$$

Deuxièmement,

$$\Gamma(\theta : \triangleleft_v(T)) \;=\; \text{Sol}(\theta) \;\cap \Gamma(\triangleleft_v(T))$$

$$
\begin{aligned}
&= \mathrm{Sol}(\theta) \cap \Gamma^{>v}(T) \\
&= \mathrm{Sol}(\theta) \bigcap_{a \in \mathrm{Act}(T)} [t_a > v] \cap \Gamma(T); \\
\Gamma(\lhd_v(\theta : T)) &= \bigcap_{a \in \mathrm{Act}(\theta:T)} [t_a > v] \cap \Gamma(\theta : T) \\
&= \bigcap_{a \in \mathrm{Act}(\theta:T)} [t_a > v] \cap \mathrm{Sol}(\theta) \cap \Gamma(T).
\end{aligned}
$$

Or, par définition $\mathrm{Act}(\theta : T) = \mathrm{Act}(T)$, donc

$$\Gamma(\lhd_v(\theta : T)) = \Gamma(\theta : \lhd_v(T)).$$

Il reste à prouver que $\mathcal{R}$ est une bisimulation. Puisque $\mathcal{R}$ n'est pas symétrique par définition, nous devons prouver que toute paire $(T_1, T_2) \in \mathcal{R}$ a les deux propriétés suivantes:

$$(\forall a(v), T_1')(T_1 \overset{a(v')}{\rightarrow} T_1' \quad \Rightarrow \quad (\exists T_2')(T_2 \overset{a(v')}{\rightarrow} T_2' \wedge (T_1', T_2') \in \mathcal{R}) \tag{12.16}$$

$$(\forall a(v'), T_2')(T_2 \overset{a(v')}{\rightarrow} T_2' \quad \Rightarrow \quad (\exists T_1')(T_1 \overset{a(v')}{\rightarrow} T_1' \wedge (T_1', T_2') \in \mathcal{R}). \tag{12.17}$$

Selon le raisonnement présenté dans la section 8.2, il suffit de considérer des paires $(T_1, T_2) \in \mathcal{R}$ telles que

$$T_1 = \lhd_v(\theta : T) \text{ et } T_2 = \theta : \lhd_v(T),$$

où $v \geq V_0, \theta \in \Theta, T \in \mathcal{L}_\beta$ sont, respectivement, un instant temporel, une contrainte descriptive et un terme feuille quelconques.

Prouvons d'abord que, pour tout $v' > v$ et toute action $a \in A_\beta$,

$$\mathrm{First}(a(v'), \theta : \lhd_v(T)) \Leftrightarrow \mathrm{First}(a(v'), \theta : T). \tag{12.18}$$

Cette équivalence sera utile dans la preuve des deux propriétés 12.16 et 12.17. On remarque que, pour tout terme $S \in \mathcal{L}_\beta$,

$$\mathrm{Act}(\theta : S) = \mathrm{Act}(\lhd_v(S)) = \mathrm{Act}(S),$$

Ensuite

$$\begin{aligned}
\text{First}(a(v'), \theta : \lhd_v(T)) &\Leftrightarrow \Gamma^{\geq v'}(\theta : \lhd_v(T)) \cap \text{Sol}(t_a = v') \neq \emptyset \\
&\Leftrightarrow \text{Sol}^{\geq v'}(\theta) \cap (\Gamma^{>v}(T))^{\geq v'} \cap \text{Sol}(t_a = v') \neq \emptyset \\
&\Leftrightarrow \text{Sol}^{\geq v'}(\theta) \cap \Gamma^{\geq v'}(T) \cap \text{Sol}(t_a = v') \neq \emptyset \\
&\Leftrightarrow \text{First}(a(v'), \theta : T).
\end{aligned}$$

On va montrer maintenant que la paire (T_1, T_2) satisfait à la condition 12.16. Supposons que T_1 effectue une transition $T_1 \stackrel{a(v')}{\rightarrow} T_1'$ avec $a \neq \delta$. Ceci est possible si et seulement si T effectue une transition $T \stackrel{a(v')}{\rightarrow} T'$, que le prédicat $\text{First}(a(v'), \theta : T)$ soit vrai, et que $v' > v$. Dans ce cas $T_1' = \theta_{v'/t_a} : T'$. Mais, puisque $v' > v$, il existe la transition $\lhd_v(T) \stackrel{a(v')}{\rightarrow} T'$. En plus, nous avons déjà prouvé (12.18) que le prédicat $\text{First}(a(v'), T_2)$ doit être vrai aussi. Par conséquent il existe une transition $T_2 \stackrel{a(v')}{\rightarrow} T_2'$ et $T_1' = T_2' = \theta_{v'/t_a} : T'$. Donc, $(T_1', T_2') \in \approx \subseteq \mathcal{R}$, ce qu'il fallait prouver.

On prouve maintenant que, pour tout v', T_1 exécute $\delta(v')$ si et seulement si T_2 bloque au temps v'. Seulement une des trois situations suivantes peut causer le blocage de T_1 ou de T_2:

1. T exécute $\delta(v')$ et $v' > v$;
2. T exécute $\delta(v')$ et $v' \leq v$;
3. $\delta \notin \text{exec}(T)$, mais $\forall a(v'') \in \text{exec}(T)$ le prédicat $\text{First}(a(v''), \theta : T)$ est faux.

Dans la situation 1, le terme T_2 aussi peut exécuter $\delta(v')$ et transite vers $\downarrow$, ce qu'l fallait montrer. Dans la situation 2, selon la proposition 8.7, T ne peut exécuter aucune autre action après le temps v'. Le sous-terme $\theta : T$ et exécute $\delta(v')$ et $\trianglelefteq_v(T)$ exécute $\delta(v)$. Donc, selon la règle $\mathbf{supp}_3$, T_2 bloque au temps v. D'un autre coté, selon la proposition 8.7, $\theta : T$ ne peut exécuter aucune action $a(v'')$ avec $v'' > v'$. Or, puisque $v' < v$, ceci implique que T_1 exécute $\delta(v)$, ce qu'il fallait prouver.

Il reste à prouver l'implication 12.17. Supposons que T_2 effectue une transition $T_2 \stackrel{a(v'')}{\rightarrow} T_2'$, avec $a \neq \delta$. Ceci n'est possible que si $T \stackrel{a(v'')}{\rightarrow} T'$, que $v'' > v$, et que

le prédicat $\text{First}(a(v''), T_2)$ est vrai. Or, nous avons déjà montré que le prédicat $\text{First}(a(v''), \theta : T)$ est alors vrai aussi. Donc, T_1 peut effectuer la transition $T_1 \stackrel{a(v'')}{\rightarrow} T_1'$. La preuve du fait que $(T_1', T_2') \in \mathcal{R}$ est identique à celle présentée pour l'implication 12.16. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur Seq

Nous considérons seulement l'opérateur $\trianglelefteq$. La preuve pour l'opérateur de décalage stricte $\triangleleft$ est analogue. On va prouver que

$$\trianglelefteq_v(\text{Seq}(T_1, T_2)) \;\approx\; \text{Seq}(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2)), \tag{12.19}$$

pour tous $T_1, T2 \in \mathcal{L}$, et $v \in D$.

Pour ce faire nous définissons les relations $\mathcal{R}_1, \mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$, telles que

$$\begin{aligned} \mathcal{R}_1 &= \{(\trianglelefteq_v(\text{Seq}(T_1, T_2)), \text{Seq}(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2))) \mid v \geq V_0, T_{1,2} \in \mathcal{L}\}, \\ \mathcal{R} &= \{(T, S) \mid (\exists (T', S') \in \mathcal{R}_1)(T \;\approx\; T' \wedge S \;\approx\; S')\} \cup \;\approx . \end{aligned}$$

et prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire nous suivons le raisonnement présenté dans la section 8.2.

Soit $(T, S) \in \mathcal{R}_1$ une paire de termes tels que

$$T = \trianglelefteq_v(\text{Seq}(T_1, T_2)), \tag{12.20}$$

$$S = \text{Seq}(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2)). \tag{12.21}$$

Les égalités

$$\alpha(T) = \alpha(S) \text{ et } \Gamma(T) = \Gamma(S)$$

sont triviales. Prouvons maintenant que $\text{exec}(T) = \text{exec}(S)$. Supposons d'abord que $\text{exec}^{\geq v}(T_1) \neq \emptyset$. Alors,

$$\begin{aligned} \text{exec}(T) &= \text{exec}^{\geq v}(\text{Seq}(T_1, T_2)) \\ &= \text{exec}^{\geq v}(T_1) \\ &= \text{exec}(S). \end{aligned}$$

Autrement, si $\text{exec}^{\geq v}(T_1) = \emptyset$, alors, puisque $\text{exec}(\text{Seq}(T_1, T_2)) = \text{exec}(T_1)$, $\text{Seq}(T_1, T_2)$ non plus ne peut exécuter aucune action au temps v ou plus tard. Ceci implique que T exécute l'action de blocage au temps $\alpha = \alpha(T)$. De plus, on a alors que $\trianglelefteq_v(T_1)$ exécute $\delta(\alpha)$. Selon la sémantique opérationnelle de la composition séquentielle, S aussi bloque au temps α. Donc, $\text{exec}(T) = \text{exec}(S)$. Soit $a(v_a) \in \text{exec}(T)$ une action quelconque et notons par

$$\begin{aligned} T' &= \text{succ}(a(v_a), T), \\ S' &= \text{succ}(a(v_a), S), \\ T_1' &= \text{succ}(a(v_a), T_1'). \end{aligned}$$

Nous allons montrer que $(T', S') \in \mathcal{R}$. Il suffit de considérer juste les actions $a \neq \delta$, la situation de blocage est triviale. Si $T_1' = \uparrow$, alors

$$T' = S' = \lhd_{v_a}(T_2).$$

Or, puisque $\approx$ est une relation d'équivalence (théorème 8.1) elle est réflexive et $(T', S') \in \approx \subseteq \mathcal{R}_1$. Si, par contre, $T_1 \neq \surd$ alors

$$\begin{aligned} T' &= \text{Seq}(T_1', T_2) \\ S' &= \text{Seq}(T_1', \trianglelefteq_{v'}(T_2)). \end{aligned}$$

Or, $v_a = \alpha(T_1') = \alpha(T')$. Donc, selon la cohérence de l'axiome **Dec2**, prouvée dans la section 12.4.2.1, on a que

$$\begin{aligned} T' &\approx \trianglelefteq_{v_a}(\text{Seq}(T_1', T_2)), \\ T_1' &\approx \trianglelefteq_{v_a}(T_1'). \end{aligned}$$

En plus, puisque $v_a \geq v \geq v'$ et étant donné que $\approx$ est une congruence par rapport à l'opérateur de composition séquentielle, on déduit que $(T', S') \in \mathcal{R}$. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur $\sum$

Nous considérons juste l'opérateur $\trianglelefteq$, la preuve pour $\triangleleft$ est analogue. Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(+(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2)), \trianglelefteq_v(+(T_1, T_2))) \mid v \geq V_0, T_{1,2} \in \mathcal{L}\} \cup \quad \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Soit $(T, S) \in \mathcal{R}$ une paire de termes telle que:

$$T = +(\trianglelefteq_v(T_1), \trianglelefteq_v(T_2)), \tag{12.22}$$

$$S = \trianglelefteq_v(+(T_1, T_2))), \tag{12.23}$$

avec $T_{1,2} \in \mathcal{L}$ et $v \geq V_0$.

Les égalités $\alpha(T) = \alpha(S)$ et $\Gamma(T) = \Gamma(S)$ sont triviales.

Prouvons maintenant que la paire (T, S) a les deux propriétés d'une bisimulation forte énoncées dans la définition 4.2.

Montrons d'abord que $\text{exec}(T) = \text{exec}(S)$. T exécute une action quelconque $a(v')$, avec $a \neq \delta$ si et seulement si au moins un des sous-termes $\trianglelefteq_v(T_i)$ l'exécute, donc, si et seulement si un de sous-termes T_i l'exécute et que $v' \geq v$. Or, cette condition est nécessaire et suffisante pour que S puisse aussi exécuter $a(v')$.

Considérons maintenant un blocage $\delta(v')$. Nous prouverons seulement que si T exécute $\delta(v')$ alors S aussi exécute $\delta(v')$; l'implication inverse est analogue.

T exécute l'action de blocage $\delta(v')$ si et seulement si une des deux conditions suivantes est satisfaite:

- (1) les deux sous-termes $\trianglelefteq_v(T_{1,2})$ exécutent $\delta(v')$,
- (2) un des sous-termes $\trianglelefteq_v(T_i)$ bloque au temps v' et le prédicat $\text{Wait}(v', \trianglelefteq_v(T_j))$ est faux, où $\{i, j\} = \{1, 2\}$.

Lorsqu'un terme $\trianglelefteq_v(T_i)$ exécute une action de blocage $\delta(v'')$, où $v'' \geq V_0$ est un instant temporel quelconque, ceci peut être dû à deux situations:

- (i) T_i exécute $\delta(v'')$ et $v'' \geq v$, ou bien
- (ii) T_i ne peut exécuter aucune action au temps v ou plus tard, auquel cas $v'' = v$.

Supposons que T bloque le scénario (1) et que la situation (i) s'applique aux deux sous-termes $T_{1,2}$. On déduit alors que $+(T_1, T_2)$ aussi exécute $\delta(v')$. Or, comme $v' \geq v$, S aussi bloque au temps v'.

Considérons ensuite une situation où T bloque selon le scénario (1), T_i selon (i), et T_j selon (ii). Puisque T_j ne peut exécuter aucune action au temps v ou plus tard, le prédicat $\text{Wait}(v, T_j)$ est faux. Or, T_i exécute $\delta(v)$. Donc, les termes $+(T_1, T_2)$ et, par conséquent, S aussi exécutent $\delta(v)$.

Finalement, supposons qu'on est dans la situation (1) et que les deux sous-termes $\trianglelefteq_v(T_{1,2})$ bloquent selon (ii). Dans ce cas on a que $v' = v$. Puisque ni T_1, ni T_2 ne peuvent exécuter aucune action au temps v ou plus tard, le terme $+(T_1, T_2)$ non plus ne peut exécuter aucune action $a(v')$ avec $v' \geq v$. Donc, S exécute l'action $\delta(v)$.

Supposons maintenant que T exécute $\delta(v')$ parce que $\trianglelefteq_v(T_i)$ bloque au temps v' et que le prédicat $\text{Wait}(v', \trianglelefteq_v(T_j))$ et faux (scénario (2)).

Selon la proposition 8.6, puisque $\text{Wait}(v'', \trianglelefteq_v(T_j))$ est vrai pour tout $v'' \leq v$. Il faut, donc, que $v' > v$. En plus, selon la même proposition, on déduit que $\text{Wait}(v', T_j)$ est faux aussi. Donc, $\trianglelefteq_v(T_i)$ ne peut bloquer que selon le scénario (i), c'est à dire, parce que T_i exécute $\delta(v')$. Or, dans ces conditions, le terme $+(T_1, T_2)$ bloque aussi au temps v'. Comme $v' > v$ ceci implique que S exécute $\delta(v')$. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur [ ]

Nous considérons juste l'opérateur $\trianglelefteq$, car la preuve pour $\triangleleft$ est analogue. Nous allons prouver que pour tout $v \geq V_0$,

$$\trianglelefteq_v([\mathcal{C}, \epsilon](T_1, \ldots, T_m)) \approx [\mathcal{C}, \epsilon](\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_m)).$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ les relations suivantes:

$$\begin{aligned}\mathcal{R}_1 &= \{(T,S) \mid T = \trianglelefteq_v([\mathcal{C},\epsilon](T_1,\ldots,T_m)), S = [\mathcal{C},\epsilon](\trianglelefteq_v(T_1),\ldots,\trianglelefteq_v(T_m)), v \geq V_0\} \\ \mathcal{R} &= \{(T,S) \mid (\exists (T',S') \in \mathcal{R}_1)(T \approx T' \ \wedge \ S \approx S')\} \cup \ \approx .\end{aligned}$$

Montrons que $\mathcal{R}$ est une bisimulation–ACTC, en suivant le raisonnement présenté dans la section 8.2. Soit (T,S) une paire quelconque de $\mathcal{R}$, telle que

$$\begin{aligned}T &= \trianglelefteq_v([\mathcal{C},\epsilon](T_1,\ldots,T_m)), \\ S &= [\mathcal{C},\epsilon](\trianglelefteq_v(T_1),\ldots,\trianglelefteq_v(T_m)).\end{aligned}$$

Les égalités

$$\alpha(T) = \alpha(S) \text{ et } \Gamma(T) = \Gamma(S)$$

sont triviales. Prouvons maintenant que pour toute transition

$$T \stackrel{a(v)}{\rightarrow} T'$$

il existe une transition

$$S \stackrel{a(v)}{\rightarrow} S'$$

telle que $(T',S') \in \mathcal{R}$.

Si $a = \delta$, alors le blocage peut être dû au fait que le sous-terme $[\mathcal{C},\epsilon](T_1,\ldots,T_m)$ exécute $\delta(v')$ ou bien au fait que $[\mathcal{C},\epsilon](T_1,\ldots,T_m)$ ne peut exécuter aucune action au temps v ou plus tard, auquel cas $v' = v = \alpha(T)$.

On remarque d'abord que, selon le théorème 8.6,

$$\text{Wait}(v,[\mathcal{C},\epsilon](T_1,\ldots,T_m)) \Leftrightarrow \bigwedge_{i=1}^{m} \text{Wait}(v,T_i).$$

Donc, $\text{exec}^{\geq v}([\mathcal{C},\epsilon](T_1,\ldots,T_m)) = \emptyset$ si et seulement s'il existe au moins un sous-terme T_i tel que $\text{exec}^{\geq v}(T_i) = \emptyset$.

Supposons que le blocage T est du au blocage de $[\mathcal{C},\epsilon](T_1,\ldots,T_m)$. On distingue alors les situations suivantes:

1. il y a un sous-terme T_i qui exécute $\delta(v')$ et le prédicat $\text{Wait}(v', T_j)$ est vrai pour tous les autres sous-termes (règle $\mathbf{Com}_2$), ou bien

2. principe de causalité: un sous-terme T_i effectue une transition $T_i \stackrel{oc(v')}{\rightarrow} T_i'$, où $oc = \epsilon(c, T_i)$ est l'action interne correspondante à l'action de communication $c \in \mathcal{C}$, et la communication ne peut pas être établie parce qu'un moins un sous-terme T_j qui communique avev T_i via l'action c ne peut exécuter son action interne correspondante $ic(v)$ au temps v; de plus, tous les sous-termes T_k, avec $k \neq j$ peuvent attendre jusqu'au temps v';

3. cette situation est similaire à la précédente et ne diffère de celle-ci que par le fait que l'action communicante $oc(v)$ est remplacée par une action $ic(v)$, dont l'exécution est urgente au temps v.

Dans la situation 1, puisque $v' \geq v$, alors il existe la transition $\trianglelefteq_v(T_i) \stackrel{\delta(v')}{\rightarrow} \downarrow$. En plus, selon la proposition 8.6, les prédicats $\text{Wait}(v', \trianglelefteq_v(T_j)$ sont vrais aussi, pour tout $j \neq i$. Donc, S exécute $\delta(v')$ et transite vers $\downarrow$, ce qu'il fallait montrer.

Similairement, dans la situation 2 on déduit que $\trianglelefteq_v(T_i)$ exécute $oc(v')$ et les prédicats $\text{Wait}(v', \trianglelefteq_v(T_j)$ sont vrais. Donc, on peut appliquer la règle $\mathbf{Com}_3$ au terme S et déduire qu'il exécute $\delta(v)$, ce qu'il fallait prouver.

Finalement, la situation 3 implique que le prédicat $\text{Urgent}(\trianglelefteq_v(T_i), ic(v''))$ est vrai et qu'il existe au moins un sous-terme $\trianglelefteq_v(T_j)$ qui communique avec T_i et qui ne peut pas exécuter son action interne correspondante à ic. Donc, en appliquant la règle $\mathbf{Com}_4$ au terme S, on déduit que S bloque au temps v. Ceci conclut les situations de blocage du terme T.

Supposons maintenant que T exécute une action $a(v')$, avec $a \neq \delta$ et $v' \geq v$. C'est trivial de montrer qu'il existe une transition $S \stackrel{a(v'')}{\rightarrow} S'$: on applique directement les règles de sémantique opérationnelle des opérateurs $\trianglelefteq$ et $[]$, ainsi que la proposition 8.6. Il reste à prouver que $(T', S') \in \mathcal{R}$. On remarque d'abord que les ensembles d'indices $I(a)$ et $J(a)$ sont les mêmes lorsque définis pour T ou S. Donc, on distingue

les situations suivantes:

1. $|I(a) \cup J(a)| \geq 2$. Dans ce cas

$$\begin{aligned} T' &= [\mathcal{C}', \epsilon']_{i \in I(a), j \in J(a)}(T'_i, \trianglelefteq_{v'}(T_j)), \\ S' &= [\mathcal{C}', \epsilon']_{i \in I(a), j \in J(a)}(T'_i, \trianglelefteq_{v'}(\trianglelefteq_{v}(T_j))), \end{aligned}$$

   où $\mathcal{C}'$ et ϵ' sont définis dans le tableau 6.6.

   Or, selon la cohérence de l'axiome $\mathbf{Dec}_2$ (section 12.4.2.1) et puisque $\approx$ est une congruence par rapport à l'opérateur [], ceci implique que $T' \approx S'$, donc que $(T', S') \in \approx \subseteq \mathcal{R}$;

2. $|I(a)| = |J(a)| = 0$. Dans ce cas alors $T' = S' = \uparrow$ et puisque $\approx$ est réflexive (théorème 8.1), on a que $(T', S') \in \approx \subseteq \mathcal{R}$;

3. $|I(a)| = 0, |J(a)| = 1$, alors $T' = \trianglelefteq_{v'}(T_j)$ et $S' = \trianglelefteq_{v'}(\trianglelefteq_{v}(T_j))$. On applique encore une fois la cohérence de l'axiome $\mathbf{Dec}_2$ et déduit que $(T', S') \in \approx \subseteq \mathcal{R}$;

4. $|I(a)| = 1, |J(a)| = 0$, alors $T' = S' = T'_i$. Donc, $(T', S') \in \approx \subseteq \mathcal{R}$.

La preuve de la propriété symétrique, c'est à dire que pour toute transition $S \stackrel{a(v')}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v'}{\rightarrow} T'$ avec $(T', S') \in \mathcal{R}$, est analogue. Nous pouvons, donc, conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur $\oplus$

Nous considérons juste l'opérateur $\trianglelefteq$, parce que la preuve pour $\triangleleft$ est analogue. Soient les relations $\mathcal{R}, \mathcal{R}_1 \subseteq \mathcal{L} \times \mathcal{L}$, définies comme suit:

$$\begin{aligned} \mathcal{R}_1 &= \{(\trianglelefteq_{v}(\oplus_{i \in I}(T_i), \oplus_{i \in I}(\trianglelefteq_{v}(T_i))), \mid V_0 \leq v, T_i \in \mathcal{L}\} \\ \mathcal{R} &= \{(T, S) \mid (\exists (T_1, S_1) \in \mathcal{R}_1)(T_1 \approx T \wedge S_1 \approx S)\} \cup \approx . \end{aligned}$$

En suivant le raisonnement présenté dans la section 8.2 on va montrer que $\mathcal{R}$ est une bisimulation–ACTC.

Soit $(T, S) \in \mathcal{R}_1$ une paire de termes tels que

$$\begin{aligned} T &= \trianglelefteq_v(\oplus_{i\in I}(T_i) \\ S &= \oplus_{i\in I}(\trianglelefteq_v(T_i)), \end{aligned}$$

où $I = \{1, \ldots, m\}$ est un ensemble quelconque d'indices avec $m \geq 2$ et $v \geq 0$.

Il suit directement des définitions des fonctions α et Γ

$$\alpha(S) = \alpha(T) = v \text{ et } \Gamma(T) = \Gamma(S).$$

La preuve du fait que que pour toute transition $T \stackrel{a(v')}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v')}{\rightarrow} S'$ telle que $(T', S') \in \mathcal{R}$ est triviale: il suffit de considérer une par une toutes les règles de sémantique opérationnelle des opérateurs $\oplus$ et $\trianglelefteq$. La preuve de la propriété inverse est analogue. Donc, (S, T) satisfait aux conditions énoncées dans la définition d'une bisimulation–ACTC et on peut conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur Loop

Nous considérons juste l'opérateur $\trianglelefteq$, la preuve pour $\triangleleft$ est analogue. Définissons d'abord les relations $\mathcal{R}, \mathcal{R}_{1,2,3} \subseteq \mathcal{L} \times \mathcal{L}$ suivantes:

$$\begin{aligned} \mathcal{R}_1 &= \{(\trianglelefteq_v(\text{Loop}(T_1)), \text{Loop}(\trianglelefteq_v(T_1))) \mid v \geq 0, T_1 \in \mathcal{L}\} \\ \mathcal{R}_2 &= \{(\text{Seq}(T_2, \text{Loop}(T_1)), \text{Seq}(T_2, \text{Loop}(\trianglelefteq_v(T_1)))) \mid 0 \leq v \leq \alpha(T_2), T_{1,2} \in \mathcal{L}\} \\ \mathcal{R}_3 &= \{(\trianglelefteq_{v'}(\text{Loop}(T_1)), \trianglelefteq_{v'}(\text{Loop}(\trianglelefteq_v(T_1)))) \mid 0 \leq v \leq v', T_1 \in \mathcal{L}\} \\ \mathcal{R} &= \{(S, T) \mid (\exists (S', T') \in \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)(S' \approx S \wedge T' \approx T)\} \cup \approx . \end{aligned}$$

Nous prouvons ici que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire il suffit de considérer d'abord une paire (T, S) de $\mathcal{R}_1$, ensuite de $\mathcal{R}_2$, et finalement de $\mathcal{R}_3$ et de prouver que (S, T) a les trois propriétés énoncées dans la définition 8.1 (voir raisonnement présenté dans la section 8.2.

Pour toute paire $(S, T) \in \mathcal{R}$ les égalités

$$\alpha(T) = \alpha(S) \text{ et } \Gamma(T) = \Gamma(S)$$

sont triviales.

Supposons d'abord que $(S,T) \in \mathcal{R}_1$:

$$\begin{aligned} T &= \trianglelefteq_v(\text{Loop}(T_1)) \\ S &= \text{Loop}(\trianglelefteq_v(T_1)). \end{aligned}$$

Il suit directement de la définition de la fonction Γ (tableau 6.2) que $\Gamma(S) = \Gamma(T)$. Par définition, $\alpha(T) = \alpha(S) = v$.

Montrons maintenant que pour toute transition $T \stackrel{a(v')}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v')}{\rightarrow} S'$ avec $(T',S') \in \mathcal{R}$. La preuve de la propriété symétrique est analogue.

On remarque d'abord que

$$\begin{aligned} \text{exec}(S) &= \text{exec}(\trianglelefteq_v(T_1)) \\ \text{exec}(\text{Loop}(T_1)) &= \text{exec}(T). \end{aligned}$$

Prouvons que pour toute action $a(v) \in \text{exec}(T) = \text{exec}(S)$ et pour tout $T' \in \text{succ}(a(v),T)$ il existe $S' \in \text{succ}(a(v),S)$ tel que $(T',S') \in \mathcal{R}$. Pour chaque règle $\mathbf{B}_1$ – $\mathbf{B}_5$ l'hypothèse $\text{Hyp}_{\mathbf{B}_1}(T)$ est vraie si et seulement si $\text{Hyp}_{\mathbf{B}_1}(S)$ est vraie. Donc, les successeurs T' et S' peuvent être tels que

$$\begin{aligned} \text{selon } \mathbf{B}_1: \quad T' &= \text{Seq}(T_1', \text{Loop}(T_1)), \\ S' &= \text{Seq}(T_1', \text{Loop}(\trianglelefteq_v(T_1)) \\ \text{selon } \mathbf{B}_2: \quad T' &= S' = T_1', \\ \text{selon } \mathbf{B}_3: \quad T' &= \triangleleft_{v'}(\text{Loop}(T_1)) \\ S' &= \triangleleft_{v'}(\text{Loop}(\trianglelefteq_v(T_1))), \\ \text{selon } \mathbf{B}_4: \quad T' &= S' \uparrow, \\ \text{selon } \mathbf{B}_5: \quad T' &= = S' \downarrow . \end{aligned}$$

Les cas décrits par les règles $\mathbf{B}_2$, $\mathbf{B}_4$ et $\mathbf{B}_5$ sont triviaux puisque $(T',S') \in \approx \subseteq \mathcal{R}$. Si T et S effectuent une transition selon la règle $\mathbf{B}_1$. On a que $v \leq v' = \alpha(T_1')$.

Donc, $(T', S') \in \mathcal{R}_2 \subseteq \mathcal{R}$. Finalement, selon la règle $\mathbf{B}_3$, puisque $v \leq v'$ on a que $(T', S') \in \mathcal{R}_3 \subseteq \mathcal{R}$.

Considérons maintenant une paire $(S, T) \in \mathcal{R}_2$:

$$\begin{aligned} T &= \text{Seq}(T_2, \text{Loop}(T_1)) \\ T &= \text{Seq}(T_2, \text{Loop}(\trianglelefteq_v(T_1)), \end{aligned}$$

avec $v \leq \alpha(T_2)$.

On remarque que $\text{exec}(S) = \text{exec}(T) = \text{exec}(T_2)$. Il reste à prouver que les successeurs T' et S' de T, respectivement S lors de l'exécution d'une même action $a(v')$ forment une paire de $\mathcal{R}$. Si $a = \delta$, la preuve est triviale et suit directement du fait que $\approx$ est réflexive (théorème 8.1).

Si $a \neq \delta$, alors notons par T_2' le successeur de T_2 après l'exécution de $a(v')$. Si $T_2' \neq \uparrow$, alors

$$\begin{aligned} T' &= \text{Seq}(T_2', \text{Loop}(T_1)) \\ S' &= \text{Seq}(T_2', \text{Loop}(\trianglelefteq_v(T_1))) \end{aligned}$$

et $(T', S') \in \mathcal{R}_2 \subseteq \mathcal{R}$. Autrement, si $T_2' = \uparrow$, alors

$$\begin{aligned} T' &= \trianglelefteq_{v'}(\text{Loop}(T_1)) \\ S' &= \trianglelefteq_{v'}(\text{Loop}(\trianglelefteq_v(T_1))) \end{aligned}$$

et $(T', S') \in \mathcal{R}_3 \subseteq \mathcal{R}$.

Finalement, soit $(S, T) \in \mathcal{R}_3$,

$$\begin{aligned} T &= \trianglelefteq_{v'}(\text{Loop}(T_1)) \\ S &= \trianglelefteq_{v'}(\text{Loop}(\trianglelefteq_v(T_1))), \end{aligned}$$

avec $v' \geq v$ et $T_1 \in \mathcal{L}$. Dans ce cas c'est évident que $\text{exec}(T) = \text{exec}(S)$. Il reste à prouver que les successeurs respectifs de S et T lors de l'exécution d'une même action $a(v'')$, notés S' et T' respectivement, forment une paire $(S', T') \in \mathcal{R}$.

Le cas $a = \delta$ est trivial. Si $a \neq \delta$, alors il suffit de remarquer que T transite vers le successeur de $\text{Loop}(T_1)$ et S vers celui de $\text{Loop}(\trianglelefteq_v(T_1)))$. De plus, pour toute action $a \neq \delta$ et tout $v' \geq v$

$$\text{succ}(a(v'), T) = \text{succ}(a(v'), \text{Loop}(T_1)) = \text{succ}(a(v'), \trianglelefteq_v(\text{Loop}(T_1)).$$

Or, $(\text{Loop}(T_1), \text{Loop}(\trianglelefteq_v(T_1))) \in \mathcal{R}_1$ et nous avons déjà montré que les paires de $\mathcal{R}_1$ sont fermés dans $\mathcal{R}$ aux transitions.

Nous pouvons, donc, conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

Distributivité de $\trianglelefteq$ par rapport à l'opérateur Ex

Nous considérons juste l'opérateur $\trianglelefteq$, puisque la preuve pour $\triangleleft$ est analogue. Soient $\mathcal{R}_1, \mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ les relations suivantes:

$$\begin{aligned}
\mathcal{R}_1 &= \{(\trianglelefteq_v(\text{Ex}(T_n, T_c, T_e), \text{Ex}(\trianglelefteq_v(T_n), \trianglelefteq_v(T_c), \trianglelefteq_{v_0}(T_e)) \mid \\
&\quad V_0 \leq v_0 \leq \alpha(T_n) \leq v, T_n, T_c, T_e \in \mathcal{L}\} \\
\mathcal{R} &= \{(T, S) \mid (\exists (T_1, S_1) \in \mathcal{R}_1)(T \approx T_1 \wedge S \approx S_1)\} \cup \approx .
\end{aligned}$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation–ACTC, en suivant le raisonnement présenté dans la section 8.2.

Soit (T, S) une paire quelconque de $\mathcal{R}_1$, où

$$\begin{aligned}
T &= \trianglelefteq_v(\text{Ex}(T_n, T_c, T_e)) \\
S &= \text{Ex}(\trianglelefteq_v(T_n), \trianglelefteq_v(T_c), \trianglelefteq_{v_0}(T_e)).
\end{aligned}$$

Par définition,

$$\alpha(S) = \alpha(\trianglelefteq_v(T_n)) = \alpha(\trianglelefteq_v(T_c)) = v.$$

Donc, $\alpha(T) = \alpha(S)$. L'egalité

$$\Gamma(T) = \Gamma(S)$$

est triviale.

Il reste à montrer que $\mathcal{R}$ est une bisimulation forte. Supposons que T effectue une transition quelconque $T \stackrel{a(v')}{\rightarrow} T'$. Nous devons montrer qu'il existe alors une transition $S \stackrel{a(v')}{\rightarrow} S'$, telle que $(T', S') \in \mathcal{R}$. La preuve de la propriété symétrique, c'est à dire que pour toute transition $S \stackrel{a(v')}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v')}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$ est analogue et ne sera pas présentée en détail.

Supposons d'abord que $\text{exec}(T) = \text{exec}^{\geq v}(\text{Ex}(T_n, T_c, T_e)) \neq \emptyset$. Si T exécute une action $a(v')$, avec $a \neq \delta$ et $v' \geq v$, alors il y a une des cinq règles de sémantique opérationnelle $\mathbf{Ex}_{1-5}$ qui s'applique.

Selon $\mathbf{Ex}_1$, $a = ic$ et il existe les transitions suivantes:

$$\begin{aligned} T_n &\stackrel{a(v')}{\rightarrow} T'_n \\ T_c &\stackrel{a(v')}{\rightarrow} T'_c \\ T &\stackrel{a(v')}{\rightarrow} T' = \text{Ex}(T'_n, T'_c, T_e). \end{aligned}$$

Puisque $v' \geq v$ il existe aussi les transitions:

$$\begin{aligned} \trianglelefteq_v(T_n) &\stackrel{a(v')}{\rightarrow} T'_n \\ \trianglelefteq_v(T_c) &\stackrel{a(v')}{\rightarrow} T'_c \\ S &\stackrel{a(v')}{\rightarrow} S' = \text{Ex}(T'_n, T'_c, \trianglelefteq_{v_0}(T_e)). \end{aligned}$$

Par la cohérence de l'axiome **Dec2** (section 12.4.2.2) on a que

$$\begin{aligned} T' &\approx \trianglelefteq_{v'}(\text{Ex}(T'_n, T'_c, T_e)), \\ T'_n &\approx \trianglelefteq_{v'}(T'_n), \\ T'_c &\approx \trianglelefteq_{v'}(T'_c). \end{aligned}$$

Or, $\approx$ est une congruence par rapport à l'opérateur Ex (théorème 8.10). Donc,

$$S' \approx \text{Ex}(\trianglelefteq_{v'}(T'_n), \trianglelefteq_{v'}(T'_c), \trianglelefteq_{v_0}(T_e)).$$

On peut alors déduire que $(T', S') \in \mathcal{R}$, ce qu'il fallait montrer.

Considérons maintenant la règle $\mathbf{Ex}_2$. Dans ce cas le prédicat $\text{Wait}(v', T_c)$ doit être vrai et il existe les transitions suivantes:

$$\begin{aligned} T_n &\overset{a(v')}{\rightarrow} T'_n, \\ T &\overset{a(v')}{\rightarrow} T' = \text{Ex}(T'_n, \trianglelefteq_{v'}(T_c), T_e). \end{aligned}$$

Selon la proposition 8.6 on a alors que le prédicat $\text{Wait}(v', \trianglelefteq_v(T_c))$ est vrai aussi. De plus, il existe les transitions:

$$\begin{aligned} \trianglelefteq_v(T_n) &\overset{a(v')}{\rightarrow} T'_n, \\ S &\overset{a(v')}{\rightarrow} S' = \text{Ex}(T'_n, \trianglelefteq_{v'}(\trianglelefteq_v(T_c)), \trianglelefteq_{v_0}(T_e)). \end{aligned}$$

On remarque que selon la cohérence de l'axiome **Dec1** (section 12.4.2.1)

$$\trianglelefteq_{v'}(\trianglelefteq_v(T_c)) \approx \trianglelefteq_{v'}(T_c).$$

On applique ensuite un raisonnement similaire au cas précédent et on déduit que $(T', S') \in \mathcal{R}$.

La règle de sémantique opérationnelle $\mathbf{Ex}_3$ est analogue à la règle $\mathbf{Ex}_2$. Considérons la règle $\mathbf{Ex}_4$. L'hypothèse de cette règle exige que le prédicat $\text{Wait}(v', T_n)$ soit vrai et que la transition

$$T_c \overset{a(v')}{\rightarrow} \uparrow$$

existe. Puisque $v' \geq v$ et selon la proposition 8.6 on a que le prédicat $\text{Wait}(v', \trianglelefteq_v(T_n)$ est vrai aussi et que la transition

$$\trianglelefteq_v(T_c) \overset{a(v')}{\rightarrow} \uparrow$$

a lieu. Ceci implique que

$$\begin{aligned} T &\overset{a(v')}{\rightarrow} T' = \trianglelefteq_v(T_e), \\ S &\overset{a(v')}{\rightarrow} S' = \trianglelefteq_v(\trianglelefteq_{v_0}(T_e)). \end{aligned}$$

Or, selon la cohérence de l'axiome **Dec1** (section 12.4.2.1),

$$(T', S') \in \approx \subseteq \mathcal{R}.$$

Les règles $\mathbf{Ex}_{5,6,7}$ sont similaires à la règle $\mathbf{Ex}_4$: on déduit que

$$(T', S') = (\uparrow, \uparrow) \in \approx \subseteq .$$

Supposons maintenant que $\text{exec}^{\geq v}(\text{Ex}(T_n, T_c, T_e)) = \emptyset$, ce qui implique que T bloque au temps $v = \alpha(T)$. Il suffit alors de remarquer que $\text{Ex}(T_n, T_c, T_e)$ peut exécuter une action quelconque $a(v')$ si et seulement si soit T_n et T_c peuvent exécuter $a(v')$ ou bien si un des sous-termes T_n ou T_c peut exécuter $a(v')$ et l'autre peut attendre jusqu'au temps v'. Ceci implique que

$$\text{exec}^{\geq v}(\text{Ex}(T_n, T_c, T_e)) = \emptyset \Leftrightarrow$$
$$\text{exec}^{\geq v}(\text{Ex}(T_n, T_c, T_e)) = \emptyset \vee \text{exec}^{\geq v}(\text{Ex}(T_n, T_c, T_e)) = \emptyset.$$

Par conséquent au moins un des sous-termes $\trianglelefteq_v(T_n)$ ou $\trianglelefteq_v(T_c)$ bloque au temps v et ou bien l'autre sous-terme peut attendre jusqu'au temps v, ou bien il bloque aussi. Donc, S et T exécutent $\delta(v)$ et transitent vers $\downarrow$. Or, $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$, ce qu'il fallait montrer.

La preuve de la propriété symétrique, c'est à dire que pour toute transition $S \stackrel{a(v')}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v')}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$ est analogue. Nous pouvons, donc, résumer la preuve de la distributivité de $\trianglelefteq$ par rapport à Ex. ∎

12.4.2.4 Cohérence de Pref1

Soit $T = a(t_a).T_1 \in \mathcal{L}_\beta$, avec $a \in \mathbf{A}$ et $T_1 \in \mathcal{L}_\beta \setminus \{\surd\}$. Nous prouvons que

$$T \approx \bigwedge_{b \in \text{Act}(T_1)} t_a < t_b : (a(t_a) \| T_1).$$

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(a(t_a).T_1, \bigwedge_{b \in \text{Act}(T_1)} t_a < t_b : \|(a(t_a), T_1)) \mid a \in A \setminus \{\delta\}, T_1 \in \mathcal{L}_\beta \setminus \{\surd\}\} \cup \approx .$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation–ACTC. Soit $(T, S) \in \mathcal{R}$ telle que

$$T = a(t_a).T_1$$
$$S = \bigwedge_{b \in \text{Act}(T_1)} t_a < t_b : \|(a(t_a), T_1)).$$

Les égalités

$$\alpha(T) = \alpha(S) = \alpha(T_1).$$

et

$$\Gamma(T) = \Gamma(S).$$

suivent directement des définitions. Il reste, donc, à prouver que $\mathcal{R}$ est une bisimulation forte. Les seules transitions de $a(t_a).T_1$ sont

$$a(t_a).T_1 \stackrel{a(v)}{\rightarrow} \lhd_v(T_1),$$

pour tout $v \geq V_0$. Similairement, les seules transitions que S peut effectuer sont

$$S \stackrel{a(v)}{\rightarrow} \bigwedge_{b \in \mathrm{Act}(T_1)} v < t_b : \trianglelefteq_v(T_1)$$

avec et $v \geq V_0$. Or, puisque $v = \alpha(\trianglelefteq_v(T_1))$,

$$\bigwedge_{b \in \mathrm{Act}(T_1)} v < t_b : \trianglelefteq_v(T_1) \stackrel{\text{Consist.cons4}}{\approx} \trianglelefteq_v(T_1).$$

Remarquons que la preuve de la cohérence de l'axiome **cons4** (section 12.4.2.9) ne dépend pas du présent résultat. Par conséquent, les successeurs respectifs de T et S sont équivalents et forment une paire de $\mathcal{R}$. ∎

12.4.2.5 Cohérence de Par1

Nous devons prouver que la relation $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ définie comme:

$$\mathcal{R} = \{(\|(T_1, T_2), \|(T_2, T_1)) \mid T_{1,2} \in \mathcal{L}_\beta\} \cup \approx$$

est une bisimulation-ACTC. Soit $(S, T) \in \mathcal{R}$ une paire quelconque telle que

$$\begin{aligned} S &= \|(T_1, T_2) \\ T &= \|(T_2, T_1), \end{aligned}$$

où $T_{1,2} \in \mathcal{L}_\beta$. Il suit directement des définitions que $\alpha(T) = \alpha(S)$ et que $\Gamma(S) = \Gamma(T)$. On remarque aussi que $\mathrm{exec}(S) = \mathrm{exec}(T)$, puisque les règles de sémantique

opérationnelle de la composition parallèle sont symétriques. Il reste à montrer que pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ la transition $S \stackrel{a(v)}{\rightarrow} S'$ est telle que $(S', T') \in \mathcal{R}$. Si $a = \delta$ la preuve est triviale et utilise le fait que $\approx$ est réflexive (théorème 8.1).

Si $a \neq \delta$ alors il doit exister une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et le prédicat Wait(v, T_j) doit être vrai, où $\{i, j\} = \{1, 2\}$. Si $T_i' = \uparrow$, alors $S' = T' = \trianglelefteq_v(T_j)$ et, par conséquent, $(S', T') \in \approx \subseteq \mathcal{R}$. Sinon, supposons sans perte de généralité, que $i = 1$ et $j = 2$. Alors,

$$S' = \|(T_1', \trianglelefteq_v(T_2)) \text{ et}$$
$$T' = \|(\trianglelefteq_v(T_2), T_1').$$

Donc, par définition, $(S', T') \in \mathcal{R}$. ∎

12.4.2.6 Cohérence de Par2

Soient $\mathcal{R}, \mathcal{R}_m \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$, avec $m \geq 3$, les relations définies comme suit:

$$\begin{aligned} \mathcal{R}_m &= \{(\|(T_1, \ldots, T_m), \|(T_1, \|(T_2, \ldots, T_m)) \mid T_i \in \mathcal{L}\} \\ \mathcal{R} &= \{(S, T) \mid (\exists m \geq 3)(\exists (S', T') \in \mathcal{R}_m)(S \approx S' \wedge T \approx T')\} \cup \approx . \end{aligned}$$

Nous allons prouver que la relation $\mathcal{R}$ est une bisimulation–ACTC.

Selon le raisonnement présenté dans la section 8.2 il suffit de considérer une paire quelconque de $(S, T) \in \mathcal{R}_m$, pour un m quelconque, et prouver que (S, T) a les propriétés énoncées dans la définition d'une bisimulation–ACTC(definition 8.1)

Il est trivial de remarquer que, pour tout $m \geq 3$ et pour toute paire $(S, T) \in \mathcal{R}_m$, $\alpha(S) = \alpha(T)$ et que $\Gamma(S) = \Gamma(T)$.

Montrons maintenant que exec(S) = exec(T), pour tout $m \geq 3$ et toute paire $(S, T) \in \mathcal{R}_m$, telle que

$$\begin{aligned} S &= \|(T_1, \ldots, T_m) \\ T &= \|(T_1, \|(T_2, \ldots, T_m)). \end{aligned}$$

Considérons d'abord les situations de blocage: S exécute une action $\delta(v)$ si et seulement si au moins un des sous-termes T_i bloque au temps v et tous les autres sous-termes peuvent attendre jusqu'au temps v. Si $i = 1$, alors selon la proposition 8.6, le prédicat $\text{Wait}(v, \|(T_2, \ldots, T_m))$ et vrai, donc S aussi peut exécuter $\delta(v)$. Sinon, c'est à dire si $\delta(v) \notin \text{exec}(T_1)$, alors d'une part le prédicat $\text{Wait}(v, T_1)$ doit être vrai. D'une autre part au moins un des sous-termes T_j bloque au temps v et pour tous les autres sous-termes T_k, avec $2 \leq j, k \leq m$, ou bien le prédicat $\text{Wait}(v, T_k)$ doit être vrai, ou bien il existe une transition $T_k \stackrel{\delta(v)}{\rightarrow} \downarrow$. Par conséquent, $\|(T_2, \ldots, T_m)$ et T aussi exécutent l'action de blocage $\delta(v)$.

Inversement, supposons que T bloque au temps v. Ceci est possible si et seulement si soit (1) T_1 exécute $\delta(v)$ et le prédicat $\text{Wait}(v, \|(T_2, \ldots, T_m))$ est vrai, ou bien si $\|(T_2, \ldots, T_m)$ bloque au temps v et le prédicat $\text{Wait}(v, T_1)$ est vrai. Dans les deux cas on déduit que T aussi exécute $\delta(v)$.

On suit un raisonnement similaire pour prouver que pour toute action $a \neq \delta$ et tout $v \geq V_0$, $a(v) \in \text{exec}(T) \Leftrightarrow a(v) \in \text{exec}(S)$.

Il reste à prouver la propriété de fermeture, c'est à dire que les successeurs de T et S lors de l'exécution d'une même action, notés T' et S' respectivement, forment une paire de $\mathcal{R}$.

Soit $m \geq 3$ et $(T, S) \in \mathcal{R}_m$, tel que

$$\begin{aligned} T &= \|(T_1, \ldots, T_m) \\ S &= \|(T_1, \|(T_2, \ldots, T_m)). \end{aligned}$$

Si $T \stackrel{a(v)}{\rightarrow} T'$ et $S \stackrel{a(v)}{\rightarrow} S'$, avec $a \neq \delta$, alors il existe un sous-terme T_i qui effectue la transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et le prédicat $\text{Wait}(v, T_j)$ est vrai pour tout $j \neq i$.

Considérons premièrement le cas où $i = 1$. Si $T_i' \neq \uparrow$, alors

$$\begin{aligned} T' &= \|(T_1', \trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m)), \\ S' &= \|(T_1', \trianglelefteq_v(\|(T_2, \ldots, T_m))). \end{aligned}$$

Puisque $\alpha(T_1') = v$ et selon la cohérence de l'axiome **Dec2**, que nous avons prouvée dans la section 12.4.2.2, on a que $T_1' \approx \trianglelefteq_v(T_1')$. En plus, la relation $\approx$ est une congruence par rapport à l'opérateur de composition parallèle $||$ (cohérence de l'axiome **Par1** prouvée dans la section 12.3.5.3). Donc,

$$\begin{aligned} T' &\approx ||(\trianglelefteq_v(T_1'), \trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m)), \\ S' &\approx ||(\trianglelefteq_v(T_1'), \trianglelefteq_v(||(T_1, \ldots, T_m))). \end{aligned}$$

Nous avons déjà prouvé la distributivité des opérateurs de décalage par rapport à la composition parallèle $||$ dans la section 12.4.2.3. Donc,

$$S' \approx ||(\trianglelefteq_v(T_1'), ||(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_m))).$$

Mais, par définition,

$$(||(\trianglelefteq_v(T_1'), \trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m)), ||(\trianglelefteq_v(T_1'), ||(\trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m))) \in \mathcal{R}_m.$$

On déduit que $(T', S') \in \mathcal{R}$.

Si $T_i' = \uparrow$ alors

$$\begin{aligned} T' &= ||(\trianglelefteq_v(T_2), \ldots, \ldots, \trianglelefteq_v(T_m)) \\ S' &= \trianglelefteq_v(||(T_2, \ldots, T_m)). \end{aligned}$$

Selon la distributivité de $\trianglelefteq$ par rapport à l'opérateur $||$ (cohérence de l'axiome **Dec3**, section 12.4.2.3), on a que $(T', S') \in \approx \subseteq \mathcal{R}$.

Si $i \geq 2$, supposons d'abord que $T_i' \neq \uparrow$. On a alors que

$$\begin{aligned} T' &= ||(\trianglelefteq_v(T_1), \ldots, T_i', \ldots, \trianglelefteq_v(T_m)) \\ S' &= ||(\trianglelefteq_v(T_1), ||(\trianglelefteq_v(T_2), \ldots, T_i', \ldots, \trianglelefteq_v(T_m))). \end{aligned}$$

On suit ensuite le même raisonnement que dans le cas $i = 1$ pour prouver que $(T', S') \in \mathcal{R}$.

Finalement, si $i \geq 2$ et $T_i' = \uparrow$, alors

$$\begin{aligned} T' &= \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{i-1}), \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_m)) \\ S' &= \|(\trianglelefteq_v(T_1), \|(\trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_{i-1}), \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_m)) \end{aligned}$$

Donc, $(T', S') \in \mathcal{R}_{m-1} \subseteq \mathcal{R}$. ∎

12.4.2.7 Cohérence de cons1

Prouvons que, pour tout $T \in \mathcal{L}_\beta$ et $\theta^{1,2} \in \Theta$:

$$\theta^1 : \theta^2 : T \;\approx\; \theta^1 \wedge \theta^2 : T.$$

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation définie comme suit:

$$\mathcal{R} = \{(\theta^1 : \theta^2 : T, \theta^1 \wedge \theta^2 : T) \mid T \in \mathcal{L}_\beta, \theta^{1,2} \in \Theta\} \cup \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Notons par

$$\begin{aligned} T_1 &= \theta^1 : \theta^2 : T \\ T_2 &= \theta^1 \wedge \theta^2 : T. \end{aligned}$$

C'est trivial de montrer que $\alpha(T_1) = \alpha(T_2)$ et

$$\begin{aligned} \Gamma(T_1) &= \mathrm{Sol}(\theta^1) \cap \Gamma(\theta^2 : T) \\ &= \mathrm{Sol}(\theta^1) \cap \mathrm{Sol}(\theta^2) \cap \Gamma(T) \\ &= \mathrm{Sol}(\theta^1 \wedge \theta^2) \cap \Gamma(T) \\ &= \Gamma(\theta^1 \wedge \theta^2 : T)) \\ &= \Gamma(T_2). \end{aligned}$$

Prouvons maintenant que $\mathcal{R}$ est une bisimulation forte. Si le terme T exécute $\delta(v)$, alors T_1 et T_2 bloquent tous les deux au temps v et transitent vers $\downarrow$. Or, $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$.

Considérons maintenant les transitions $T \stackrel{a(v)}{\rightarrow} T'$ avec $a \neq \delta$. On remarque d'abord que

$$\begin{aligned} \text{First}(a(v), T_1) &\Leftrightarrow \text{First}(a(v), T_2) \text{ et} \\ \text{First}(a(v), T_1) &\Rightarrow \text{First}(a(v), \theta^2 : T). \end{aligned}$$

On distingue trois cas: (1) le prédicat $\text{First}(a(v), \theta^2 : T)$ est faux pour toute action $a(v)$ que T peut exécuter; (2) il existe des transitions qui satisfont $\text{First}(a(v), \theta^2 : T)$ mais aucune qui satisfait $\text{First}(a(v), T_1)$; (3) il existe au moins une action $a(v)$ que T peut exécuter et qui satisfait en même temps $\text{First}(a(v), T_1)$ et $\text{First}(a(v), \theta^2 : T)$.

Dans les deux premiers cas T_1 et T_2 bloquent tous les deux au temps $v = \alpha(T_2) = \alpha(T_1) = \alpha(T)$ (par définition). Dans le troisième cas T_1 et T_2 peuvent exécuter $a(v)$ et transitent ensuite vers les terme T_1' et T_2' respectivement, où

$$\begin{aligned} T_1' &= \theta^1_{v/t_a} : \theta^2_{v/t_a} : T', \\ T_2' &= (\theta^1 \wedge \theta^2)_{v/t_a} : T' \\ &= (\theta^1_{v/t_a} \wedge \theta^2_{v/t_a}) : T'. \end{aligned}$$

Donc, $(T_1', T_2') \in \mathcal{R}$, ce qui conclut la preuve de la proposition. ∎

12.4.2.8 Cohérence de cons2

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(\text{Vrai} : T, T) \mid T \in \mathcal{L}\} \cup \approx .$$

Nous allons montrer que $\mathcal{R}$ est une bisimulation–ACTC. On remarque d'abord que $\Gamma(\text{Vrai} : T) = \Gamma(T)$ et que, par définition, $\alpha(\text{Vrai} : T) = \alpha(T)$.

C'est trivial de vérifier que $a(v) \in \text{exec}(T) \Leftrightarrow a(v) \in \text{exec}(\text{Vrai} : T)$, pour toute action temporisée $a(v)$. En plus, lorsque les termes T et $\text{Vrai} : T$ exécutent une même action, ils transitent soit vers T' et $\text{Vrai} : T'$, respectivement, ou bien tous les deux vers $\uparrow$. Or, $(T', \text{Vrai} : T') \in \mathcal{R}$ et $(\uparrow, \uparrow) \in \approx \subseteq \mathcal{R}$. ∎

12.4.2.9 Cohérence de cons3

Soient $T \in \mathcal{L}_\beta$ un terme feuille et v un instant temporel quelconques tels que $V_0 \leq v \leq \alpha(T)$. Prouvons que, pour toute action $a \in \text{Act}(T)$ on a que:

$$t_a \geq v : T \approx T.$$

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\mathcal{R} = \{(T, t_a \geq v : T) \mid T \in \mathcal{L}_\beta, V_0 \leq v \leq \alpha(T), a \in \text{Act}(T)\} \cup \approx .$$

Soit $a \in \text{Act}(T)$ une action quelconque. Notons par $S = t_a \geq v : T$. Alors, selon la définiton de α, $\alpha(T) = \alpha(S)$. L'egalité $\Gamma(T) = \Gamma(S)$ suit directement de la proposition 12.1.

Il reste à prouver que $\mathcal{R}$ est une bisimulation, c'est à dire que T et S peuvent exécuter les mêmes actions en même temps et que $(T', S') \in \mathcal{R}$, où nous notons par T' et S' les successeurs respectifs de T et S lors de l'exécution d'une même action temporisée.

Toute transition de T a lieu au temps $\alpha(T)$ ou plus tard (proposition 7.7). En plus, pour toute action $b(v')$, que T peut exécuter, le prédicat $\text{First}(b(v'), T)$ est vrai (Proposition 7.9). Donc, puisque $v \leq \alpha(T) \leq v'$, le prédicat $\text{First}(b(v'), S)$ est vrai aussi. Donc, S exécute $b(v'), b \in \text{Act}$ si et seulement si T aussi exécute $b(v')$.

Il reste à prouver que $(T', S') \in \mathcal{R}$. On remarque d'abord que $T' = \uparrow \Leftrightarrow S' = \uparrow$. Or, des paires de termes identiques sont dans $\approx$, donc aussi dans $\mathcal{R}$.

T effectue une transition $T \xrightarrow{b(v')} T'$, avec $T' \neq \downarrow$ si et seulement si S exécute $b(v')$ et transite vers $S' = (t_a \geq v)_{v'/t_b} : T'$. Si $a = b$, alors $S' = v' \geq v : T' = \text{Vrai} : T'$. Or, selon la cohérence de l'axiome **cons3** (section 12.4.2.8), $(S', T') \in \approx \subseteq \mathcal{R}$. Sinon, $S' = t_a \geq v : T'$ et, puisque $v \leq \alpha(T) \leq \alpha(T')$ (proposition 7.7), on déduit que $(T', S') \in \mathcal{R}$. ∎

12.4.2.10 Cohérence de cons4

Nous prouverons que pour tout terme $T \in \mathcal{L}_\beta$ tel que

$$T = \|(\theta^1 : T_1, \ldots, \theta^n : T_n),$$

où $\theta_i \in \Theta$ et $T_i \in \mathcal{L}_\beta$ pour tout $i, 1 \leq i \leq n$ on a que

$$\bigwedge_{i=1}^{n} \theta_i : \|(T_1, \ldots, T_n) \;\approx\; \|(\theta^1 : T_1, \ldots, \theta^n : T_n).$$

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\begin{aligned}\mathcal{R} \;=\; \{(S,T) \mid S \;\approx\; \bigwedge_{i=1}^{n} \theta^i : \|(T_1, \ldots, T_n)\; T \;\approx\; \|(\theta^1 : T_1, \ldots, \theta^n : T_n),\\ \mathrm{Act}(\theta^i) \subseteq \mathrm{Act}(T_i) \text{ et } \mathrm{Act}(T_i) \cap \mathrm{Act}(T_j) = \emptyset \text{ si } i \neq j\} \cup \;\approx .\end{aligned}$$

Nous allons prouver que $\mathcal{R}$ est une bisimulation ACTC. Pour ce faire il suffit de considérer une paire $(S,T) \in \mathcal{R}$, telle que

$$\begin{aligned} S &= \bigwedge_{i=1}^{n} \theta^i : \|(T_1, \ldots, T_n) \text{ et} \\ T &= \|(\theta^1 : T_1, \ldots, \theta^n : T_n). \end{aligned}$$

On remarque d'abord que $\alpha(S) = \alpha(T)$ et que $\Gamma(T) = \Gamma(S)$ (selon les définitons de α et Γ).

Il reste à prouver que $\mathcal{R}$ est une bisimulation forte, c'est à dire que les termes S et T peuvent exécuter les mêmes actions en même temps et que leurs successeurs respectifs lors de ces transitions forment des paires des $\mathcal{R}$.

On va d'abord considérer les situations où T et S ne bloquent pas et par la suite les situations de blocage.

Prouvons que pour toute action temporisée $a(v_a)$, $a \neq \delta$,

$$\mathrm{First}(a(v_a), T) \Leftrightarrow \mathrm{First}(a(v_a), S). \tag{12.24}$$

Cette équivalence est prouvée par les équations suivantes:

$$\mathrm{First}(a(v_a), S) \;\Leftrightarrow\; \Gamma^{\geq v_a}(S) \cap \mathrm{Sol}(t_a = v_a) \neq \emptyset$$

$$
\begin{aligned}
&\Leftrightarrow \ \mathrm{Sol}^{\geq v_a}(\bigwedge_{i=1}^{n} \theta^i) \ \cap \Gamma^{\geq v_a}(\|(T_1,\ldots,T_n)) \ \cap \mathrm{Sol}(t_a = v_a) \neq \emptyset \\
&\Leftrightarrow \ \bigcap_{i=1}^{n} \mathrm{Sol}^{\geq v_a}(\theta^i) \ \cap \bigcap_{i=1}^{n} \Gamma^{\geq v_a}(T_i) \ \cap \mathrm{Sol}(t_a = v_a) \neq \emptyset \\
&\Leftrightarrow \ \bigcap_{i=1}^{n} \Gamma^{\geq v_a}(\theta^i : T_i) \ \cap \mathrm{Sol}(t_a = v_a) \neq \emptyset \\
&\Leftrightarrow \ \bigwedge_{i=1}^{n} \mathrm{First}(a(v_a), \theta^i : T_i) \\
&\Leftrightarrow \ \mathrm{First}(a(v_a), T).
\end{aligned}
$$

Montrons ensuite que

$$\mathrm{exec}^{\delta}(T) = \mathrm{exec}^{\delta}(S). \tag{12.25}$$

Nous allons prouver seulement l'inclusion

$$\mathrm{exec}^{\delta}(S) \subset \mathrm{exec}^{\delta}(T),$$

c'est à dire que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ avec $a \neq \delta$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$. La preuve de l'inclusion inverse est analogue.

Supposons que S exécute une action $a(v)$, avec $a \neq \delta$. Ceci implique que le sous-terme $\|(T_1,\ldots,T_2)$ exécute $a(v)$ et le prédicat $\mathrm{First}(a(v), S)$ est vrai. Or, si $\|(T_1,\ldots,T_n)$ exécute $a(v)$ il existe un sous-terme T_k, avec $1 \leq k \leq n$ qui exécute $a(v)$ et le prédicat $\mathrm{Wait}(v, T_i)$ est vrai pour tout $i \neq k$.

Puisque, par l'équation 12.24, les prédicats $\mathrm{First}(a(v), S)$ et $\mathrm{First}(a(v), T)$ sont équivalents, et que

$$\mathrm{First}(a(v), T) \Rightarrow \mathrm{First}(a(v), \theta^k : T_k)$$

on déduit que le terme $\theta^k : T_k$ peut alors exécuter $a(v)$ et transite vers $\theta^k_{v/t_a} : T'_k$. Il reste à vérifier que les prédicats $\mathrm{Wait}(v, \theta^i : T_i)$ sont vrais, pour tout $i \neq k, 2 \leq i \leq n$.

Par conséquent il existe une transition

$$T \stackrel{a(v)}{\rightarrow} T'$$

ce qu'il fallait prouver.

Considérons maintenant les situations de blocage et prouvons que

$$S \overset{\delta(v)}{\rightarrow} \downarrow \Leftrightarrow T \overset{\delta(v)}{\rightarrow} \downarrow \ . \tag{12.26}$$

S exécute $\delta(v)$ si et seulement si une des deux situations suivantes se produit:

1. il existe un sous-terme $T_i, 1 \leq i \leq n$, qui exécute $\delta(v)$ et pour tous les autres sous-termes $T_j, j \neq i$ le prédicat $\text{Wait}_v(T_j)$ est vrai; ou bien

2. le prédicat $\text{First}(a(v'), S)$ est faux pour toute transition $\|(T_1, \ldots, T_n) \overset{a(v')}{\rightarrow} S''$ et le blocage de S se produit alors au temps $v = \alpha(S)$.

Dans le premier cas (1), puisque $T_i \overset{\delta(v)}{\rightarrow} \downarrow$, le termes $\theta^i : T_i$ aussi bloque au temps v. Or, par la proposition 6.1, puisque $\mathcal{T}_i$ est un terme feuille, on a que

$$\begin{aligned} v &\overset{\text{prop.6.1}}{=} \alpha(T_i) \overset{\text{def}}{=} \alpha(\theta^i : T_i) \\ &\overset{\text{def}}{=} \alpha(T_j) \overset{\text{def}}{=} \alpha(\theta^j : T_j) \\ &\overset{\text{def}}{=} \alpha(T) = \alpha(S), \end{aligned}$$

pour tout $j, 1 \leq j \leq n$.

Selon la proposition 7.11 tout terme $T \in \mathcal{L}$ peut exécuter au moins une action. De plus, selon la proposition 7.7, $\text{exec}(T) = \text{exec}^{\geq \alpha(T)}(T)$. On déduit que le prédicat $\text{Wait}(\alpha(T), T)$ est vrai pour tout terme T. Donc,

$$(\forall j \neq i, 1 \leq j \leq n)\text{Wait}_v(\theta^j : T_j) = \text{Vrai}.$$

Par conséquent, le terme

$$T = \|(\theta^1 : T_1, \ldots, \theta^i : T_i, \ldots, \theta^n : T_n)$$

exécute l'action de blocage $\delta(v)$, ce qu'il fallait montrer.

Dans le cas (2), le prédicat $\text{First}(a(v_a), S)$ est faux pour toute action $a(v_a) \in \text{exec}(S)$. Or, par les équations 12.24 et 12.25 il suit que $\text{First}(a(v_a, T)$ est faux pour

toute action $a(v_a) \in \text{exec}(T)$. Donc, T exécute $\delta(\alpha(T))$. De plus, nous avons déjà montré que $\alpha(T) = alpha(S)$. Nous pouvons maintenant conclure que

$$\text{exec}(T) = \text{exec}(S).$$

Il reste alors à prouver que $\mathcal{R}$ est fermée aux transitions, autrement dit que pour toute action $a(v) \in \text{exec}(S) = \text{exec}(T)$

- $(\forall S' \in \text{succ}(a(v), S))(\exists T' \in \text{succ}(a(v), T))((S', T') \in \mathcal{R})$;
- $(\forall T' \in \text{succ}(a(v), T))(\exists S' \in \text{succ}(a(v), S))((S', T') \in \mathcal{R})$.

Nous prouvons la première propriété seulement; la preuve de la propriété symétrique est analogue.

Le cas où $a = \delta$ est trivial: on a que $S' = T' = \downarrow$, donc $(S', T') \in \approx \subseteq \mathcal{R}$.

Si $a \neq \delta$, alors $a(v) \in \text{exec}(S)$ seulement s'il existe un sous-terme $T_k, 1 \leq k \leq n$ tel que $T_k \xrightarrow{a(v)} T'_k$. Nous distinguons alors les situations où $T'_k = \uparrow$ et $T'_k \neq \uparrow$.

Si $T'_1 = \uparrow$ alors

$$\begin{aligned} S' &= \bigwedge_{i=1}^{n} \theta^i_{v/t_a} : \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{k-1}), \trianglelefteq_v(T_{k+1}), \ldots, \trianglelefteq_v(T_n)) \text{ et} \\ T' &= \|(\trianglelefteq_v(\theta^1 : T_1), \ldots, \trianglelefteq_v(\theta^{k-1} : T_{k-1}), \trianglelefteq_v(\theta^{k+1} : T_{k+1}), \ldots, \trianglelefteq_v(\theta^n : T_n)). \end{aligned}$$

Puisque $\trianglelefteq$ est distributif par rapport à θ (théorème 9.6, section 12.4.2.3),

$$T' \approx \|(\theta^1 : \trianglelefteq_v(T_1), \ldots, \theta^{k-1} : \trianglelefteq_v(T_{k-1}), \theta^{k+1} : \trianglelefteq_v(T_{k+1}), \ldots, \theta^n : \trianglelefteq_v(T_n)).$$

Étant donné que pour tout $j \neq k$ le prédicat $\text{Wait}(v, \theta^j : T_j)$ est vrai, il existe des transitions que les termes $\theta^j : T_j$ peuvent effectuer au temps v ou plus tard, donc les termes $\trianglelefteq_v(\theta^j : T_j) \approx \theta^j : \trianglelefteq_v(T_j)$ ne bloquent pas. Il suffit alors de remarquer que, puisque $\text{Act}(T_i) \cap \text{Act}(T_k) = \emptyset$ pour tout i et que $\text{Act}(\theta^i \subseteq \text{Act}(T_i)$, pour tout $i \neq k$, alors

$$S' \approx \bigwedge_{i \neq k, i=1}^{n} \theta^i_{v/t_a} : \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{k-1}), \trianglelefteq_v(T_{k+1}), \ldots, \trianglelefteq_v(T_n)).$$

Donc, $(S', T') \in \mathcal{R}$, ce qu'il fallait démontrer.

Si $T'_k \neq \uparrow$ alors,

$$\begin{aligned} S' &= \bigwedge_{j=1}^{n} \theta^j_{v/t_a} : \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{k-1}), T'_k, \trianglelefteq_v(T_{k+1}), \ldots, \trianglelefteq_v(T_n)) \text{ et} \\ T' &= \|(\trianglelefteq_v(\theta^1 : T_1), \ldots, \trianglelefteq_v(\theta^{k-1} : T_{k-1}), \theta^k_{v/t_a} : T'_k, \trianglelefteq_v(\theta^{k+1} : T_{k+1}), \ldots, \trianglelefteq_v(\theta^n : T_n)). \end{aligned}$$

Selon la distributivité de $\trianglelefteq$ par rapport à θ (théorème 9.6, section 12.4.2.3) et puisque $\approx$ et une congruence par rapport à l'opérateur $\|$ (théorème 8.10), nous avons que

$$T' \approx \|(\theta^1 : \trianglelefteq_v(T_1), \ldots, \theta^{k-1} : \trianglelefteq_v(T_{k-1}), \theta^k_{v/t_a} : T'_k, \theta^{k+1} : \trianglelefteq_v(T_{k+1}), \ldots, \theta^n : \trianglelefteq_v(T_n)).$$

Donc, $(S', T') \in \mathcal{R}$.

On peut, donc, conclure que $\mathcal{R}$ est une bisimulation. ∎

12.4.2.11 Cohérence de cons5

Montrons que, pour tout $\theta \in \Theta$, tout $T \in \mathcal{L}_\beta$, et tout opérateur réactif ROp,

$$\theta : \mathrm{ROp}(T) \approx \mathrm{ROp}(\theta : T).$$

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\begin{aligned} \mathcal{R} = \{ \quad (\quad & \theta : \mathrm{ROp}(o, H, [m, M], T), \mathrm{ROp}(o, H, [m, M], \theta : T)) \mid \\ & T \in \mathcal{L}_\beta, \theta \in \Theta, \mathrm{ROp} = \text{Max ou Min}\} \cup \approx . \end{aligned}$$

On va prouver que $\mathcal{R}$ est une bisimulation–ACTC. Puisque les preuves pour les opérateurs Max et Min sont analogues, on ne va montrer que le cas ROp = Max.

Nous introduisons les notations suivantes:

$$\begin{aligned} S_1 &= ROp(o, H, [m, M], T) \\ S_2 &= \theta : T \\ T_1 &= \theta : S_1 \\ T_2 &= \mathrm{ROp}(o, H, [m, M], S_2). \end{aligned}$$

Selon le raisonnement introduit dans la section 8.2, il suffit de considérer les paires (T_1, T_2).

Par définition $\alpha(T_1) = \alpha(T_2)$. De plus, c'est trivial de vérifier que $\Gamma(T_1) = \Gamma(T_2)$. Il reste à prouver que $\mathcal{R}$ est une bisimulation forte.

Parmi les transitions $T_i \stackrel{a(v)}{\rightarrow} T_i'$ de T_1 et T_2 on distingue les transitions héritées de T (où l'action $a(v)$ est aussi exécutée par le sous-terme T) et les blocages sémantiques (introduits par la sémantique des opérateurs de composition θ et Max).

Considérons d'abord les transitions héritées de T. Le cas où T exécute $\delta(v)$ est trivial: le termes S_i et T_i bloquent au temps v et transitent vers $\downarrow$. Il suffit alors de remarquer que $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$.

Dans le cas des transitions qui ne signalent pas un blocage notons que

$$\text{First}(a(v), T_1) \Leftrightarrow \text{First}(a(v), S_2) \Leftrightarrow \text{First}(a(v), T_2).$$

Donc, si T peut effectuer une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq o, \delta$ et telle que le prédicat $\text{First}(a(v), S_2)$ soit vrai, T_1 et T_2 peuvent tous les deux exécuter $a(v)$. Il suffit alors de montrer que $(T_1', T_2') \in \mathcal{R}$.

Lorsque $T' = \uparrow$, T_1' et T_2' sont aussi égaux à $\uparrow$ et la propriété est satisfaite trivialement. Autrement, si $T' \neq \uparrow$ on doit considérer deux cas: (1) $H \neq \{a\}$ et (2) $H = \{a\}$. Dans le premier cas les termes T_1' et T_2' forment une paire de $\mathcal{R}$ puisque:

$$T_1' = \theta_{v/t_a} : \text{ROp}(o, H \setminus \{a\}, [m, M], T')$$
$$T_2' = \text{ROp}(o, H \setminus \{a\}, [m, M], \theta_{v/t_a} : T').$$

Dans le deuxième cas on a que:

$$T_1' = \theta_{v/t_a} : m \leq t_o - v \leq M : T'$$
$$T_2' = m \leq t_o - v \leq M : \theta_{v/t_a} : T'$$

et, selon la commutativité de l'opérateur θ (corollaire 2 et théorème 9.6),

$$(T_1', T_2') \in \approx \subseteq \mathcal{R}$$

.

Considérons maintenant les situations de blocage des termes T_1 et T_2. Ces blocages sont toujours signalés au temps de démarrage de T_1 et T_2, respectivement. Dans cette discussion nous incluons aussi les blocages des sous-termes S_1 et S_2, hérités par T_1 et T_2, respectivement.

On remarque que, selon la définiton de α,

$$\begin{aligned} \alpha(T) &= \alpha(S_2) = \alpha(T_2) \\ &= \alpha(S_1) = \alpha(T_1) \\ &\stackrel{\text{not.}}{=} \alpha. \end{aligned}$$

Puisque après un blocage T_1 et T_2 transitent vers le même état $\downarrow$, il suffit alors de prouver que T_1 signale un blocage sémantique si et seulement si T_2 bloque.

Nous devons considérer trois cas : (1) aucune transition de T ne satisfait les contraintes θ, (2) la seule action que T peut exécuter est l'action puits o, et (3) les situations (1) et (2) combinées, c'est à dire que T ne peut exécuter que l'action source o et aucune de ces exécutions ne satisfait θ.

Si (1), alors d'une part S_2 exécute $\delta(v)$, et d'une autre part S_1 non plus ne peut effectuer aucune transition qui satisfait θ. Par conséquent T_1 et T_2 exécutent $\delta(\alpha)$. Un raisonnement analogue s'applique aussi dans la situation (2). Finalement, si (3) alors S_1 et S_2 signalent tous les deux un blocage sémantique au temps α et, par conséquent, T_1 et T_2 exécutent $\delta(\alpha)$. ∎

12.4.2.12 Cohérence de ROp1

Nous voulons prouver que $\mathrm{ROp}_1(\mathrm{ROp}_2(T)) \approx \mathrm{ROp}_2(\mathrm{ROp}_1(T))$, pour tous $T \in \mathcal{L}_\beta, \mathrm{ROp}_i = \mathrm{Min/Max}(o_i, H_i, [m_i, M_i], \cdot)$.

On distingue trois situations différentes:

1. $\mathrm{ROp}_1 = \mathrm{Max}, \mathrm{ROp}_2 = \mathrm{Max}$;

2. $\mathrm{ROp}_1 = \mathrm{Max}, \mathrm{ROp}_2 = \mathrm{Min}$;

3. $\mathrm{ROp}_1 = \mathrm{Min}, \mathrm{ROp}_2 = \mathrm{Min}$.

Nous allons considérer seulement la première situation, car les deux autres sont analogues.

Soit $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\begin{aligned}\mathcal{R} &= \{(\mathrm{ROp}_1(\mathrm{ROp}_2(T)), \mathrm{ROp}_2(\mathrm{ROp}_1(T_1))) \mid \\ &\quad T_1 \in \mathcal{L}_\beta, \mathrm{ROp}_i = \mathrm{Max}(o_i, H_i, [m_i, M_i], \cdot), i = 1, 2\} \cup \approx .\end{aligned}$$

On va prouver que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire nous considérons une paire quelconque $(S, T) \in \mathcal{R}$ telle que

$$\begin{aligned}S &= \mathrm{Max}(o_1, H_1, [m_1, M_1], \mathrm{Max}(o_2, H_2, [m_2, M_2], (T_1)) \\ T &= \mathrm{Max}(o_2, H_2, [m_2, M_2], \mathrm{Max}(o_1, H_1, [m_1, M_1], (T_1)).\end{aligned}$$

et prouvons qu'elle satisfait aux trois conditions énoncées dans la définition 8.1 d'une bisimulation–ACTC (voir raisonnement de la section 8.2).

Les deux premières conditions sont triviales: par définition $\Gamma(S) = \Gamma(T) = \Gamma(T_1)$ et $\alpha(S) = \alpha(T) = \alpha(T_1) = \alpha$.

Il reste à montrer que $\mathcal{R}$ est une bisimulation forte. Nous allons commencer par prouver que T exécute $\delta(v)$ si et seulement si S exécute $\delta(v)$, pour tout $v \geq V_0$. Lors d'un blocage les termes transitent vers $\downarrow$ et $(\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$.

Il est évident que si T_1 exécute $\delta(v)$, alors T et S bloquent au temps v. Supposons maintenant que la seule action que T_1 peut exécuter est o_2 (si l'unique action que T_1 peut exécuter est o_1, on suit un raisonnement analogue). Dans ce cas le sous-terme $\mathrm{Max}(o_2, H_2, [m_2, M_2], T_1)$, ainsi que le terme S exécutent l'action $\delta(\alpha)$. Or, puisque $o_1 \neq o_2$,

$$\mathrm{exec}(\mathrm{Max}(o_1, H_1, [m_1, M_1], T_1)) \subseteq \mathrm{exec}(T_1).$$

Donc, $\mathrm{Max}(o_1, H_1, [m_1, M_1], T_1)$ ne peut exécuter que l'action o_2. Par conséquent T exécute $\delta(\alpha)$, ce qu'il fallait prouver.

Supposons maintenant que T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$. Implicitement, alors $a \notin \{o_1, o_2\}$ et $a(v) \in \text{exec}(T_1)$. Par conséquent, S peut effectuer la transition $S \stackrel{a(v)}{\rightarrow} S'$. De manière analogue on prouve que si S peut exécuter une action $a(v)$, alors T aussi peut l'exécuter.

Il reste à prouver que $(S', T') \in \mathcal{R}$. Si $T_1' = \uparrow$, alors $T' = S' = \uparrow$ et $(S', T') \in \approx \subseteq \mathcal{R}$ (théorème 8.1).

Si $H_1 = H_2 = \{a\}$, alors

$$\begin{aligned} S' &= m_1 \leq t_{o_1} - v \leq M_1 : m_2 \leq t_{o_2} - v \leq M_2 : T_1' \\ T' &= m_2 \leq t_{o_2} - v \leq M_2 : m_1 \leq t_{o_1} - v \leq M_1 : T_1'. \end{aligned}$$

Or, nous avons déjà prouvé que l'opérateur θ est commutatif (cohérence de l'axiome **cons1**, section 2). Donc, $(S', T') \in \approx \subseteq \mathcal{R}$.

Si $H_1 = \{a\}$ et $|H_2 \setminus \{a\}| \geq 1$, alors

$$\begin{aligned} S' &= m_1 \leq t_{o_1} - v \leq M_1 : \text{Max}(o_2, H_2 \setminus \{a\}, [m_2, M_2], T_1') \\ T' &= \text{Max}(o_2, H_2 \setminus \{a\}, [m_2, M_2], m_1 \leq t_{o_1} - v \leq M_1 : T_1'). \end{aligned}$$

Puisque θ est distributif par rapport à Max (cohérence de l'axiome **cons6**, section 12.4.2.11), on déduit que $(S', T') \in \approx \subseteq \mathcal{R}$.

Finalement, si $|H_2 \setminus \{a\}| \geq 1, |H_2 \setminus \{a\}| \geq 1$, alors

$$\begin{aligned} S' &= \text{Max}(o_1, H_1 \setminus \{a\}, [m_1, M_1], \text{Max}(o_2, H_2 \setminus \{a\}, [m_2, M_2], T_1')) \text{ et} \\ T' &= \text{Max}(o_2, H_2 \setminus \{a\}, [m_2, M_2], \text{Max}(o_1, H_1 \setminus \{a\}, [m_1, M_1], T_1')). \end{aligned}$$

Par définition, on a alors que $(S', T') \in \mathcal{R}$. ∎

12.4.2.13 Cohérence de ROp2

Nous prouvons que pour tous $T_1, \ldots, T_m \in \mathcal{L}_\beta$, où $m \geq 2$ et tout opérateur réactif $\text{ROp}(o, H, [m, M], \cdot)$ s'il existe un sous-terme T_i, avec $1 \leq i \leq m$, tel que $H \cup \{o\} \subseteq$

$\mathrm{Act}(T_i)$, alors

$$\mathrm{ROp}(o, H, [m, M], \|(T_1, \ldots, T_m)) \;\approx\; \|(T_1, \ldots, \mathrm{ROp}(o, H, [m, M], T_i), \ldots, T_m).$$

Puisque $\|$ est associatif et commutatif il suffit de considérer juste les compositions en parallèle de deux sous-termes seulement et supposer que $i = 2$. Nous prouvons la proposition avec $\mathrm{ROp} = \mathrm{Max}(o, H, [m, M], \cdot)$. Si $\mathrm{ROp} = \mathrm{Min}(o, H, [m, M], \cdot)$ la preuve est analogue.

Nous définissons la relation suivante $\mathcal{R} \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$

$$\begin{aligned} \mathcal{R}_1 &= \{(\mathrm{ROp}(\|(T_1, T_2)), \|(T_1, \mathrm{ROp}(T_2))) \mid \{o\} \cup H \subseteq \mathrm{Act}(T_2)\} \\ \mathcal{R} &= \{(T, S) \mid T, S \in \mathcal{L}, (\exists (T', S') \in \mathcal{R}_1)(T \approx T' \wedge S \approx S')\} \cup \approx \end{aligned}$$

et remarquons que pour toute paire $(T, S) \in \mathcal{R}_1$ c'est trivial de montrer que $\alpha(T) = \alpha(S)$, que $\Gamma(T) = \Gamma(S)$ et et que (T, S) a les propriétés énoncées dans la définition 4.2. ∎

12.4.2.14 Cohérence de C1

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(+(T_1, T_2), +(T_2, T_1)) \mid T_{1,2} \in \mathcal{L}\} \cup \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Soit $(S, T) \in \mathcal{R}$ une paire quelconque de R telle que

$$S = +(T_1, T_2)$$
$$T = +(T_2, T_1).$$

Premièrement, $\Gamma(S) = \Gamma(T)$ et $\alpha(T) = \alpha(S) = \alpha(T_i)$ (proposition 7.2), avec $i = 1, 2$. Il reste à montrer que (S, T) satisfait aux conditions énoncées dans la définition d'une bisimulation forte, c'est à dire que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$ avec $(S', T') \in \mathcal{R}$. Or, il suffit de remarquer que les règles de sémantique opérationnelle de S et T sont identiques et, donc, que $\mathrm{exec}(S) = \mathrm{exec}(T)$ et $S' = T'$. ∎

12.4.2.15 Cohérence de C2

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\sum(T_1, \ldots, T_m), \sum(T_1, \sum(T_2, \ldots, T_m))) \mid m \geq 3, T_i \in \mathcal{L}, i = 1, \ldots, m\} \cup \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Pour ce faire il suffit de prouver qu'une paire quelconque $(S, T) \in \mathcal{R}$, telle que

$$\begin{aligned} S &= \sum(T_1, \ldots, T_m) \\ T &= \sum(T_1, \sum(T_2, \ldots, T_m)), \end{aligned}$$

satisfait aux conditions énoncées dans la définition d'une bisimulation– ACTC.

Les égalités $\Gamma(T) = \Gamma(S)$ et $\alpha(S) = \alpha(T) = \alpha(T_i)$ sont triviales.

Supposons que S effectue une transition $S \stackrel{a(v)}{\rightarrow} S'$. Nous voulons montrer qu'il existe alors une transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $(S', T') \in \mathcal{R}$. La preuve de la propriété symétrique, c'est à dire que pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$, avec $(S', T') \in \mathcal{R}$, est analogue.

Si $a = \delta$, alors il existe un sous-terme T_i qui bloque au temps v et le prédicat $\text{Wait}(v, T_j)$ est faux, pour tout $j \neq i$. Si $2 \leq i \leq m$ il suit directement de la sémantique opérationnelle de l'opérateur $\sum$ que $T \stackrel{\delta(v)}{\rightarrow} \downarrow$. Si $i = 1$, alors selon la proposition 8.6 le prédicat $\text{Wait}(v, \sum(T_2, \ldots, T_m))$ est faux. Donc, T exécute $\delta(v)$.

Autrement, si $a \neq \delta$ alors il existe un sous-terme T_i qui effectue une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et $S' = T_i'$. Si $i = 1$ alors il suffit d'appliquer le règles de sémantique opérationnelle $\sum_{1,2}$ pour conclure que $T \stackrel{a(v)}{\rightarrow} T'$ et $T' = T_i'$. Si $2 \leq i \leq m$, alors premièrement

$$\sum(T_2, \ldots, T_m) \stackrel{a(v)}{\rightarrow} T_i'$$

et implicitement $T \stackrel{a(v)}{\rightarrow} T_i'$, Donc, $(S', T') \in \approx \subseteq \mathcal{R}$. ∎

12.4.2.16 Cohérence de C3

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante

$$\mathcal{R} = \{(\sum(T,T) \approx T) \mid T \in \mathcal{L}\} \cup \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Soient $S, T \in \mathcal{L}$ tels que $S = \sum(T,T)$. Alors, par définition

$$\alpha(S) = \min(\alpha(T), \alpha(T))$$

et

$$\Gamma(S) = \Gamma(T) \cup \Gamma(T) = \Gamma(T).$$

De plus, c'est trivial de prouver que $\text{exec}(T) = \text{exec}(S)$ et que pour toute action $a(v) \in \text{exec}(T)$, $\text{succ}(T, a(v)) = \text{succ}(S, a(v))$. Or, comme toute paire de termes identiques est dans $\approx$ (théorème 8.1), on peut déduire que $\mathcal{R}$ est une bisimulation–ACTC. ∎

12.4.2.17 Cohérence de Seq1

Nous allons prouver que la relation $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$, définie comme suit

$$\mathcal{R} = \{(\text{Seq}(T_1, \text{Seq}(T_2, T_3)), \text{Seq}(T_1, T_2), T_3)) \mid T_{1,2,3} \in \mathcal{L}\} \cup \approx$$

est une bisimulation–ACTC. Soient $T_{1,2,3}, T, S \in \mathcal{L}$ tels que

$$\begin{aligned} T &= \text{Seq}(T_1, \text{Seq}(T_2, T_3)) \text{ et} \\ S &= \text{Seq}(T_1, T_2, T_3). \end{aligned}$$

D'abord, selon la définition de α,

$$\begin{aligned} \alpha(T) &= \min(\alpha(T_1), \alpha(\text{Seq}(T_2, T_3)) \\ &= \min(\alpha(T_1), \min(\alpha(\text{Seq}(T_2), \alpha(T_3))) \\ &= \min(\alpha(T_1), \alpha(T_2), \alpha(T_3)) \\ &= \alpha(S). \end{aligned}$$

L'égalité

$$\Gamma(T) = \Gamma(S)$$

est déduite directement à partir de la définiton de la fonction Γ. Il reste à prouver que la paire (T, S) a les propriétés énoncées dans la définition d'une bisimulation forte.

Premièrement, il est évident que

$$\begin{aligned} \text{exec}(S) &= \text{exec}(\text{Seq}(T_1, T_2)) && (12.27)\\ &= \text{exec}(T_1) && (12.28)\\ &= \text{exec}(T). && (12.29) \end{aligned}$$

Prouvons ensuite que pour toute action $a(v) \in \text{exec}(S)$ et tous les successeurs $S' = \text{succ}(S, a(v))$ et $T' = \text{succ}(T, a(v))$ sont tels que $(T', S') \in \mathcal{R}$.

Si $a = \delta$ il n'y a rien à prouver: $(T', S') = (\downarrow, \downarrow) \in \approx \subseteq \mathcal{R}$. Sinon, alors à toute exécution de l'action $a(v)$ par T ou S il correspond une transition $T_1 \stackrel{a(v)}{\rightarrow} T_1'$. Si $T_1' \neq \surd$ alors

$$\begin{aligned} T' &= \text{Seq}(T_1', \text{Seq}(T_2, T_3))\\ S' &= \text{Seq}(\text{Seq}(T_1', T_2), T_3). \end{aligned}$$

Donc, par définiton $(T', S') \in \mathcal{R}$. Autrement, si $T_1' = \uparrow$ alors,

$$\begin{aligned} T' &= \trianglelefteq_v(\text{Seq}(T_2, T_3))\\ S' &= \text{Seq}(\trianglelefteq_v(T_2), T_3). \end{aligned}$$

Or, selon la distributivité des opérateurs de décalage par rapport à l'opérateur de composition séquentielle (cohérence de l'axiome **Dec3**, section 12.4.2.3), on a que $(T', S') \in \approx \subseteq \mathcal{R}$. ∎

12.4.2.18 Cohérence de Seq2

Nous voulons montrer que, pour tous $T_1, T_2, T_3 \in \mathcal{L}$:

$$\text{Seq}(+(T_1, T_2), T_3) \approx +(\text{Seq}(T_1, T_3), \text{Seq}(T_2, T_3)).$$

Pour ce faire, nous définissons la relation $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ suivante:

$$\mathcal{R} = \{(\text{Seq}(+(T_1, T_2), T_3), +(\text{Seq}(T_1, T_3), \text{Seq}(T_2, T_3)) \in \mathcal{L} \times \mathcal{L} \mid T_1, T_2, T_3 \in \mathcal{L}\} \cup \approx$$

et prouvons que $\mathcal{R}$ est une bisimulation–ACTC.

Soit $(T, S) \in \mathcal{R}$ une paire quelconque de $\mathcal{R}$ telle que

$$\begin{aligned} T &= \text{Seq}(+(T_1, T_2), T_3) \text{ et} \\ S &= +(\text{Seq}(T_1, T_3), \text{Seq}(T_2, T_3)), \end{aligned}$$

avec $T_1, T_2, T_3 \in \mathcal{L}$. Alors, selon le lemme 7.2,

$$\alpha(T) = \alpha(T_i) = \alpha(S),$$

pour tout $i \in \{1, 2, 3\}$. De plus, par définition de la fonction Γ

$$\begin{aligned} \Gamma(T) &\stackrel{\text{def}}{=} \Gamma(+(T_1, T_2)) \stackrel{\text{def}}{=} \Gamma(T_1) \cup \Gamma(T_2), \\ \Gamma(S) &\stackrel{\text{def}}{=} \Gamma(\text{Seq}(T_1, T_3)) \cup \Gamma(\text{Seq}(T_2, T_3)) \stackrel{\text{def}}{=} \Gamma(T_1) \cup \Gamma(T_2). \end{aligned}$$

Donc, $\alpha(T) = \alpha(S)$ et $\Gamma(T) = \Gamma(S)$. Ensuite, selon les règles de SOp on remarque que

$$\begin{aligned} \text{exec}(T) &= \text{exec}(+(T_1, T_2)) = \text{exec}(T_1) \cup \text{exec}(T_2), \\ \text{exec}(S) &= \text{exec}(\text{Seq}(T_1, T_3)) \cup \text{exec}(\text{Seq}(T_2, T_3)) = \text{exec}(T_1) \cup \text{exec}(T_2). \end{aligned}$$

On déduit que T peut exécuter une action quelconque $a(v)$ si et seulement si S peut exećuter $a(v)$. Il reste à prouver maintenant que la relation $\mathcal{R}$ est fermée aux transitions, c'est à dire que pour tout successeur T' de T après l'exécution d'une action quelconque $a(v)$ il existe un successeur S' de S tel que $(T', S') \in \mathcal{R}$ et vice-versa.

Soit $T \stackrel{a(v)}{\rightarrow} T'$ une transition quelconque de T. Puisque $\text{exec}(T) = \text{exec}(S)$ il existe donc une transition $S \stackrel{a(v)}{\rightarrow} S'$.

On distingue les situations suivantes:

1. $+(T_1, T_2) \stackrel{a(v)}{\rightarrow} \uparrow$ et $T' = \lhd_v(T_3)$,

2. $+(T_1, T_2) \stackrel{\delta(v)}{\rightarrow} \downarrow$ et $T' = \downarrow$, ou bien

3. $+(T_1, T_2) \stackrel{a(v)}{\rightarrow} T_i'$ et $T' = \mathrm{Seq}(T_i', T_3)$.

Dans le premier cas il faut qu'un des sous-termes T_i de T effectue la transition $T_1 \stackrel{a(v)}{\rightarrow} \uparrow$. Supposons, sans perte de généralité que $i = 1$. Alors il existe une transition $\mathrm{Seq}(T_1, T_3) \stackrel{a(v)}{\rightarrow} \lhd_v(T_3)$. Donc, S peut effecuter la transition suivante:

$$S \stackrel{a(v)}{\rightarrow} S' = \lhd_v(T_3) = T'.$$

Or, puisque $\approx$ est reflexive, $(T', S') \in \approx \subseteq \mathcal{R}$.

Dans le deuxième cas il y a deux possibilités:

- 2.(a) soit $T_1 \stackrel{\delta(v)}{\rightarrow} \downarrow$ et $T_2 \stackrel{\delta(v)}{\rightarrow} \downarrow$ ou bien
- 2.(b) $T_i \stackrel{\delta(v)}{\rightarrow} \downarrow$ et $\neg\mathrm{Wait}(v, T_j)$ avec $\{i, j\} = \{1, 2\}$; supposons, sans perte de généralité, que $i = 1$ et que $j = 2$.

Dans le cas $2.(a)$ on déduit que les sous-termes $\mathrm{Seq}(T_1, T_3)$ et $\mathrm{Seq}(T_2, T_3)$ bloquent aussi aux temps v. Donc, S effectue la transition $S \stackrel{\delta(v)}{\rightarrow} \downarrow$. Dans la situation $2.(b)$ on a que $\mathrm{Seq}(T_1, T_3) \stackrel{\delta(v)}{\rightarrow}$ et, puisque $\mathrm{exec}(\mathrm{Seq}(T_2, T_3)) = \mathrm{exec}(T_2)$, le prédicat $\mathrm{Wait}(v, \mathrm{Seq}(T_2, T_3))$ est faux. Donc, S bloque au temps v. On déduit que $T' = S' = \downarrow$ et, par reflexivité de la relation $\approx$, $(T', S') \in \approx \subseteq \mathcal{R}$.

Finalement, dans le troisième cas, il faut qu'un des sous-termes T_i de T effectue une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$, avec $T_i' \notin \{\uparrow, \downarrow\}$. Supposons, sans perte de généralité, que $i = 1$. On déduit alors qu'il existe une transition

$$\mathrm{Seq}(T_1, T_3) \stackrel{a(v)}{\rightarrow} \mathrm{Seq}(T_1', T_3).$$

Or, ceci implique qu'il existe une transition

$$S \stackrel{a(v)}{\rightarrow} S' = \mathrm{Seq}(T_1', T_3).$$

Donc, encore une fois, $T' = S'$ et $(T', S') \in \approx \subseteq \mathcal{R}$.

L'implication inverse, c'est à dire que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$, est analogue. Donc, on peut conclure que la proposition est vraie. ∎

12.4.2.19 Cohérence de Seq3

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(\text{Seq}(T_1, \oplus(T_2, T_3)), \oplus(\text{Seq}(T_1, T_2), \text{Seq}(T_1, T_3))) \mid$$
$$T_1, T_2, T_3 \in \mathcal{L},\ T_1 \text{ est un } \textit{observateur}\} \cup \approx .$$

Rappelons qu'un *observateur* est un terme qui ne contient pas des actions de direction sortie. Prouvons que $\mathcal{R}$ est une bisimulation–ACTC.

Soit $(T, S) \in \mathcal{R}$ une paire telle que

$$\begin{aligned} T &= \text{Seq}(T_1, \oplus(T_2, T_3)), \\ S &= \oplus(\text{Seq}(T_1, T_2), \text{Seq}(T_1, T_3)). \end{aligned}$$

Les égalités $\Gamma(T) = \Gamma(S)$ et $\alpha(T) = \alpha(S)$ suivent directement des définitions de Γ et α, respectivement.

Il reste à prouver que (T, S) satisfait aux conditions énoncées dans la définiton d'une bisimulation forte.

T_1 ne contient pas des actions de sortie et par conséquent la règle $\mathbf{CR}_4$ ne s'applique jamais. On peut alors déduire les équations suivantes: sont évidentes:

$$\begin{aligned} \text{exec}(S) &= \text{exec}(\text{Seq}(T_1, T_2)) \cup \text{exec}(\text{Seq}(T_1, T_3)) && (12.30) \\ &= \text{exec}(T_1) \cup \text{exec}(T_1) && (12.31) \\ &= \text{exec}(T_1) && (12.32) \\ &= \text{exec}(T). && (12.33) \end{aligned}$$

Il reste à prouver que $\mathcal{R}$ est fermée aux transitions. Soit $a(v) \in \text{exec}(T)$ une action quelconque, différente de δ. Notons par $T' = \text{succ}(a(v), T)$, $S' = \text{succ}(a(v), S)$, et $T_1' = \text{succ}(a(v), T_1)$.

Si $T_1' \neq \surd$, alors

$$\begin{aligned} T' &= \mathrm{Seq}(T_1', \oplus(T_2, T_3)) \\ S' &= \oplus(\mathrm{Seq}(T_1', T_2), \mathrm{Seq}(T_1', T_2). \end{aligned}$$

Donc, $(T', S') \in \mathcal{R}$.

Si $T_1' = \uparrow$, alors

$$\begin{aligned} T' &= \trianglelefteq_v(\oplus(T_2, T_3)), \\ S' &= \oplus(\trianglelefteq_v(T_2), \trianglelefteq_v(T_3)). \end{aligned}$$

Or, l'opérateur $\trianglelefteq$ est distributif par rapport à l'opérateur $\oplus$ (cohérence de l'axiome **Dec3**). Par conséquent, $(T', S') \in \approx \subseteq \mathcal{R}$. ∎

12.4.2.20 Cohérence de Seq4

La preuve de la distributivité droite de la composition séquentielle par rapport à l'opérateur de choix retardé est similaire à celle présentée pour l'opérateur de choix non-déterministe, dans la section 12.4.2.18, et sera omise du présent document.

12.4.2.21 Cohérence de Seq5

Soient $T, T_n, T_c, T_e \in \mathcal{L}$ des termes quelconques. Nous montrerons que si T est un *observateur*, alors

$$\mathrm{Seq}(T, \mathrm{Ex}(T_n, T_c, T_e)) \approx \mathrm{Ex}(\mathrm{Seq}(T, T_n), \mathrm{Seq}(T, T_c), T_e).$$

La preuve est similaire à celle présentée dans la section 12.4.2.19 et ne sera rédigée en détail. ∎

12.4.2.22 Cohérence de DC1

Nous voulons prouver que la relation $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ suivante est une bisimulation–ACTC:

$$\mathcal{R} = \{(\oplus(T_1, T_2), \oplus(T_2, T_1)) \mid T_{1,2} \in \mathcal{L}\} \cup \approx .$$

Soit $(T, S) \in \mathcal{R}$ une paire quelconque telle que

$$\begin{aligned} T &= \oplus(T_1, T_2) \\ S &= \oplus(T_2, T_1)). \end{aligned}$$

Les égalités $\alpha(T) = \alpha(S)$ et $\Gamma(T) = \Gamma(T)$ suivent directement des définitions des fonctions α et Γ. Pour montrer que (T, S) a les propriétés énoncés dans la définition d'une bisimulation forte, il suffit de remarquer que les règles de sémantique opérationnelle de l'opérateur $\oplus$ sont symétriques, ne dépendent pas de l'ordonnancement des sous-termes T_i. Nous pouvons, donc, conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

12.4.2.23 Cohérence de DC3

Montrons que, pour tout $T \in \mathcal{L}$, si T est un observateur, c'est à dire qu'il ne contient aucune action de direction sortie, alors

$$\oplus(T, T) \approx T.$$

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{(T, \oplus(T, T)) \mid T \in \mathcal{L}, T \text{ observateur}\} \cup \approx$$

Nous allons montrer que $\mathcal{R}$ est une bisimulation–ACTC. Soit $(T, S) \in \mathcal{R}$ une paire telle que T soit un observateur et $S = \oplus(T, T)$.

Par définition, $\alpha(S) = \alpha(T)$ et et

$$\Gamma(S) = \Gamma(T) \cup \Gamma(T) = \Gamma(T).$$

Montrons que (T, S) a les propriétés énoncées dans la définition 4.2.

Supposons que T effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$. On va montrer qu'il existe une transition $S \stackrel{ic(v)}{\rightarrow} S'$ telle que $(T', S') \in \mathcal{R}$.

Si $a = ic$ et $T' \neq \uparrow$ on applique la règle $\oplus_1$ et déduit que $S \stackrel{ic(v)}{\rightarrow} \oplus(T', T')$. Donc, par définition, $(T', S') \in \mathcal{R}$.

Si $a = ic$ et $T' = \uparrow$ alors, selon la règle $\oplus_3$, S effectue la transition $S \stackrel{ic(v)}{\rightarrow} \uparrow$. Or, $(T', S') = (\uparrow, \uparrow) \in \approx \subseteq \mathcal{R}$.

Finalement, si $a = \delta$ alors $S \stackrel{\delta(v)}{\rightarrow} \downarrow$ (règle $\oplus_6$) et, comme dans le cas précédent, $(T', S') \in \mathcal{R}$.

La preuve de la propriété symétrique, c'est à dire que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$, est analogue. ∎

12.4.2.24 Cohérence de PCom1

Nous voulons prouver que pour tout terme $[\mathcal{C}, \epsilon](T_1, T_2) \in \mathcal{L}$,

$$[\mathcal{C}, \epsilon](T_1, T_2) \approx [\mathcal{C}, \epsilon](T_2, T_1).$$

La preuve suit directement des définitions et des règles de sémantique opérationnelle de l'opérateur de composition parallèle avec communication. ∎

12.4.2.25 Cohérence de PCom2

Nous voulons prouver que pour tous terme $[\mathcal{C}, \epsilon](T_1, \ldots, T_m) \in \mathcal{L}$ les équivalences suivantes sont vraies:

$$[\mathcal{C}, \epsilon](T_1, \ldots, T_m) \approx [\mathcal{C}, \epsilon]([\emptyset, \cdot](T_1, \ldots, T_{m-1}), T_m) \quad (12.34)$$

$$\approx [\mathcal{C}, \epsilon](T_1, [\emptyset, \cdot](T_2, \ldots, T_m)). \quad (12.35)$$

Nous présentons preuve pour l'équivalence 12.35, celle pour l'équivalence 12.34 étant analogue. Pour ce faire on va montrer que la relation $\mathcal{R} \subset \mathcal{L} \times \mathcal{L}$ définie comme suit:

$$\mathcal{R} = \{(T, S) \mid (\exists([\mathcal{C}, \epsilon](T_1, \ldots, T_m), [\mathcal{C}, \epsilon](T_1, [\emptyset, \cdot](T_2, \ldots, T_m))) \in \mathcal{L} \times \mathcal{L}) \mid (T \approx [\mathcal{C}, \epsilon](T_1, \ldots, T_m) \wedge S \approx [\mathcal{C}, \epsilon](T_1, [\emptyset, \cdot](T_2, \ldots, T_m)))\} \cup \approx$$

est une bisimulation-ACTC. Soit $(T, S) \in \mathcal{R}$ une paire quelconque de $\mathcal{R}$ telle que

$$T = [\mathcal{C}, \epsilon](T_1, \ldots, T_m),$$

$$S = [\mathcal{C}, \epsilon](T_1, [\emptyset, \cdot](T_1, \ldots, T_m)).$$

Notons le sous-terme $[\emptyset, \cdot\,](T_2, \ldots, T_m)$ de S par S_1. On remarque alors que, par la définition de la fonction Act,

$$\mathrm{Act}(S_1) = \bigcup_{i=2}^{m} \mathrm{Act}(T_i),$$

et, par conséquent,

$$\mathrm{Act}(T) = \mathrm{Act}(S).$$

Donc, on peut appliquer la définition de la fonction Γ et déduire que

$$\Gamma(T) = \Gamma(S).$$

De plus, par la proposition 7.2 on a que

$$(\forall i \in \{1, \ldots, m\})(\alpha(T_i) = \alpha(T) = \alpha(S)).$$

Il reste à prouver que $\mathcal{R}$ est une bisimulation forte. Commençons par prouver que pour toute action $a \in A$ et tout $v \in D$

$$a(v) \in \mathrm{exec}(T) \Leftrightarrow a(v) \in \mathrm{exec}(S).$$

Remarquons d'abord que, puisque S_1 ne contient aucune action de communication,

$$a(v) \in \mathrm{exec}(S_1) \Leftrightarrow (\exists i \in \{2, \ldots, m\})(a(v) \in \mathrm{exec}(T_i) \ \wedge \ (\forall j \neq i) Wait(v, T_j)).$$

Il suffit ensuite de remarquer que les ensembles d'actions $\mathrm{Act}(T_i)$ sont, par définition, disjoints, et d'appliquer la proposition 8.6 au termes S_1, S, et T pour déduire que $\mathrm{exec}(T) = \mathrm{exec}(S)$. Il reste à prouver que la relation $\mathcal{R}$ est fermée aux transitions, c'est à dire que pour toute action $a(v) \in \mathrm{exec}(T)$ il existe des transitions $T \stackrel{a(v)}{\rightarrow} T'$ et $S \stackrel{a(v)}{\rightarrow} S'$ telles que $(T', S') \in \mathcal{R}$.

Si $a = \delta$ il n'y a rien à prouver quisque dans ce cas $T' = S' = \downarrow$. Or, par la reflexivité de la relation $\approx$, on déduit que $(T', S') \in \approx \subseteq \mathcal{R}$, ce qu'il fallait prouver.

Supposons alors que $a \neq \delta$. Dans ce cas seule la règle $\mathbf{Com_1}$ puet s'appliquer. Introduisons les notations suivantes, qui correspondent aux règles de SOppour le terme T:

$$I_T(a) = \{i \in \{1, \ldots, m\} \mid \epsilon(a) \cap \mathrm{Act}(T_i) \neq \emptyset\}, \tag{12.36}$$

et pour le terme S_1:

$$I_{S_1}(a) = I_T(a) \setminus \{1\}, \tag{12.37}$$

et finalement pour le terme S:

$$\begin{aligned} I_S(a) &= \{1\}, \text{ si } \epsilon(a) \cap \mathrm{Act}(T_1) \neq \emptyset \ \wedge \ \epsilon(a) \cap \mathrm{Act}(S_1) = \emptyset, && (12.38)\\ &= \{2\}, \text{ si } \epsilon(a) \cap \mathrm{Act}(T_1) = \emptyset \ \wedge \ \epsilon(a) \cap \mathrm{Act}(S_1) \neq \emptyset, && (12.39)\\ &= \{1,2\}, \text{ si } \epsilon(a) \cap \mathrm{Act}(T_1) \neq \emptyset \ \wedge \ \epsilon(a) \cap \mathrm{Act}(S_1) \neq \emptyset, && (12.40) \end{aligned}$$

Puisque selon la définition syntaxique du langage $\mathcal{L}$, les ensembles d'actions $\mathrm{Act}(T_i)$ sont disjoints, et que

$$\mathrm{Act}(S_1) = \bigcup_{i=2}^{m} \mathrm{Act}(T_i),$$

on déduit que

$$I_S(a) = \{1\} \Leftrightarrow I_T(a) \ \{1\} \Leftrightarrow I_{S_1}(a) = \emptyset,$$

et que

$$I_S(a) = \{2\} \Leftrightarrow (1 \notin I_T(a)),$$

et

$$2 \in I_S(a) \Rightarrow I_{S_1}(a) \neq \emptyset.$$

Supposons d'abord que $I_T(a) = I_S(a) = \{1\}$. Dans ce cas a n'est pas une action de communication et T exécute $a(v)$ seulement s'il existe une transition $T_1 \stackrel{a(v)}{\rightarrow} T_1'$. Si $T_1' \neq \uparrow$ alors selon la règle $\mathbf{Com}_1$ le terme T effectue la transition suivante:

$$T \stackrel{a(v)}{\rightarrow} [\mathcal{C}, \epsilon](T_1', \trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m))$$

et le terme S effectue la transition

$$S \stackrel{a(v)}{\rightarrow} [\mathcal{C}, \epsilon](T_1', \trianglelefteq_v(S_1)).$$

Mais nous avons déjà prouvé que l'opérateur $\trianglelefteq$ est distributif par rapport à la composition parallèle $[\ ]$. Donc,

$$\trianglelefteq_v(S_1) \approx [\emptyset, \ \cdot\ ](\trianglelefteq_v(T_2), \ldots, \trianglelefteq_v(T_m)).$$

Donc, par définition de la relation $\mathcal{R}$, $(T', S') \in \mathcal{R}$.

Considérons maintenant le cas où $I_S(a) = \{2\}$, donc $a(v) \in \text{exec}(T)$ mais $a(v) \notin \text{exec}(T_1)$. Dans ce cas les termes T, S, et S_1 efectuent les transitions suivantes:

$$T \stackrel{a(v)}{\rightarrow} [\mathcal{C}', \epsilon']_{\substack{i \in I_T(a) \\ j \in J_T(a)}} (T'_i, \trianglelefteq_v(T_j))$$

et le terme S effectue la transition:

$$S \stackrel{a(v)}{\rightarrow} [\mathcal{C}, \epsilon](\trianglelefteq_v(T_1), S'_1),$$

et S_1 la transition:

$$S_1 \stackrel{a(v)}{\rightarrow} [\emptyset, \ \cdot \ ]_{\substack{i \in I_T(a) \\ j \in J_{S_1}(a)}} (T'_i, \trianglelefteq_v(T_j)),$$

où $J_T(a) = \{1, \ldots, m\} \backslash I_T(a)$, pour tout terme T. On remarque alors que $(T', S') \in \mathcal{R}$, par définition, ce qu'il fallait prouver. ∎

12.4.2.26 Cohérence de PCom3

Soit $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{L}$ la relation suivante:

$$\mathcal{R} = \{([\emptyset, \epsilon](T_1, T_2), \|(T_1, T_2)) \mid T_1, T_2 \in \mathcal{L}_\beta\} \cup \ \approx .$$

Prouvons que $\mathcal{R}$ est une bisimulation–ACTC. Soit $(T, S) \in \mathcal{R}$ une paire quelconque telle que

$$\begin{aligned} T &= [\emptyset, \epsilon](T_1, T_2), \text{ et} \\ S &= \|(T_1, T_2). \end{aligned}$$

Selon la définition de α c'est évident que $\alpha(T) = \alpha(S)$. On remarque que, puisque l'ensemble d'actions de communication de T est vide,

$$\text{Act}(T) = \text{Act}(S) = \text{Act}(T_1) \ \cup \ \text{Act}(T_2)$$

et que $\epsilon(a) = \{a\}$, pour toute action $a \in \text{Act}(T)$. Donc,

$$\begin{aligned} \Gamma([\emptyset, \epsilon](T_1, T_m)) &\stackrel{\text{def}}{=} \Gamma(T_1) \ \cap \Gamma(T_2) \\ &\stackrel{\text{def}}{=} \Gamma(S). \end{aligned}$$

Il reste à prouver que pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$ il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$ telle que $(T', S') \in \mathcal{R}$ et, inversement, que pour toute transition $S \stackrel{a(v)}{\rightarrow} S'$ il existe une transition $T \stackrel{a(v)}{\rightarrow} T'$ telle que $(T', S') \in \mathcal{R}$. Nous ne prouverons que la premère de ces deux implications, la deuxième est similaire.

Remarquons d'abord que les règles $\mathbf{C}om_2$ et $\|_4$, qui définissent les transitions de T et S lorsqu'un des sous-termes exécute $\delta(v)$, sont identiques. De plus, les règles $\mathbf{Com}_3$ et $\mathbf{Com}_4$, qui concernent l'exécution d'actions de communication, ne s'appliquent jamais à T, dont l'ensemble d'actions de communication est vide.

Il reste à considérer la règle $\mathbf{Com}_1$. Soit $a(v)\text{exec}(T)$ une action temporisée quelconque que T peut exécuter. Puisque T ne contient pas d'actions de communications, on déduit qu'il existe un sous-terme unique T_i tel que $a(v) \in \text{exec}(T_i)$ et $I(a) = \{a\}$. Donc, il existe une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et, de plus, $\text{Wait}(v, T_j)$ est vrai pour tout $i \neq i$. Mais ceci correspond aux conditions ennoncées par la règle $\mathbf{Par}_2$. S peut exécuter l'action $a(v)$ et on déduit que

$$a(v) \in Exec(T) \Leftrightarrow a(v) \in \text{exec}(S).$$

Si $T_i' = \uparrow$, alors $T' = S' = \trianglelefteq_v(T_j)$ et $(T', S') \in \approx \subseteq \mathcal{R}$. Sinon, supposons sans perte de généralité que $i = 1$ et $j = 2$. Alors,

$$T' = [\emptyset, \epsilon](T_1', \trianglelefteq_v(T_j)),$$
$$S' = \|(T_1', \trianglelefteq_v(T_j)).$$

Donc, $(T', S') \in \mathcal{R}$. On peut alors conclure que $\mathcal{R}$ est une bisimulation–ACTC. ∎

12.4.2.27 Cohérence de Ex1

L'idempotence restreinte de l'opérateur d'exception est définie comme suit: pour tout observateur $T_n \in \mathcal{L}$ et tout $E \in \mathcal{L}$,

$$\text{Ex}(T_n, T_n, E) \approx T_n.$$

La preuve est similaire que celle de la cohérence de l'axiome d'idempotence du choix retardé **DC3** (section 12.4.2.23). ∎

12.5 Preuves du chapitre 10

12.5.1 Preuve du lemme 10.2

Rappelons l'énoncé du lemme:

Pour tous $T, T' \in \mathcal{L}_\beta, a \in A \setminus \{\delta\}, v \geq V_0$,

$$T \stackrel{a(v)}{\rightarrow} T' \Rightarrow \Gamma(T') \cap [t_a = v] \subseteq \Gamma(T).$$

Preuve: la preuve du lemme a déjà été présentée en partie dans la section 10.2.1. Ici il reste à prouver que pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$,

$$\mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) \subseteq \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')).$$

La base de l'induction sont les termes $a(t_a) \in \mathcal{L}_\beta$, avec $a \neq \delta$, et est triviale. Prouvons le pas d'induction.

Le cas

$$T = a(t_a).S \in \mathcal{L}_\beta.$$

est trivial.

Prouvons le pas d'induction pour un terme

$$T = \|(T_1, \ldots, T_n) \in \mathcal{L}_\beta.$$

Pour toute transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$, un des sous-termes effectue $T_i, 1 \leq i \leq n$, une transition $T_i \stackrel{a(v)}{\rightarrow} T_i'$ et pour tous les autres sous-termes T_j le prédicat $\mathrm{Wait}_v(T_j)$ est vrai. On distingue les deux situations suivantes:

1. $T' = \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{i-1}), T_i', \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_n))$, lorsque $T_i' \neq \uparrow$;
2. $T' = \|(\trianglelefteq_v(T_1), \ldots, \trianglelefteq_v(T_{i-1}), \trianglelefteq_v(T_{i+1}), \ldots, \trianglelefteq_v(T_n))$, lorsque $T_i' = \uparrow$;

Considérons le cas (1). Le cas (2) est analogue. En appliquant l'hypothèse d'induction on déduit que

$$\mathrm{Prj}(\Gamma(T_i'), \mathrm{Act}(T_i')) \subseteq \mathrm{Prj}(\Gamma(T_i), \mathrm{Act}(T_i')). \tag{12.41}$$

De plus, pour tout $j \neq i, 1 \leq j \leq n$ on a que

$$\Gamma^{\geq v}(T_j) \subseteq \Gamma(T_j). \tag{12.42}$$

Or, par définition, les ensembles d'actions des sous-termes de T et de T' sont disjoints. On applique alors la proposition 5.1 pour déduire que

$$\begin{aligned}
\mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) &= \bigcap_{j=1, j\neq i}^{j=n} \mathrm{Prj}(\Gamma^{\geq v}(T_j, \mathrm{Act}(T_j)) \cap \mathrm{Prj}(\Gamma(T_i', \mathrm{Act}(T_i')) \\
&\overset{12.41}{\subseteq} \bigcap_{j=1, j\neq i}^{j=n} (\mathrm{Prj}(\Gamma(T_j), \mathrm{Act}(T_j)) \cap \mathrm{Prj}(\Gamma(T_i), \mathrm{Act}(T_i')) \\
&\overset{\text{def}}{=} \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')),
\end{aligned}$$

ce qu'il fallait prouver.

Soit

$$T = \lhd_v(S) \in \mathcal{L}_\beta.$$

Nous ne traitons que l'opérateur de décalage stricte $\lhd_v$; la preuve pour l'opérateur $\trianglelefteq_v$ est analogue. T effectue une transition quelconque avec $a \neq \delta$

$$T = \lhd_v(T_0) \overset{a(v')}{\rightarrow} T'$$

si et seulement si T_0 exécute $a(v')$, transite ensuite vers T' et $v' > v$. Selon l'hypothèse d'induction

$$\mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) \subseteq \mathrm{Prj}(\Gamma(T_0), \mathrm{Act}(T')).$$

Or, selon la proposition 7.3, $\alpha(T') = v'$. De plus, selon la cohérence de l'axiome **Dec1** on a que $T' \approx \trianglelefteq_{\alpha(T')}(T')$. Donc, par la définition de $\approx$ on déduit que

$$\Gamma(T') = \Gamma^{\geq v'}(T').$$

De plus, $v' \geq v$, donc

$$\Gamma^{\geq v}(T') = \Gamma(T')$$

On déduit alors que

$$\begin{aligned} \mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) &= \mathrm{Prj}(\Gamma^{\geq v}(T'), \mathrm{Act}(T')) \\ &\subseteq \mathrm{Prj}(\Gamma^{\geq v}(T_0), \mathrm{Act}(T')) \\ &\stackrel{\text{def}}{=} \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')), \end{aligned}$$

ce qu'il fallait prouver.

Soit

$$T = \theta : S \in \mathcal{L}_\beta,$$

avec $\theta \in \Theta$ et $S \in \mathcal{L}_\beta$. T peut exécuter une action $a(v)$ différente de δ si et seulement si S peut effectuer une transition

$$S \stackrel{a(v)}{\rightarrow} S'$$

et le prédicat $\mathrm{First}(a(v), T)$ est vrai. Il y a deux situations possibles:

1. $T' = \theta_{v/t_a} : S'$, lorsque $S' \neq \uparrow$.

2. $T' = S' = \uparrow$, et

Prouvons la proposition dans la situation (1); la situation (2) est triviale. On a que

$$\Gamma(T') = \mathrm{Sol}(\theta_{v/t_a}) \cap \Gamma(S')$$

et

$$\Gamma(T) = \mathrm{Sol}(\theta) \cap \Gamma(S).$$

Par l'hypothèse d'induction

$$\mathrm{Prj}(\Gamma(S'), \mathrm{Act}(S')) \subseteq \mathrm{Prj}(\Gamma(S), \mathrm{Act}(S)).$$

Or, par définition et selon la proposition 6.1

$$\begin{aligned} \mathrm{Act}(T) &= \mathrm{Act}(S) \\ \mathrm{Act}(T') &= \mathrm{Act}(S') \\ \mathrm{Act}(T) &= \mathrm{Act}(S') \cup \{a\}. \end{aligned}$$

Nous pouvons, donc, déduire que l'inclusion

$$\mathrm{Prj}(\Gamma(T'), \mathrm{Act}(T')) \subseteq \mathrm{Prj}(\Gamma(T), \mathrm{Act}(T')),$$

ce qu'il fallait prouver.

Il reste à considérer les opérateurs réactifs Max et Min. Nous allons seulement prouver que la proposition est vérifiée par l'opérateur Max; la preuve pour Min est analogue.

Soit

$$T = \mathrm{Max}(o, H, [m, M], S) \in \mathcal{L}_\beta$$

un terme qui effectue une transition $T \stackrel{a(v)}{\rightarrow} T'$, avec $a \neq \delta$. Ceci implique qu'il existe une transition $S \stackrel{a(v)}{\rightarrow} S'$ et on distingue les trois cas suivants:

1. $|H \setminus \{a\}| \geq 1$ et alors $T' = \mathrm{Max}(o, \setminus\{a\}, [m, M], S'$,
2. $H = \{a\}$ et $T' = m \leq t_o - v \leq M : S'$, et
3. $T' = S' = \uparrow$.

Dans le premier cas il est évident que $\Gamma(T) = \Gamma(S)$ et $\Gamma(T') = \Gamma(S')$. Il suffit alors d'appliquer l'hypothèse d'induction aux termes S et S' pour conclure que la proposition est vérifiée par T et T'.

La suite d'inclusions suivante prouve que la proposition est vérifiée dans le cas (2) aussi:

$$\begin{aligned} \Gamma(T') \cap \mathrm{Sol}(t_a = v) &= \Gamma(S') \cap \mathrm{Sol}(m \leq t_o - v \leq M) \cap \mathrm{Sol}(t_a = v) \\ &\subseteq \Gamma(S')\mathrm{Sol}(t_a = v) \end{aligned}$$

$$\begin{array}{ll} & \overset{\text{hyp. d'ind.}}{\subseteq} \quad \mathrm{Sol}(\Gamma(S)) \\ & = \quad \mathrm{Sol}(\Gamma(T)). \end{array}$$

Ceci conclut la preuve de la proposition. ∎

12.5.2 Preuve de la proposition 10.4

Rappelons l'énoncé de la proposition:

Soit $T \in \mathcal{L}_\beta$ un terme qui ne contient aucun opérateur réactif. Soit

$$\vec{v} = (v_1, \ldots, v_n) \in \Gamma(T)$$

tel que $\mathrm{Act}(\vec{v}) = \mathrm{Act}(T)$ *un vecteur d'instants temporels qui satisfait aux contraintes temporelles de T. Soit $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ une permutation des indices i telle que les valeurs temporelles permutées de $\vec{v}$ sont en ordre croissant:*

$$v_{\sigma(i)} \leq v_{\sigma(i+1)},$$

pour tout $i : 1 \leq i \leq n-1$. Notons l'action correspondante à la valeur temporelle $v_{\sigma(i)}$ par $a_i \in \mathrm{Act}(T)$. Alors il existe une trace $\mu \in \mathbb{T}(T)$ telle que

$$\mu = a_{\sigma(1)}(v_{\sigma(1)}) \ldots a_{\sigma(n)}(v_{\sigma(n)}).$$

Preuve:

La preuve sera faite par induction structurelle. La base de l'induction est donnée par les termes $a(t_a)$, avec $a \in A \setminus \{\delta\}$, qui satisfont la proposition trivialement. Prouvons le pas d'induction.

Supposons d'abord que

$$T = \lhd_v(T_1),$$

Ce cas est trivial. Il suffit de remarquer que, selon les règles SOp,

$$\mathbb{T}(T) = \{\mu = a_1(v_1)...a_n(v_n) \mid \mu \in \mathbb{T}(T_1) \wedge \ v_1 > v\}$$

et que, par définition,

$$\Gamma(T) = \Gamma^{>v}(T_1).$$

Supposons maintenant que

$$T = a(t_a).T_1,$$

avec $a \in \mathcal{L} \setminus \delta$ et $T_1 \in \mathcal{L}_\beta$ et soit

$$\vec{v} = (v_0, v_1 \ldots, v_n) \in \Gamma(T),$$

tel que $\mathrm{Act}(T) = \vec{v}$. Notons par

$$\sigma(\vec{v}) = (v_{\sigma(0)}, \ldots, v_{\sigma(n)})$$

la permutation monotone croissante du vecteur $\vec{v}$. Puisque

$$\Gamma(T) = \mathrm{Sol}(\bigwedge_{b \in \mathrm{Act}(T_1)} t_b < t_a) \cap \Gamma(T_1),$$

alors $\sigma(\vec{v})$ doit affecter une valeur quelconque $v_{\sigma(0)} \geq V_0$ à la variable temporelle t_a et

$$(v_{\sigma(1)}, \ldots, v_{\sigma(n)}) \in \Gamma(T_1).$$

Donc, il existe une transition

$$T \overset{a(v_{\sigma(0)})}{\rightarrow} \trianglelefteq_{v_{\sigma(0)}}(T_1).$$

De plus, selon l'hypothèse d'induction, il existe une trace

$$\mu_1 = b_1(v_{\sigma(1)}) \ldots b_n(v_{\sigma(n)}) \in \mathbb{T}(T_1),$$

avec $v_{\sigma(1)} \geq v_{\sigma(0)}$. Ceci implique que

$$\mu_1 \in \mathbb{T}(\trianglelefteq_{v_{\sigma(0)}}(T_1)).$$

On déduit alors que

$$\mu = a(v_{\sigma(0)}).\mu_1 \in \mathbb{T}(T),$$

ce qu'il fallait montrer.

Soit

$$T = \theta : T_1$$

avec $\theta \in \Theta$ et $T_1 \in \mathcal{L}_\beta$ un terme feuille qui ne contient aucun opérateur réactif. Soit

$$\vec{v} = (v_1, \ldots, v_n)$$

un vecteur quelconque tel que

$$\text{Act}(\vec{v}) = \text{Act}(T) \text{ et } \vec{v} \in \Gamma(T).$$

Soit $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ une permutation telle que

$$v_i \leq v_{i+1}, \forall i \in \{1, \ldots, n\}$$

et notons par $a_{\sigma(i)}$ l'action correspondantes à la valeur $v_{\sigma(i)}$. Notons par

$$\mu(\sigma, \vec{v}) = a_{\sigma(1)}(v_{\sigma(1)}) \ldots a_{\sigma(n)}(v_{\sigma(n)})$$

la trace d'actions correspondante au vecteur $\vec{v}$ permuté. La preuve du pas d'induction structurelle se fait par induction sur n, la longueur du vecteur $\vec{v}$.

On remarque d'abord que

$$\vec{v} \in \Gamma(T) \subseteq \Gamma(T_1).$$

De plus,

$$\vec{v} \in \Gamma(T) \Rightarrow \text{First}(a_{\sigma(1)}(v_{\sigma(1)}), T). \tag{12.43}$$

Donc, puisque $\Gamma(T) = \text{Sol}(\theta) \wedge \Gamma(T_1)$, on déduit que le prédicat $\text{First}(a_{\sigma(1)}(v_{\sigma(1)}), T_1)$ est vrai aussi

$$\vec{v} \in \Gamma(T) \Rightarrow \text{First}(a_1(v_{\sigma(1)}), T_1). \tag{12.44}$$

Supposons d'abord que $n = 1$, donc $\text{Act}(T) = \text{Act}(T_1) = \{a_{\sigma(1)}\}$. Alors, selon l'hypothèse d'induction structurelle on déduit que

$$\mu \in \mathbb{T}(T_1).$$

Puisque $\text{Act}(T_1) = \{a_{\sigma(1)}\}$, ceci implique que

$$T_1 \overset{a_{\sigma(1)}(v_{\sigma(1)})}{\rightarrow} \uparrow .$$

Donc, en tenant compte de l'équation 12.43 on déduit que

$$T \overset{a_{\sigma(1)}(v_{\sigma(1)})}{\rightarrow} \uparrow ,$$

ce qu'il fallait prouver.

Prouvons le pas d'induction $n \Rightarrow n+1$. Le pas d'induction consiste à prouver que si la proposition est vrai pour $n = k$, où $k \geq 1$ est un entier quelconque, alors elle est vrai pour $n = k + 1$. Soit

$$\vec{v} = \langle v_1 \dots v_n \rangle \in \Gamma(T)$$

un vecteur quelconque tel que $\text{Act}(\vec{v}) = \text{Act}(T)$. Si $\vec{v}$ est une solution de $\Gamma(T)$ et de $\Gamma(T_1)$ alors, selon l'hypothèse d'induction structurelle, il existe alors une trace $\mu = a_{\sigma(1)}(v_{\sigma(1)}) \dots a_{\sigma(n)}(v_{\sigma(n)})$, telle que

$$T_1 \overset{a_{\sigma(1)}(v_{\sigma(1)})}{\rightarrow} T_2 \overset{a_{\sigma(2)}(v_{\sigma(2)})}{\rightarrow} T_3 \dots \overset{a_{\sigma(n)}(v_{\sigma(n)})}{\rightarrow} \uparrow .$$

On va prouver que T aussi peut exécuter la trace μ. Puisque le prédicat $\text{First}(a_{\sigma(1)}(v_{\sigma(1)}), T)$ est vrai (12.43) le terme $T = \theta : T_1$ peut effectuer la transition

$$T \overset{a_{\sigma(1)}(v_{\sigma(1)})}{\rightarrow} \theta_{v_{\sigma(1)}/t_{a_{\sigma(1)}}} : T_2$$

où T_2 est le successeur de T_1 après l'exécution de $a_{\sigma(1)}(v_{\sigma(1)})$. Alors le sous-vecteur

$$\vec{v_1} = \langle a_{\sigma(2)}(v_{\sigma(2)}) \dots a_{\sigma(n)}(v_{\sigma(n)}) \rangle$$

est une solution de la contrainte $\Gamma(\theta_{v_{\sigma(1)}/t_{a_{\sigma(1)}}} : T_2)$. Selon l'hypothèse d'induction sur le nombre d'actions n, il existe alors une trace

$$\mu' = a'_{\sigma(2)}(v_{\sigma(2)}) \dots a'_{\sigma(n)}(v_{\sigma(n)}) \in \mathbb{T}(\theta_{v_{\sigma(1)}/t_{a_{\sigma(1)}}} : T_1)$$

telle que

$$\theta_{v_{\sigma(1)}/t_{a_{\sigma(1)}}} : T_2 \overset{\mu'}{\rightarrow} \uparrow .$$

Alors la trace concaténée $\mu'' = a_1(v_1).\mu'$ est une trace de T, telle que

$$T \xrightarrow{\mu''} \uparrow ,$$

ce qui complète la preuve par induction sur n du pas d'induction structurelle pour le cas $T = \theta : T_1$.

Finalement, il reste à considérer le cas où

$$T = \|(T_1, \ldots, T_m),$$

avec $T_1, \ldots, T_m$ des termes feuilles quelconques qui ne contiennent aucun opérateur réactif. Pour simplifier, supposons que $m = 2$; le cas générale où T contient un nombre arbitraire $m \geq 2$ de sous-termes est analogue.

Similairement au cas précédent, nous faisons une induction sur le nombre n d'actions du terme T. La base de l'induction est le cas où T ne contient que deux actions, une par sous-terme. Soient

$$\mathrm{Act}(T_1) = \{a_1\} \text{ et } \mathrm{Act}(T_2) = \{a_2\}.$$

Soit

$$\vec{V} = \langle v_1 = t_{a_1}, v_2 = t_{a_2} \rangle \in \Gamma(T) \stackrel{\text{def}}{=} \Gamma(T_1) \cap \Gamma(T_2)$$

un vecteur quelconque qui satisfait la contrainte $\Gamma(T)$ et, par conséquent, les contraintes $\Gamma(T_1)$ et $\Gamma(T_2)$ aussi. Alors, selon l'hypothèse d'induction structurelle, il existe les transitions

$$\begin{array}{ll} T_1 & \xrightarrow{a_1(v_1)} \uparrow \\ T_2 & \xrightarrow{a_2(v_2)} \uparrow . \end{array}$$

Supposons, sans perte de généralité que $v_1 \leq v_2$. Alors, selon les règles SOp, il existe les transitions

$$\begin{array}{lll} T & \xrightarrow{a_1(v_1)} & \trianglelefteq_{v_1}(T_2) \\ & \xrightarrow{a_2(v_2)} & \uparrow . \end{array}$$

Donc, la trace $\mu = (a_1(v_1), a_2(v_2))$ est une trace de T et est telle que $T \xrightarrow{\mu} \uparrow$, ce qu'il fallait montrer.

Prouvons maintenant le pas d'induction $n-1 \Rightarrow n$, c'est à dire que si la proposition est vrai pour tout terme $T = \|(T_1, T_2)$ avec k actions, où $2 \leq k \leq n-1$, elle est vrai aussi pour tout terme $T = \|(T_1, T_2)$ avec n actions. Supposons que

$$\begin{aligned} \mathrm{Act}(T_1) &= \{a_1^1, \ldots, a_{n_1}^1\}, \\ \mathrm{Act}(T_2) &= \{a_1^2, \ldots, a_{n_2}^2\}, \end{aligned}$$

avec $n_1, n_2 \geq 1$ et $n_1 + n_2 = n$. Soit $\vec{v}$ une solution de la contrainte $\Gamma(T)$. Soient $\vec{v}_1$ et $\vec{v}_2$ les projections de $\vec{v}$ sur les variables temporelles de T_1, respectivement de T_2,

$$\vec{v}_1 = \langle v_1^1, \ldots, v_{n_1}^1) \rangle, \text{ respectivement}$$
$$\vec{v}_2 = \langle v_1^2, \ldots, v_{n_2}^2 \rangle.$$

Alors, on remarque que pour tout terme feuille $T \in \mathcal{L}_\beta$,

$$\mathrm{Act}(T) = \mathrm{Act}(\Gamma(T)). \tag{12.45}$$

Donc, puisque $\Gamma(T) = \Gamma(T_1) \wedge \Gamma(T_2)$, on déduit que

$$\vec{v}_1 \in \Gamma(T_1) \text{ et } \vec{v}_2 \in \Gamma(T_2).$$

Selon l'hypothèse d'induction structurelle il existe alors des traces

$$\begin{aligned} \mu_1 &= (a_1^1(v_1^1), \ldots, a_{n_1}^1(v_{n_1}^1)) \\ \mu_2 &= (a_1^2(v_1^2), \ldots, a_{n_2}^2(v_{n_2}^2)) \end{aligned}$$

et les transitions

$$\begin{aligned} T_1 &\xrightarrow{\mu_1} \uparrow \\ T_2 &\xrightarrow{\mu_2} \uparrow . \end{aligned}$$

Supposons, sans perte de généralité que $v_1^1 \leq v_1^2$. Alors le prédicat $\text{Wait}(v_1^1, T_2)$ est vrai et T peut effectuer la transition

$$T \overset{a_1^1(v_1^1)}{\rightarrow} T'.$$

On distingue deux cas: soit (1) $n_1 = 1$, ou bien (2) $n_1 \geq 2$.

Dans le premier cas on a que

$$T' = \trianglelefteq_{v_1^1}(T_2).$$

Donc, $\vec{v}_2 \in \Gamma(T')$ et selon l'hypothèse d'induction structurelle, il existe la transition $T' \overset{\mu_2}{\rightarrow} \uparrow$. Donc, T peut effectuer la transition

$$T \overset{(a_1^1(v_1^1).\mu_2)}{\rightarrow} \uparrow$$

ce qu'il fallait montrer.

Si $n_1 \geq 2$ on a que

$$T' = \|(T_1', \trianglelefteq_v(T_2),$$

où T_1' est le successeur de T_1 après l'exécution de $a_1^1(v_1^1)$. On déduit que (Proposition 6.1)

$$\text{Act}(T') = \text{Act}(T) \setminus \{a_1^1\} \text{ et } |\text{Act}(T') = n - 1|.$$

En plus, selon le lemme 10.2

$$\Gamma(T_1') \cap [t_{a_1} = v_1] \subseteq \Gamma(T_1).$$

Or, selon la proposition 7.3, $\alpha(T_1') = v_1$. De plus, selon la cohérence du l'axiome **Dec1** on a que $T_1' \approx \trianglelefteq_{\alpha(T_1')}(T_1')$. Donc, par la définition de $\approx$ on déduit que

$$\Gamma(T_1') = \Gamma^{\geq v_1}.$$

On remarque alors que

$$\vec{v}_1' = (v_2^1, \ldots v_{n_1}^1) \in \Gamma(T_1').$$

De plus,

$$\vec{v}_2 \in \Gamma(\trianglelefteq_{v_1^1}(T_2))$$

parce que $v_1^1 \leq v_i^2$, pour tout $1 \leq i \leq n_2$. Étant donné que les ensembles d'actions de T_1' et $\trianglelefteq_v(T_2)$ sont disjoints, on peut déduire que la concaténation (en ordre croissant des valeurs v_i^j) de $\vec{v}_1'$ et $\vec{v}_2$, noté $\vec{v}'$, est une solution de $\Gamma(T')$. Or, T' a $n-1$ actions (Proposition 6.1). Donc, selon l'hypothèse d'induction sur le nombre d'actions de T, on conclut qu'il existe une trace μ', telle que

$$T' \xrightarrow{\mu'} \uparrow$$

et telle que les temps d'occurrence des actions des μ' correspondent aux valeurs temporelles de $\vec{v}'$. Finalement, ceci implique que

$$T \xrightarrow{a_1^1(v_1^1).\mu'} \uparrow,$$

ce qu'il fallait prouver. ∎

12.5.3 Preuve du lemme 10.7

On va prouver que pour tout $m \geq 2$ et tous $a_1, \ldots, a_m \in \mathrm{A}$

$$a_1(t_{a_1}) \ldots a_m(t_{a_m}) \;\approx\; \bigwedge_{i=1}^{i=m-1} t_i < t_{i+1} : \|(a_1, a_2, \ldots, a_m).$$

La preuve sera faite par induction sur m. Comme base d'induction on va prouver que

$$a_1.a_2 \;\approx\; (t_{a_1} < t_{a_2}) : \|(a_1, a_2).$$

Le pas d'induction consiste à prouver que si la proposition est vraie pour $m = k-1$, où $k > 2$, alors elle est vrai aussi pour $m = k$. Soit $\mathcal{R}_k \subseteq \mathcal{L}_\beta \times \mathcal{L}_\beta$ la relation suivante:

$$\begin{aligned}\mathcal{R}_k = \{ \quad & (\quad a_1.a_2 \ldots a_k, \bigwedge_{i=1}^{i=k-1} t_{a_i} < t_{a_{i+1}} : \|(a_1, a_2, \ldots, a_k) \\ & \mid \quad k \geq 2, a_i \in A \setminus \{\delta\}, 1 \leq i \leq k\} \cup \;\approx .\end{aligned}$$

Nous allons prouver que $\mathcal{R}_k$ est une bisimulation ACTC. Étant données les actions $a(1), \ldots, a(k)$ nous introduisons les notations suivantes pour $j = 1, 2$:

$$\begin{aligned} T^j &= a_j.a_{j+1} \ldots a_k. \uparrow \\ \theta^j &= \bigwedge_{i=j}^{i=k-1} t_{a_i} < t_{a_{i+1}} \\ P^j &= \|(a_j, a_{j+1}, \ldots, a_k) \\ S^j &= \theta^j : P^j. \end{aligned}$$

Les seules transitions que T^1 peut effectuer sont $T^1 \stackrel{a_1(v)}{\rightarrow} T'$ où $T' = \trianglelefteq_v(T^2)$, pour tout $v \geq V_0$. Similairement, les seules transitions possibles pour S^1 sont $S^1 \stackrel{a_1(v)}{\rightarrow} S'$, où $v \geq V_0$ et

$$S' = \theta^1_{v/t_{a_1}} : \|(\trianglelefteq_v(a_2), \ldots, \trianglelefteq_v(a_k)).$$

Selon la distributivité de $\trianglelefteq$ par rapport à $\|$ (Proposition 12.4.2.3) et étant donnée que $\approx$ est une congruence par rapport à l'opérateur θ (Théorème 8.10),

$$S' \approx \theta^1_{v/t_{a_1}} : \trianglelefteq_v(P^2).$$

Or,

$$\begin{aligned} \theta^1_{v/t_{a_1}} &= \theta^2 \wedge t_{a_2} \geq v \\ &= \theta^2 \bigwedge_{i=2}^{k} t_{a_i} \geq v. \end{aligned}$$

Alors,

$$\begin{aligned} S' \quad \approx \quad & \theta^2 : \bigwedge_{i=2}^{k} t_{a_i} \geq v : \trianglelefteq_v(P^2) \\ \stackrel{\textbf{Cons5},th.9.6}{\approx} \quad & \theta^2 : \trianglelefteq_v(P^2) \\ \stackrel{\textbf{Dec3},th.9.6}{\approx} \quad & \trianglelefteq_v(\theta^2 : P^2) \\ = \quad & \trianglelefteq_v(S_2). \end{aligned}$$

Selon l'hypothèse d'induction, $R_2 \approx S_2$. Donc, puisque $\approx$ est une congruence par rapport à $\trianglelefteq$,

$$T' = \trianglelefteq_v(R_2) \approx \trianglelefteq_v(S_2) = S'.$$

Donc, $(T', S') \in \approx \subseteq \mathcal{R}$. ∎

12.5.4 Preuve du lemme 10.10

Rappelons l'énoncé du lemme:

Soit $\mathrm{ROp}(o, H, [m, M], T)$ *un terme réactif de* $\mathcal{L}_\beta$, *où* $\mathrm{ROp} \in \{\mathrm{Max}, \mathrm{Min}\}$, $o \in O$, $T \in \mathcal{L}_\beta$, *et* $[m, M]$ *est l'intervalle réactif de* ROp. *Alors,*

$$\mathrm{ROp}(o, H, [m, M], T) \sim \sum_{a \in H} \theta_{(a,\mathrm{ROp})} : T,$$

où nous notons par $\theta_{(a,\mathrm{ROp})}$ *la contrainte temporelle suivante:*

$$\theta_{(a,\mathrm{Max})} \stackrel{\text{def}}{=} \bigwedge_{b \in H} t_a \geq t_b \ \wedge \ m \leq t_o - t_a \leq M,$$

$$\theta_{(a,\mathrm{Min})} \stackrel{\text{def}}{=} \bigwedge_{b \in H} t_a \leq t_b \wedge \ m \leq t_o - t_a \leq M.$$

Preuve: Considérons juste le cas où ROp = Min. Le cas ROp = Max est analogue. Nous prouverons que

$$\mathrm{Min}(o, H, [m, M], T) \sim \sum_{a \in H} \theta_{(a,\mathrm{Min})} : T,$$

pour tout terme réactif $\mathrm{Min}(o, H, [m, M], T) \in \mathcal{L}^r_\beta$. Pour ce faire nous appliquons le théroème 4.8, selon lequel

$$\begin{aligned} \mathrm{Min}(o, H, [m, M], T) &\sim \sum_{a \in H} \theta_{(a,\mathrm{Min})} : T \Leftrightarrow \\ \{\mathrm{Min}(o, H, [m, M], T)\} &\simeq \{\sum_{a \in H} \theta_{(a,\mathrm{Min})} : T\} \end{aligned}$$

Soit $\mathcal{R} \subseteq 2^{\mathcal{L}} \times 2^{\mathcal{L}}$ la relation suivante:

$$\begin{aligned} \mathcal{R} = \{(\{T\}, \{S\}) \mid T &= \mathrm{Min}(o, H, [m, M], T_0), \\ \{S\} &\approx \{\sum_{a \in H} \theta_{(a,T)} : T_0\}, T_0 \in \mathcal{L}_\beta\} \cup \simeq \ . \end{aligned}$$

Nous montrerons que $\mathcal{R}$ est une bisimulation forte, donc que

$$\{\mathrm{Min}(o, H, [m, M], T)\} \simeq \{\sum_{a \in H} \theta_{(a,\mathrm{Min})} : T\},$$

pour tout $\mathrm{Min}(o, H, [m, M], T) \in \mathcal{L}^r_\beta$.

Selon le raisonnement présenté dans la section 8.2, il suffit de considérer les paires $(\{T\}, \{S\}) \in \mathcal{R}$ telles que

$$T = \mathrm{Min}(o, H, [m, M], T_0), S = \sum_{a \in H} \theta_{(a,T)} : T_0$$

et de prouver que

1. pour toute transition $\{T\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{T}$ il existe une transition $\{S\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{S}$ telle que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$;
2. pour toute transition $\{S\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{S}$ il existe une transition $\{T\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{T}$ telle que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$.

Prouvons l'implication 1 d'abord.

La relation de transition $\rightarrow$ est fermée dans $\mathcal{L}_\beta$ (proposition 7.1). Donc, puisque $T \in \mathcal{L}_\beta$, tout terme accessible à partir de T est un terme feuille. Selon le théorème 6.4, ACTC_β est déterministe. Donc, pour tout terme $T \in \mathcal{L}_\beta$ et toute action $a_0(v_0) \in \mathrm{exec}(T)$ il existe un terme unique T' tel que

$$T \stackrel{a_0(v_0)}{\rightarrow} T'$$

Donc, dans les système à transitions par parties associé à T, ceci implique que tous les sous-ensembles de termes accessibles à partir de $\{T\}$ contiennent une unique terme $T' \in \mathcal{L}_\beta$. Donc, pour toute trace μ on a que

$$T \stackrel{\mu}{\rightarrow} T' \Leftrightarrow \{T\} \stackrel{\mu}{\rightarrow} \{T'\}.$$

Puisque T est réactif, selon le théorème 7.14, aucune trace de T ne mène vers un blocage et il existe au moins une trace

$$\mu = a_0(v_0).a_1(v_1) \ldots a_n(v_n) \in \mathbb{T}(T)$$

telle que $\{a_0, \ldots, a_n\} = \mathrm{Act}(\mu) = \mathrm{Act}(T)$. Il existe alors une séquence de transitions

$$T \stackrel{a_0(v_0)}{\rightarrow} T_1 \stackrel{a_1(v_1)}{\rightarrow} \ldots T_{n-1} \stackrel{a_n(v_n)}{\rightarrow} \uparrow .$$

Selon la sémantique opérationnelle des opérateurs Min on a que

$$\mathbb{T}(T) \subseteq \mathbb{T}(T_0),$$

parce que T et, par conséquent T_0 aussi, sont réactifs. De plus, puisque $\mathrm{Act}(T) = \mathrm{Act}(T_0)$, on déduit que

$$\mu \in \mathbb{T}(T_0) \text{ et } T_0 \overset{a_0(v_0}{\rightarrow} T_1' \overset{a_1(v_1)}{\rightarrow} \ldots T_{n-1}' \overset{a_n(v_n)}{\rightarrow} \uparrow \ .$$

Soit $a_j \in H$ la première action source de la trace μ et notons les sous-traces de μ qui précèdent, respectivement suivent, l'action a_j par μ_0 et μ_1, respectivement:

$$\mu = \mu_0.a_j(v_j).\mu_1$$

Donc,

$$T \overset{\mu_0}{\rightarrow} T_{j-1} \overset{a_j(v_j)}{\rightarrow} m \leq t_o - v_j \leq M : T_j' \overset{\mu_1}{\rightarrow} \uparrow ,$$

Notons par

$$\vec{w} = (v_{j+1}, \ldots, v_n).$$

Alors, selon la proposition 10.4,

$$\begin{aligned} \vec{w} &= \mathrm{Sol}(\bigwedge_{i=j+1}^{n} t_{a_i} = v_i) \\ &\subseteq \Gamma(m \leq t_o - v_j \leq M : T_j') \\ &\overset{\text{def}}{=} \Gamma(T_j') \cap \mathrm{Sol}(m \leq t_o - v_j \leq M). \end{aligned}$$

Mais, selon la même proposition,

$$\begin{aligned} \Gamma(T_j') &\subseteq \Gamma(T_0) \cap \mathrm{Sol}(\bigwedge_{i=0}^{i=j} t_{a_i} = v_j) \\ &\overset{\text{def.}}{\subseteq} \Gamma(T_0) \cap \mathrm{Sol}(\bigwedge_{i=0}^{n} t_{a_0} \leq t_{a_i}). \end{aligned}$$

Donc,

$$\vec{w} \in \Gamma(T_0) \cap \mathrm{Sol}(m \leq t_o - v_j \leq M) = \Gamma(S).$$

De plus $\vec{w}$ est une validation pour le prédicat $\text{First}(a_0(v_0), \theta_{(a_j,T)} : T_0)$. Donc, il existe une transition du sous-terme de S

$$\theta_{(a_j,T)} : T_0 \stackrel{a_0(v_0)}{\rightarrow} S'.$$

Selon la sémantique de l'opérateur $+$, S aussi peut effectuer la transition

$$S \stackrel{a_0(v_0)}{\rightarrow} S'$$

et, par conséquent, une transition

$$\{S\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{S}$$

avec $S' \in \mathcal{S}$. Pour prouver la propriété 1, alors, il reste à montrer que $\mathcal{T} = \{T'\}$ et $\mathcal{S}$ sont tels que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$.

Notons par $H_0 \subseteq H$ le sous-ensemble d'actions sources $a \in H$ telles que la contrainte $\theta_{(a,T)}$ ne contredit pas l'exécution de a_0:

$$H_0 = \{a \in H \mid \text{First}(a_0(v_0), \theta_{(a,T)} : T_0)\}.$$

Alors la transition $\{S\} \stackrel{a_0(v_0)}{\rightarrow} \mathcal{S}$, est telle que

$$\mathcal{S} = \{\theta_{(a,T),(v_0/t_{a_0})} : T_0' \mid a \in H_0\}.$$

Il y a trois situations à considérer:

1. $j = 0$ (donc $a_0 \in H$) et $\uparrow \neq \mathcal{T}'$,
2. $j > 0$ (donc $a_0 \notin H$) et $\uparrow \neq T'$, et
3. $\uparrow = T'$.

Dans le premier cas

$$T' = \text{Min}(o, H, [m, M], T_1').$$

Par conséquent, nous devons prouver que

$$\mathcal{S} \simeq \sum_{a \in H} \theta_{(a,T')} : T_1'.$$

Or,

$$\theta_{(a,T),(v_0/t_{a_0})} = \theta_{(a,T')} \wedge v_0 \leq t_a.$$

Donc,

$$\mathcal{S} = \{\theta_{(a,T')} \wedge v_0 \leq t_a : T_1' \mid a \in H_0\}$$

Selon la proposition 10.9,

$$\mathcal{S} \simeq \sum_{a \in H_0} (\theta_{(a,T')} \wedge v_0 \leq t_a : T_1').$$

Étant donné que $\alpha(T_1') = v_0$ on a que $v_0 \leq t_a : T_1' \approx T_1'$. Donc, parce que $\approx$ est une congruence par rapport aux opérateurs θ et $\sum$ (théorème 8.10),

$$\sum_{a \in H_0} (\theta_{(a,T')} \wedge v_0 \leq t_a : T_1') \approx \sum_{a \in H_0} (\theta_{(a,T')} : T_0').$$

Mais, par définition, $\approx \subseteq \simeq$ et $\simeq$ est transitive. Donc,

$$\mathcal{S} \simeq \sum_{a \in H_0} (\theta_{(a,T')} : T_1').$$

Pour montrer que

$$\mathcal{S} \simeq \sum_{a \in H} (\theta_{(a,T')} : T_1')$$

il reste alors à prouver que tous les sous-termes

$$[Source(T') = a'] : T_1' \text{ avec } a' \in H \setminus H_0$$

sont non-viables, c'est à dire qu'ils exécutent l'action de blocage $\delta(v_0)$. Or, pour toute $a' \in H' \setminus H_0$ le prédicat $\text{First}(a_0(v_0), \theta_{(a',T)} : T_0)$ est faux. Donc, par définition,

$$\begin{aligned}
&(\Gamma(\theta_{(a',T)} : T_0) \cap \text{Sol}(\textstyle\bigwedge_{b \in \text{Act}(T)} t_{a_0} \leq t_b)_{v_0/t_{a_0}}) \\
&= \text{Sol}(\theta_{(a',T),(v_0/t_{a_0})}) \cap \Gamma(T)_{v_0/t_{a_0}} \cap \text{Sol}(\textstyle\bigwedge_{b \in \text{Act}(T)} v_0 \leq t_b) \\
&= \text{Sol}(\theta_{(a',T')}) \cap \Gamma(T)_{v_0/t_{a_0}} \cap \text{Sol}(\textstyle\bigwedge_{b \in \text{Act}(T)} v_0 \leq t_b) \\
&= \emptyset.
\end{aligned}$$

Mais $\Gamma(T) = \Gamma(T_0)$, $\text{Act}(T) = \text{Act}(T_0)$ et selon le lemme 10.2,

$$\Gamma(T_0') \cap \text{Sol}(t_{a_0} = v_0) \subseteq \text{Sol}(\Gamma(T_0)).$$

De plus, puisque $\alpha(T_0') = v_0$ (proposition 7.3)

$$\Gamma(T_0') = \Gamma(T_0') \cap \text{Sol}(\bigwedge_{b \in \text{Act}(T_0')} v_0 \leq t_b).$$

Donc,

$$\begin{aligned}
&\text{Sol}(\theta_{(a',T')} \wedge v_0 \leq t_{a'} : T_0') \\
&= \Gamma(\theta_{(a',T')}) \cap \Gamma(T_0') \cap \text{Sol}(\textstyle\bigwedge_{b \in \text{Act}(T_0)} v_0 \leq t_b)) \\
&\subseteq \Gamma(\theta_{(a',T')}) \cap \Gamma(T)_{v_0/t_{a_0}} \cap \text{Sol}(\textstyle\bigwedge_{b \in \text{Act}(T_0)} v_0 \leq t_b)) \\
&= \emptyset.
\end{aligned}$$

Nous pouvons, donc, conclure que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R}$. Nous avons fini la preuve du cas (1). Les cas (2) et (3) sont triviaux: dans le cas (2) on a que

$$\mathcal{T} = \mathcal{S} = \{m \leq t_o - v_0 \leq M : T'\}.$$

Donc, $(\mathcal{T}, \mathcal{S}) \in \approx \subseteq \mathcal{R}$.

Dans le cas (3), puisque $T' = \uparrow$, alors

$$\mathcal{T} = \mathcal{S} = \{\uparrow\}.$$

Nous avons maintenant fini la preuve de la propriété 1. La preuve de la propriété 2 est similaire. Nous concluons que $(\mathcal{T}, \mathcal{S}) \in \mathcal{R} \subseteq \simeq$, ce qui conclut la preuve de la proposition. ∎

Bibliographie

[1] M. Abadi and L. Lamport. Composing specifications. In *Proceedings of the REX Workshop on Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 1–41, 1990.

[2] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi della Rocha, editors, *Automata languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 1–17, 1989.

[3] J. F. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11), November 1983.

[4] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 414–425. Computer Society Press, 1990.

[5] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time systems. *Information and Computation*, 104(1):2–34, 1993.

[6] R. Alur and D. Dill. Automata for modeling real time systems. In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, 1990.

[7] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[8] R. Alur and D. L. Dill. The theory of timed automata. In *Lecture Notes in Computer Science 600*, pages 45–73. Springer-Verlag, 1991.

[9] R. Alur and D. L. Dill. Automata-theoretic verification of real-time systems. In C. Heitmeyer and D. Mandrioli, editors, *Formal Methods for Real-Time Computing*, pages 1–32. John Wiley and Sons Ltd., 1996.

[10] R. Alur, A. Itai, R. Kurshan, and M. Yannakakis. Timing verification by successive approximation. In *Proceedings of the Forth Intern. Conf. on Computer-aided Verification*. Springer-Verlag, 1993.

[11] T. Amon. *Specification, Simulation and Verifiction of Timing Behavior*. PhD thesis, University of Washington, 1993.

[12] T. Amon, H. Hulgard, G. Boriello, and S. Burns. Timing analysis of concurrent systems: An algorithm for determining time separation of events. In *Proceedings of the ICCD*, 1993.

[13] J. Baeten and J. Bergstra. Real-time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[14] J. Baeten and J. Bergstra. Real space process algebra. *Formal Aspects of Computing*, 5(6):481–529, 1993.

[15] J. C. M. Baeten and J. A. Bergstra. Discrete time process algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.

[16] J. C. M. Baeten and S. Mauw. Delayed choice: an operator for joining message sequence charts. In *Proceedings of the 8th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, Bern, Switzerland, 1996.

[17] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[18] F. Balarin and A. L. Sangiovani-Vincentelli. A verification strategy for timing constrained systems. In *Proceedings of the Fourth International Workshop on Computer Aided Verification*, pages 151–163. Springer-Verlag, 1993.

[19] J. Bergstra and J. Klop. Algebra of communicating processes with abstraction. *Journal of Theoretical Computer Science*, 37:77–121, 1985.

[20] B. Berkane and E. Cerny. Vérification des chronogrammes hiérarchiques l'aide de 'ccs + contraintes'. In *Actes du Colloque BMW-94, Méthodes Mathématiques pour la Synthèse des Systèmes Informatiques*, ACFAS'94, pages 211–224, Montréal, Canada, 1994.

[21] B. Berkane, S. Gandrabur, and E. Cerny. Timing diagrams: Semantics and timing analysis. In *Proceedings of the Third Asia Pacific Conference on Hardware Description Languages APCHDL'96*, pages 112–119, Bangalore, India, 1996.

[22] B. Berkane, S. Gandrabur, and E. Cerny. Algebra for communicating timing charts for describing and verifying hardware interfaces. In *Proceedings of the Conference on Hardware Description Languages CHDL'97*, Toledo, Spain, 1997.

[23] T. Bolognese and E. Brinksma. Introduction to the iso specification language lotos. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[24] T. Bolognesi and F. Lucidi. Lotos-like process algebra with urgent or times interactions. In *Proceedings of the REX Workshop "Real-Time: Theory and Practice"*, Mook, the Netherlands, june 1991.

[25] G. Boriello. *A New Interface Specification Methodology and its Application to Transducer Synthesis.* PhD thesis, University of California, 1988.

[26] G. Boriello. Formalized timing diagrams. In *Proceedings of the European Conference on Design Automation*, Brussuls, Belgium, 1992.

[27] G. Bruno. *Model-based Software Engineering.* Chapman-Hall, London, 1995.

[28] J. A. Brzozovski, T. Gahlinger, and F. Mavaddat. Consistency and satisfiability of waveform timing specifications. *Networks*, 21:91–107, 1991.

[29] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic model checking: 10 to the power of 20 states and beyond. In *Proceedings of the Fifth Conference on Logic in Computer Science*, 1990.

[30] Richard Stevens Burington. *Handbook of Mathematical Tables and Formulas.* McGraw–Hill, 1965.

[31] E. Cerny, B. Berkane, P. Girodias, and K. Khordoc. *Hierarchical Annotated Action Diagrams – An Interface-Oriented Specification and Verification Method.* Kluwer Academic Publishers, 1998.

[32] E. Cerny and K. Khordoc. Interface specifications with conjunctive timing constraints: Realisability and compatibility. In *Proceedings of the 2nd AMAST Workshop on Real-Time Systems*, Bordeaux, France, 1995.

[33] E. M. Clarke and E. A. Emerson. Synthesis of synchronisation skeletons for branching-time temporal logic. In *Proceedings of the Logic of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, New York, 1981. Springer-Verlag.

[34] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transaction on Programming LAnguages and Systems*, 8(2):244–263, 1986.

[35] A. Coen-Porsini, R. Kemmerer, and D. Mandrioli. A formal framework for astral intralevel proof obligations. *IEEE Trans. on Software Engineering*, 20(8):548–561, 1994.

[36] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Lecture Notes in Computer Science*, 575:399–410, 1991.

[37] J.-P. Courtiat and R. C. Oliveira. On rt-lotos and its application to the formal design of multimedia protocols. *Annalls of Telecommunications*, 50(11-12):888–906, 1995.

[38] J.-P. Courtiat and R. C. Oliveira. A reachability analysis of rt-lotos specifications. In *Proceedings of the 8th Intern. Conference on Formal Description Techniques (FORTE'95)*, Montreal, Canada, 1995. Chapman and Hall.

[39] J.-P. Courtiat and R. C. Oliveira. Proving temporal consistency in a new multimedia synchronization model. In *Proceedings of the ACM Multimedia '96*, Boston, USA, 1996.

[40] D. Dill and H. Wong-Toi. Verification of real-time systems by successive over and under approximation. In *Proceedings of the Fourth Inter. Conf. on Computer Aided Verification*. Springer-Verlag, 1995.

[41] A. J. Gahlinger. *Coherence and Satisfiability of Waveform Timing Specifications.* PhD thesis, University of Waterloo, 1990. Research Report CS-90-11.

[42] C. Ghezzi, D. Mandrioli, and A. Morzenti. Trio, a logic language for executable specification of real-time systems. *Journal of Systems and Software*, 12(2):107–123, May 1990.

[43] P. Girodias. *Vérification des propriétés temporelles des interfaces matérielles à l'aide de la programmation logique avec contraintes.* PhD thesis, Univ. de Montréal, 1997.

[44] J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[45] A. P. Gupta. Formal hardware verification methods: A survey. *Formal Methods in System Design*, 1:151–238, 1992.

[46] C. Heitmeyer and N. Lynch. Formal verification of real-time systems using timed automata. In C. Heitmeyer and D. Mandrioli, editors, *Formal Methods for Real-Time Computing*, pages 83–106. John Wiley and Sons Ltd., 1996.

[47] C. Heitmeyer and D. Mandrioli. Formal methods for real-time computing: An overview. In C. Heitmeyer and D. Mandrioli, editors, *Formal Methods for Real-Time Computing*, pages 1–32. John Wiley and Sons Ltd., 1996.

[48] Y. Hirshfeld and F. Moller. Decidability results in automata and process theory. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency*, pages 102–148. Springer Verlag, 1996.

[49] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hal, New York, 1985.

[50] F. Jahanian and A. K. Mok. Safety analysis of timing properties of real-time systems. *IEEE Trans. on Software Eng.*, 12(9):890–904, 1986.

[51] F. Jahanian and A. K. Mok. Modechart: a specification language for real-time systems. *IEEE Trans. on Software Eng.*, 20(10):879–889, 1994.

[52] F. Jahanian and D. A. Stuart. A method for verifying properties of modechart specifications. In *Proceedings of the Real Time Systems Symposium*, December 1988.

[53] K. Khordoc and E. Cerny. Modeling cell-processing hardware with action diagrams. In *Proceedings of the ISCAS-94*, 1994.

[54] K. Khordoc and E. Cerny. Semantics and verification of action diagrams with linear timing constraints. *ACM Transactions on Design Automation of Electronic Systems*, 1995. accepté conditionellement à une revision et re-soumis en octobre 1996.

[55] K. Khordoc, M. Dufresne, E. Cerny, P. A. Babkine, and A. Silburt. Integrating behavior and timing in executable specifications. In *Proceedings of the IFIP Conference on HDL and their Applications*, pages 385–402, 1993.

[56] A. S. Klusner. *Models and axioms for a fragment of real time process algebra.* PhD thesis, CWI, 1993.

[57] R. P. Kurshan. *Computer Aided Verification of Coordinating Processes: the automata-theoretic approach.* Princeton University Press, 1994.

[58] S. Lenk. Extended timing diagrams as specification language. In *Proceedings of the European Conference on Design Automation (EURO DACS)*, Grenoble, France, 1994.

[59] K. L. McMillan. *Symbolic model checking – an approach to the state explosion problem.* PhD thesis, Carnegie Mellon Univ., 1992.

[60] K. L. McMillan and D. Dill. Algorithms for interface timing verification. In *Proceedings of the ICCD*, 1992.

[61] P. Merlin and D. Farber. Recoverability offfff communication protocols. *IEEE Trans. on Communications*, 24(9):1036–1043, 1976.

[62] R. Milner. *Communication and Concurrency.* Prentice-Hall, New-Jork, 1989.

[63] F. Moller and G. Birtwistle, editors. *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes In Computer Science.* Springer-Verlag, 1996.

[64] F. Moller and C. Tofts. A temporal calculus of communicating processes. In J. C. M.Baeten and J. W. Klop, editors, *Proceedings of CONCUR'90, Theories of Concurrency: Unification and Extension*, volume 458 of *LNCS*, pages 401–415, Amsterdam, the Netherlands, 1990. Springer-Verlag.

[65] J. Moller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. On the symbolic verification of times systems. Technical Report IT-TR-1999-024, Department of Information Technology, Technical University of Denmark, 1999.

[66] X. Nicollin, J.-L. Richier, J. Sifakis, and J. Voiron. ATP: an algebra for timed processes. In *Proceedings of the IFIP TC2 Working Conference on PRogramming Concepts and Methods*, Israel, 1990.

[67] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Proceedings of the Computer Aided Verification Conference*, volume 575 of *Lecture Notes in Computer Science*, 1991.

[68] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE TSE Special Issue on Real-Time Systems*, 18(9):794–804, 1992.

[69] J. Ostroff. *Temporal Logic for Real-Time Systems.* Research Studies Press LTD., Taunton, Somerset, England, 1989.

[70] J. Ostroff. Formal methods for the specification and design of real-time safety-critical systems. *Journal of Systems and Software*, 33(60):890–904, 1992.

[71] J. Ostroff. Visual tools for verifying real-time systems. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development*, volume 2 of *AMAST Series in Computing*, pages 83–101, Singapore, 1994. World Scientific Publishers Co.

[72] C. A. Petri. *Kommunikationen mit Automaten.* PhD thesis, Univ. of Bonn, 1962.

[73] C. A. Petri. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood, 1981.

[74] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Århus University, Computer Science Departement, Århus, the Netherlands, 1991.

[75] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[76] P. Rony. Interfacing fundamentals: Timing diagram conventions. *Computer Design*, pages 152–153, 1980.

[77] J. Rushby and F. von Henke. Formal verification of algorithms for critical systems. In *Proceedings of the ACM SIGSOFT Conf. on Software for Critical Systems*, pages 1–15, December 1991.

[78] K. Stolen, F. Dederichs, and R. Weber. Specification and refinement of networks of asynchronously communicating agents using the assumption/commitment paradigm. *Formal Aspects of Comuting*, 8(2):127–161, 1996.

[79] W. Thomas. Automata on infinte objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier Science Publishers, 1990.

[80] R. J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, Vrije Universiteit te Amsterdam, 1990.

[81] M. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency : Structure versus Automata*, volume 1043 of *LNCS*, pages 238–266. Springer-Verlag, 1996.

[82] H. Wong-Toi and D. L. Dill. Approximations for verifying timing properties. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development*, pages 177–204. World Scientific Publishing, 1994.

[83] Sifakis X. Nicollin, J. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114:131–178, 1994.

[84] J. Yang, A. K. Mok, and F. Wang. Symbolic model-checking for event-driven real-time systems. In *Proceedings of the 14th IEEE Real-Time Syst. Symp.*, Raleigh-Durham, December 1993.

[85] T. Yoneda, H. Schlingloff, and E. M. Clarke. Efficient timing verification based on one-safe petri-nets and real-time logic. In *FTC27 Workshop*, Toyama, Japan, 1992.

[86] T. Yoneda, A. Shibayama, H. Schlingloff, and E. M. Clarke. Efficient verification of parallel real-time systems. *Lecture Notes in Computer Science*, 697:321–332, 1993.

Index