ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

THÈSE

présentée
pour obtenir

LE GRADE DE DOCTEUR EN SCIENCES
DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN
Spécialité INFORMATIQUE

PAR

Yan JURSKI

Expression de la Relation Binaire d'Accessibilité pour les Automates à Compteurs Plats et les Automates Temporisé

Soutenue le 26 octobre 1999 devant le jury composé de :

MM.	Hubert COMON	Directeur de thèse
	Laurent FRIBOURG	Examinateur
	Denis LUGIEZ	Examinateur
	Nicolas HALBWACHS	Examinateur
	Andreas PODELSKI	Rapporteur
	Pierre WOLPER	Rapporteur

Laboratoire **S**pécification et **V**érification – CNRS UMR 8643;
ENS de Cachan; 61, Avenue du Président Wilson; 94235 CACHAN Cedex

0

Table des matières

Chapitre 1

Introduction

Ces dernières années, on a pu constater une mobilisation de moyens considérable en faveur de la certification de logiciels, elle se traduit même, sur la certification pour le passage à l'an 2000, par des campagnes d'information grand public en direction des PME et des PMI [1]. Il faut dire que, récemment, quelques problèmes sérieux de fiabilité se sont posés dans l'industrie du transport et des télécommunication. La prévention des problèmes liés à la gestion automatique de nos intérêts préoccupe dáutant plus les décideurs que malgrès l'importance des moyens mis en oeuvre il reste un cruel manque de savoir faire.

Les thèmes de vérification dans les programmes scientifiques sont donc jugés aujourd'hui stratégiques et prioritaires. Il faut reconnaître que les enjeux sont inchiffrables et la confiance que l'on porte en des outils puissants augmente d'autant la déception qui accompagne une défaillance. Lorsque cette confiance porte sur des projets industriels, des transactions financières, le transport ou l'énergie, la défaillance peut alors être ressentie comme une trahison.

Mais les problèmes rencontrés ne sont pas simples, et les outils qui sont employés pour le développement de programmes sont à la fois très puissants et difficilement maîtrisables, on sait d'ailleurs que des méthodes de vérifications générales n'existent pas, en effet les langages de programmation utilisés constituent tous des modèles de calculs universels, et aucun problème non trivial sur ces machines n'est décidable [Ric53]. Si l'on s'en tient là, il reste peu d'espoir de réaliser des programmes qui soient certifiés corrects de manière automatique. On peut par exemple s'en remettre au test, c'est à dire à une validation par des cycles d'essais puis de corrections. Cette approche, si elle est parfois nécessaire, pose tout de même des problèmes de confiance, comment être sur en effet d'avoir envisagé l'ensemble de tous les comportements ? Toutes les approches, y compris le test, profiteront d'une approche la plus formelle possible.

---

1. voir http://www.an2000.gouv.fr/

Démonstrateurs - Vérificateurs de modèle

Pour formaliser la vérification de programme, nous pouvons commencer par analyser les causes prévisibles d'erreurs. La première d'entre elles survient lors de la phase d'analyse du cahier des charges du projet, celui-ci est souvent imprécis et cela conduit à de mauvaises interprétations. Le résultat en est qu'il peut arriver qu'un programme soit réalisé "correctement" mais qu'il se révèle défaillant parce que non conforme aux attentes mal formulées. Ces erreurs d'interprétation seront malheureusement toujours difficiles à éviter puisque les caractéristiques du comportement d'un système sont exprimées en langue naturelle qui est elle-même imprécise et ambiguë. Pour essayer de limiter cette cause d'erreur on propose d'utiliser des langages formels de spécification, qui sont des langages de haut niveau souvent associés à des logiques. L'utilisateur doit alors respecter un certain nombre de règles d'écriture qui s'apparentent, dans un premier temps, à un code de bonne conduite. L'un de ces avantages est donc l'aide à l'écriture des spécifications du programme, et grâce à son cadre plus rigide que le langage naturel, d'éviter certaines formulations imprécises.

Un autre type d'erreur peut apparaître lors de la réalisation du programme lui même. Pour prendre en compte ces erreurs d'écriture de programme, deux approches se sont distinguées, qui se révèlent plus complémentaires que concurrentes.

La première approche consiste à exploiter les spécifications formelles et la logique dans laquelle elles sont exprimées en utilisant des démonstrateurs automatiques. Un programme écrit dans le même langage que les spécifications sera comparé par le démonstrateur à la propriété que l'on cherche à vérifier, ou à ses spécifications, en cherchant à montrer qu'elles en sont bien une conséquence logique. En pratique, ce problème est bien entendu encore indécidable, et la suite de règles de déduction doit être guidée continuellement par le programmeur, ceci même à des niveaux d'abstraction très faibles et pour des programmes qui semblent corrects quasiment à vu d'oeil. Des outils comme COQ [BBC$^+$96a] ou PVS [ORS92] permettent cette approche.

La seconde approche consiste à distinguer le langage sur lequel est écrit le programme de celui qui exprime les propriétés à vérifier. C'est l'approche dite de vérification de modèle, ou de *model checking* (voir par exemple [Sch99]). Le processus de vérification peut alors être mixte, s'appuyant à la fois sur le modèle disponible et sur le domaine d'expression des propriétés. Par exemple une des premières techniques de model checking pour des propriétés exprimées dans la logique CTL [CES86, QS82] consiste en un étiquetage des différents états du modèle par des sous propriétés de la propriété à vérifier. Des outils pour le model checking sont par exemple Kronos [Yov97] ou Hytech [HHW97].

De la même façon que pour les démonstrateurs automatiques, le model checking s'attaque à des problèmes indécidables, on cherche donc à s'appuyer sur des hypothèses supplémentaires. Elles s'inspirent d'un regard plus précis sur la complexité des programmes usuels dont en général l'écriture ne nécessite qu'une faible, voire très faible, part de l'expressivité des langages de programmation. Ainsi par exemple bien des programmes n'utilisent qu'un petit nombre de variables, ou de variables dépendantes.

Cette remarque est d'autant plus pertinente si l'on regarde de la même façon les propriétés que l'on cherche à vérifier. En général une grande partie des comportements d'un programme sont indépendants de ces propriétés, et par exemple la question de l'existence d'une réaction ne nécessite pas obligatoirement de connaître son amplitude.

Ces deux remarques justifient l'introduction de mécanismes d'abstractions ou d'approximations. Ainsi, on va chercher à abstraire les comportements d'un système dans des modèles les plus simples possibles qui peuvent encore rendre compte de propriétés. Dans certains cas, on modélisera les systèmes pqr un ensemble fini d'états, dans d'autres cas il sera nécessaire d'utiliser des modèles plus expressifs. De la même façon, pour les propriétés à vérifier, les logiques temporelles [Pnu77, Eme90] n'expriment qu'un certain type de propriétés.

Il faut reconnaître au model checking cette double possibilité d'abstraction : en distinguant le domaine des propriétés de celui des modèles, on peut de manière naturelle faire des hypothèses différentes sur chacun d'eux, soit en restreignant le type de propriétés que l'on veut vérifier, soit en restreignant l'expressivité des programmes par des contraintes sur leurs syntaxes. Une bonne abstraction est faite lorsque le modèle rend compte de suffisamment de comportements du système, et qu'il existe des algorithmes de vérification automatique des propriétés choisies sur ces modèles.

Modularité - Approche Symbolique

Mais les causes de l'impossibilité de la vérification ne sont pas toutes d'ordre théorique. En dehors des points de frictions entre l'indécidabilité et la décidabilité que nous venons d'évoquer avec l'utilisation d'abstractions, il reste les problèmes de réalisabilité. Il arrive souvent que les modèles soient toujours complexes et volumineux, et d'une taille prohibitive. Les raisons sont généralement qu'ils sont composés d'un grand nombre de sous systèmes, ou (et) qu'ils comportent des variables. Les méthodes de résolution qui font écho à ces deux problèmes sont respectivement l'analyse modulaire et l'approche symbolique.

Pour un système qui se compose de nombreux sous-systèmes, l'état global est décrit par le produit des états de chacun des sous-systèmes. Cette *explosion combinatoire* introduit une difficulté supplémentaire dans l'analyse en multipliant de la même façon les ressources nécessaires et les temps de calculs. L'*analyse modulaire* consiste à essayer de montrer des propriétés du système global par des déductions à partir d'analyses individuelles des sous-systèmes, ainsi il ne sera pas nécessaire de représenter effectivement l'ensemble de tous les états possibles. Le gain espéré en terme de ressources peut aller jusqu'à une diminution exponentielle et rendre ainsi l'analyse possible.

De l'autre côté, l'utilisation de variables dans les systèmes lorsqu'elles peuvent prendre un grand nombre de valeurs, voire même un nombre non borné, fait elle aussi exploser le nombre de cas à étudier puisque chaque état est décrit par une instance de ces variables. C'est en particulier dans ce cas que l'utilisation de méthodes de tests est sujette à caution. Pour pouvoir espérer effectuer cependant une analyse exhaustive, il est alors nécessaire de pouvoir représenter de manière symbolique des ensembles de valeurs possibles pour les variables, ou éventuellement des approximations de ces ensembles. Ainsi par exemple on a pu utiliser des ensembles périodiques [BW94a], ou des régions dans l'espace des valeurs caractérisées par des polyèdres [CH78]. Le récent succès de ces méthodes provient de l'utilisation d'une structure appelée BDD [Bry92] tout à fait adaptée à la description d'ensembles de valeurs de variables booléennes. Ce formalisme est d'ailleurs déjà souvent employé pour la vérification de circuits imprimés.

Les méthodes d'approximations [CH78, BGP97, DP99, BB99] lorsqu' elles sont "conservatives" permettent encore certaines analyses. En outre certaines d'entre elles assurent la terminaison des processus de semi-décisions, et d'autres permettent "d'accélérer" ce processus, et donc de le rendre possible.

Dans un système à variables, un module peut dans certains cas être vu comme une boite noire qui effectue des transformations sur les variables. Dans ce cas, il peut être utile de représenter de façon symbolique les relations sur les variables d'entrées et de sorties, et de pouvoir utiliser cette représentation dans le langage qui exprime les propriétés. Cette approche a le mérite de combiner à la fois le model checking et la démonstration automatique, on peut alors parler de model-checking déductif. Un outil réalisant cette possibilité est l'outil STEP [BBC$^+$96b].

Cadre de la thèse

Les résultats de cette thèse la placent dans le cadre du model checking déductif pour des systèmes utilisant des variables numériques, sans utiliser d'approximations pour autant. Ces modèles peuvent par exemple être utilisés pour étudier des systèmes à compteurs comme des péages d'autoroute, ou des systèmes dont les comportements sont soumis au passage continu du temps comme la gestion du trafic aérien. Les modélisations que nous avons retenues sont celles des *automates à compteurs* et des *automates temporisés* [AD94].

Nous regardons ces systèmes comme des opérateurs sur leurs variables qui réalisent une relation entre leurs valeurs d'entrées et de sorties, l'expression de cette relation nous donne une caractérisation de type boite noire du système. Nous pouvons utiliser cette caractérisation pour exprimer un certain nombre de propriétés dans une théorie qui se distingue des logiques temporelles utilisées habituellement. Cette logique est une extension de la théorie du premier ordre sur les réels munie de l'addition et des prédicats $<, \leq$. Lorsque le modèle est un automate à compteurs plats ou un automate temporisé donné, alors les propriétés de cette logique sont *décidables* (vérifiables).

Notre contribution consiste dans un premier temps en l'étude d'un fragment nouveau des automates à compteurs, que nous appelons ici automates à compteurs plats. Nous montrons alors que les automates temporisés peuvent être vus en un certain sens comme des automates à compteurs plats, ce qui conduit à de nouveaux résultats de décision sur ces automates. Cette réduction qui peut sembler surprenante illustre à quel point l'étude des automates plats semble pertinente.

Différentes connections

Cette thèse s'inscrit plus directement dans l'approche initiée, à notre connaissance, par H. Olsen et L. Fribourg [FO97] qui pour une classe particulière de réseau de Petri ont montré qu'il existe un ensemble significatif d'exécutions décrit par un automate plats. Au vu du caractère très particulier des opérations autorisées dans un réseau de Petri, on peut en déduire immédiatement que la relation binaire d'accessibilité est expressible.

Cette approche peut se nourrir d'une extension des opérations utilisées dans le modèle. De ce point de vue nous rejoignons les travaux de B. Boigelot [Boi98] qui a étudié les transformations linéaires "si $M\vec{X} \leq P$ alors $f(\vec{X}) = A\vec{X} + B$".

En comparaison, notre approche reprend celle de [FO97] avec des opérations étendues qui sont différentes de celles de B. Boigelot et leur sont toutefois incomparables.

Si les deux résultats [FO97, Boi98] ont principalement inspirés nos travaux, on peut également remarquer que les automates plats sont utilisés pour des systèmes à passage de messages, et donc non strictement numériques, par Boigelot-Wolper [BB96], Bouajjani-Habermehl [BH97] et Bouajjani-? [PA98].

Une conséquence stricte de nos résultats est de pouvoir exprimer en particulier la cloture transitive d'une relation. Ce résultat peut être vu comme une possibilité de "widening" exact [CH78] (ou de "meta-transition" [BW94a]) et donc être utilisé dans des outils d'approximations [BGP97, DP99]. D'autre part, et c'est ce que nous esquissons dans le chapitre 7 l'expression de la cloture transitive, semble permettre de vérifier les propriété expressibles dans les logiques temporelles classiques [NI97], mais également de nouvelles propriétés.

Plus techniquement, le raisonnement employé pour exprimer la relation binaire d'accessibilité pour les automates temporisés utilise des représentants "normaux" des exécutions en ordonnançant ses sous-exécutions. Dans cet esprit on retrouve les préoccupations rencontrées dans la communauté des réductions par ordre partiel.

Enfin pour justifier l'étude de la relation binaire d'accessibilité, on peut remarquer qu'en général on s'est interessé à l'accessibilité de deux facons symétrique que l'on a pris l'habitude de distinguer. La première consiste au calcul itératif de l'ensemble des états atteignables, partant d'une configuration initiale précise. On vérifie par la suite que cet ensemble ne contient pas d'états d'erreurs. La seconde approche consiste en une analyse à l'envers, partant d'une caractérisation de l'ensemble des erreurs possibles, on regarde comment atteindre en une étape cet ensemble, puis on itère ce procéssus en caractérisant l'ensemble des configurations initiales succeptibles de mener en moins de n étapes à un états d'erreurs. On peut alors s'assurer que selon la configuration initiales il existe ou pas une facon d'atteindre un état non souhaité. L'expression de la relation binaire d'accessibilité, qui permet de n'instancier ni l'état initial, ni un ensemble final d'erreurs, permet donc une vue uniforme de ces deux approches. Elle ne se réduit pas pour autant à elles : pouvoir utiliser cette relation permet en particulier d'exprimer une parties des propriétés dites "d'autostabilistation". Typiquement pour nous il s'agit de s'assurer que quelle que soit la configuration initiale du système, celui ci, finalement, atteindra un comportement correct. Ce type de robustesse est requis par exemple dans les sytèmes qui peuvent être perturbés par des manipulations et que l'on ne peut pas réinitialiser physiquement.

Epilogue

Le sujet que nous considérons, la vérification, est un sujet très appliqué où la complexité des processus de décision est très critique pour juger de l'importance des résultats. Toutefois nous ne sommes pas toujours capable de cerner les raisons exactes du succès de telle ou telle méthode. Cette thèse, si elle présente des techniques dont la complexité est inintéressantes en pratique (exponentielles imbriquées) a plutôt pour objet une étude de fondements, esquissant un cadre pour expliquer le succès de certains modèles. Nous avons défendu notre point de vue en particulier pour les automates temporisés. Cette réflexion permet d'ouvrir, nous semble t'il, de nouvelles perspectives dans le choix de modèles.

Organisation du document

Chapitre 2 Nous définissons les modèles d'automates à variables et la relation binaire d'accessibilité. Nous définissons également les automates à compteurs, les automates à compteurs plats et les automates temporisés. Nous y précisons la transformation d'un automate temporisé en automates à compteurs.

Chapitre 3 Nous reformulons le problème de l'expression de la relation binaire d'accessibilité en problème de graphe paramétré par la longueur des exécutions. Nous y montrons la correspondance des deux formulations, et nous montrons comment la procédure d'élimination, utile pour exprimer la relation au premier ordre, s'exprime plus simplement sur les graphes.

Chapitre 4 Nous y résolvons le problème de l'expression de la relation binaire d'accessibilité pour les automates à compteurs plats en analysant le problème de graphe associé. Il s'agit de caractériser les chemins dans des graphes par une formule du premier ordre, puis des ensembles pour leurs représentants de poids minimum.

Chapitre 5 Ce chapitre présente une condition suffisante qui permet la réduction du problème de l'expression de la relation binaire d'accessibilité d'un automate quelconque à celui d'automates plats. Ce résultat, présenté comme un résultat de théorie des langages, n'est pas spécifique aux automates à compteurs et pourrait s'appliquer à d'autres types de modèles.

Chapitre 6 Nous illustrons ici le fait que la réduction proposée dans le chapitre 5 est applicable pour une classe non triviale d'automates étendus, les automates temporisés qui se sont imposés comme le modèle de référence des systèmes temps réels. Il faut pour cela exhiber une relation sur les exécutions des automates temporisés qui respecte les conditions du chapitre 5. Il doit s'agir d'une congruence droite d'indice fini, qui permette également de commuter deux séquences équivalentes en préservant les valeurs de horloges. Ce chapitre est le plus technique de cette thèse.

Chapitre 7 Nous regroupons dans ce chapitre les conséquences, positives pour le model checking, des résultats d'expression de la relation binaire d'accessibilité. Nous y définissons une logique étendue par des modalités qui s'interprètent avec ces relations, et nous donnons des éléments de comparaison avec les logiques temporelles. Nous y faisons aussi la conjecture que cette logique est strictement plus expressive que $TCTL$ sans en apporter la preuve.

Chapitre 8 Conclusion et perspectives.

Chapitre 2

Automates étendus

2.1 Introduction

Pour étudier les systèmes informatiques, on a cherché à les représenter de manière indépendante de leur implémentation en utilisant des structures mathématiques. Pour n'en citer que quelques unes, on peut parler des automates, des automates étendus comme les automates à compteurs ou les automates à files, bien entendu les machines de Turing ou encore les réseaux de Petri.

Tous ces modèles ont en commun une distinction, que l'on retrouve dans les programmes, entre contrôles et données : un programme étant un ensemble de directives, il n'est composé que d'un ensemble fini d'instructions (le contrôle), alors que les informations qu'il traite (les données) peuvent être de taille arbitraire.

Sans présenter en détail chacun des modèles précédents, on peut dire que les automates à compteurs comme les réseaux de Petri travaillent sur des variables numériques, que les machines de Turing ont à leur disposition des "rubans" du type bande magnétique, et que les automates à files travaillent sur des "canaux" qui se comportent comme des tuyaux dont la latence sert de mémoire.

L'étude des automates d'états finis, c'est-à-dire sans structure de données, a été productive et a apporté des satisfactions concrètes puisque leurs liens avec la logique a permis de développer des outils pour la vérification automatique. Mais un bon nombre de programmes réels sont dénaturés par leur approximation dans ce modèle. Par exemple on ne peut pas toujours regarder un programme critique qui doit réagir en fonction de données continues, telles que le temps ou la température, comme un programme sans variable, puis vérifier des propriétés concernant son temps de réaction ou une relation entre les variables. Cet aspect est l'une des raisons du regain d'intérêt pour les modèles avec structure de données. D'un autre côté, on peut modéliser avec des automates d'états finis, des systèmes dont les structures de données sont de taille bornée. Mais cela se fait au prix d'un codage des valeurs possibles qui se retrouve dans le nombre d'états des automates. Ce procédé conduit alors souvent à des modèles de taille prohibitive, ce qui empêche leur analyse effective. Mais ce blocage est artificiel puisqu'il procède certainement plus d'une mauvaise méthodologie que d'un problème intrinsèque : la méthode utilisée est à rapprocher d'un programmeur qui construit son programme en énumérant l'ensemble des cas possibles. Cette mauvaise approche motive elle aussi l'étude des systèmes qui séparent contrôles et données. Et plus encore, pour éviter de retomber dans cette tentation de codage, on désire être capable d'analyser des systèmes pour lesquels les données peuvent être de taille arbitraire.

Nous introduisons ici une modélisation pour de tels systèmes : les automates étendus. Le fonctionnement de ces modèles est décrit par une sémantique, qui permet de définir la relation binaire d'accessibilité. Par la suite nous introduisons la classe des automates à compteurs et discutons de leur expressivité. Apres un rappel de la définition des automates temporisés, nous montrons comment ils peuvent être vus comme des automates à compteurs.

La restriction des automates à compteurs aux automates dits "plats" sera étudiée dans le chapitre 4, et la restriction correspondant au codage d'un automate temporisé dans le chapitre 6.

2.2 Automates étendus

2.2.1 Définition - Sémantique - Relation Binaire d'Accessibilité

Définition 1 (Automates étendus) *Un automate étendu* $A = (Q, X, \Sigma, \Delta)$ *est composé d'un ensemble d'états* Q*, d'un nombre fini de noms de variables* X*, d'un ensemble d'étiquettes* Σ*, et d'une relation de transition* $\Delta \subset Q \times \Sigma \times Q$*. On associe à ces automates un ensemble de noms de variables primés* X'*, de sorte que si* $X = \{x_1 \dots x_n\}$*,* X' *soit egal à* $\{x'_1 \dots x'_n\}$*. Les variables sont interprétées sur un domaine* $\mathcal{D}$*.*

Cette structure inerte est animée par des règles sémantiques qui précisent les calculs de l'automate entre deux configurations.

Définition 2 *Une configuration d'un automate étendu* (Q, X, Σ, Δ) *est une paire* $(q, V) \in Q \times \mathcal{D}^{|X|}$ *représentant un état de contrôle* q *et une valuation* V *des variables.*

Définition 3 (Sémantique, Calculs) *Une sémantique d'un automate* $A = (Q, X, \Sigma, \Delta)$ *associe, par une fonction* Φ*, à chaque élément de* Σ *une relation sur le domaine des variables. Ces relations permettent de définir les pas de calculs et les exécutions d'un automate :*

On appelle un pas de calcul de l'automate A*, muni d'une sémantique* Φ*, une relation ternaire* $\xrightarrow[\Phi,A]{}$ *, telle que* $(q, V) \xrightarrow[\Phi,A]{a} (q', V')$ *ssi* $(q, a, q') \in \Delta$ *et* $\Phi(a)(V, V')$*.*

Une exécution, ou calcul, est une succession de pas de calcul.

On étend cette relation sur les mots : pour $w \in \Sigma^*$, $(q,V) \xrightarrow[\Phi,A]{w} (q',V')$, ssi il existe une exécution $(q,V) \xrightarrow[\Phi,A]{a_1} (q_1,V_1) \xrightarrow[\Phi,A]{a_2} (q_2,V_2) \ldots \xrightarrow[\Phi,A]{a_n} (q_n,V_n) = (q',V')$ avec $w = a_1 \cdots a_n$.

Lorsqu'il n'y a aucune ambiguité dans le contexte sur l'automate et la sémantique dont il s'agit on notera $\xrightarrow{a}$ au lieu de $\xrightarrow[\Phi,A]{a}$.

Remarque 1 *D'ordinaire, on étiquette directement les transitions par la formule $\Phi(a)$. Cette distinction entre sémantiques nous sera utile de facon localisée, comme dans la remarque 2, pour l'émulation (Définition 7), ou pour pour une transformation des formules selon le domaine d'interprétation entier ou dense dans le chapitre 3. Lorsqu'il n'y a pas lieu de distinguer différentes sémantiques nous pourrons utiliser les notations habituelles qui consistent en un étiquetage direct des transitions par la formule.*

Définition 4 (Relation binaire d'accessibilité)
La relation binaire $\xrightarrow[\Phi,A]{*}$, est la projetée de $\xrightarrow[A]{w}$ pour $w \in \Sigma^*$: est définie ssi il existe $w \in \Sigma^*$ tel que $(q,V) \xrightarrow[\Phi,A]{w} (q',V')$. Elle est appelée relation binaire d'accessibilité. Si l'on s'intéresse à la relation entre variables lorsque les états sont fixés, on note $\xrightarrow[q,q',\Phi,A]{}$ (ou $\xrightarrow[q,q']{}$ si le contexte est clair).

Définition 5 *On étend la fonction de sémantique aux mots, en définissant la relation binaire sur $\mathcal{D}$, $\Phi(a_1 \cdots a_n)$ comme étant la composée des relations $\Phi(a_i)$:*

$$\Phi(a_1 \cdots a_n)(V,V') \textit{ ssi } \exists V_0 = V, \exists V_1 \ldots, V_n, (V_n = V') \bigwedge_i \Phi(a_i)(V_{i-1},V_i)$$

Et aux langages en posant $\Phi(L)(V,V')$ ssi $\exists w \in L$ tel que $\Phi(w)(V,V')$.

Nous présentons ici une suite d'exemples qui viennent illustrer les définitions précédentes. La première sémantique que nous introduisons est la sémantique des chemins :

Définition 6 (sémantique des chemins) *la fonction Φ associée est telle que l'image de chaque élément de Σ est la relation universelle sur $\mathcal{D} \times \mathcal{D}$.*

Les exécutions deviennent alors indépendantes des variables, on peut donc les noter $q \xrightarrow[A]{w} q'$. C'est sur leurs domaines que l'on distinguera les calculs avec sémantique des chemins des calculs avec une autre sémantique. Ainsi :

Remarque 2 *Pour tout automate A muni d'une sémantique Φ on peut décomposer les exécutions en utilisant la correspondance :*

$$\left.\begin{array}{l} q \xrightarrow[A]{w} q' \scriptstyle(\textit{sémantique de chemin}) \\ \Phi(w)(V,V') \end{array}\right] \Leftrightarrow (q,V) \xrightarrow[\Phi,A]{w} (q',V')$$

Exemple 1 (Les automates de mots) *Les variables sont interprétées sur l'ensemble des langages rationnels, la fonction Φ_m associe à chaque lettre $a \in \Sigma$ la relation telle que $\Phi_m(a)(V, V')$ ssi $V' = V \cdot a$. ($\cdot$ est l'opération de concaténation). Le langage des mots qui relient deux états q à q' d'un automate A, noté $L_{q,q'}$, se définit comme étant $\{w, (q, \epsilon) \xrightarrow[\Phi_m, A]{*} (q', w)$*

On peut encore préciser la remarque 2 puisque l'on a aussi :

Remarque 3 *Pour tout automate A muni d'une sémantique Φ on a :*

$$(V, V') \models \Phi(L_{q,q'}) \Leftrightarrow V \xrightarrow[q,q',A,\Phi]{*} V'$$

Exemple 2 (Les automates à une file) *L'alphabet de ces automates se partitionne en deux ensembles $\Sigma \uplus \overline{\Sigma}$ qui sont respectivement l'alphabet d'émission et de réception. Les variables sont interprétées sur l'ensemble des langages rationnels sur Σ. La fonction Φ associe à chaque lettre $a \in \Sigma$ la relation telle que $\Phi(a)(V, V')$ ssi $V' = V \cdot a$, et à chaque lettre $\overline{a} \in \overline{\Sigma}$ la relation telle que $\Phi(\overline{a})(V, V')$ ssi $a \cdot V' = V$*

Ces deux exemples sont tels que $\Phi(w) = \Phi(w') \neq \emptyset \Leftrightarrow w = w'$. On peut bien entendu introduire des relations plus grossières. C'est le cas par exemple des automates à files FIFFO [SFRC99] (First In First Fireable Out), qui modélisent des systèmes à canaux où une réception peut se faire même si la donnée n'est pas encore présente à l'extrémité d'un canal :

Exemple 3 (Les automates à une file FIFFO) *L'alphabet de ces automates se partitionne en deux ensembles $\Sigma \uplus \overline{\Sigma}$ qui sont respectivement l'alphabet d'émission et de réception. Les variables sont interprétées sur l'ensemble des langages rationnels sur Σ. La fonction Φ associe à chaque lettre $a \in \Sigma$ la relation telle que $\Phi(a)(V, V')$ ssi $V' = V \cdot a$, et à chaque lettre $\overline{a} \in \overline{\Sigma}$ la relation telle que $\Phi(\overline{a})(V, V')$ ssi V peut se décomposer en $V_1 \cdot a \cdot V_2$ avec $V_1 \in (\Sigma \setminus \{a\})^*$ et $V' = V_1 \cdot V_2$*

Dans le cas de la sémantique FIFFO, on n'a plus $\Phi(w) = \Phi(w') \neq \emptyset \Leftrightarrow w = w'$, puisque par exemple $\Phi(\overline{a} \cdot \overline{b}) = \Phi(\overline{b} \cdot \overline{a})$. On peut utiliser cette propriété en exploitant la remarque 2, ainsi lorsque Φ est une relation d'équivalence sur Σ^*, il est suffisant de connaître des représentants de $\Sigma^*_{/\Phi}$ pour caractériser la relation $\xrightarrow{*}$.

Remarque 4 *Soit A un automate étendu dont la fonction de sémantique Φ est une relation d'équivalence sur Σ^*, on a :*

$$\xrightarrow[A,\Phi]{*} = \bigcup_{c \in \Sigma^*_{/\Phi}} \xrightarrow[A,\Phi]{c}$$

Nous développerons cette idée dans le chapitre 5 pour l'appliquer lors de l'étude des automates temporisés dans le chapitre 6.

Lorsque l'on considère deux automates à compteurs différents, avec deux sémantiques toutes aussi différentes, il peut y avoir là aussi une certaine correspondance entre les exécutions de l'un et de l'autre qui préserve en un sens la relation binaire d'accessibilité. Nous définissons ici formellement une telle correspondance, l'émulation.

Définition 7 (émulation) *Un automate étendu* $A' = (Q', X, \Sigma', \Delta')$ *muni d'une sémantique* Φ' *émule l'automate* $A = (Q, X, \Sigma, \Delta)$ *muni de la sémantique* Φ *s'il existe une application* $\Theta : Q' \times Q' \rightarrow Q \times Q$ *telle que :*

$$(q, V) \xrightarrow[\Phi, A]{*} (q', V') \Leftrightarrow \exists p, p', \Theta(p, p') = (q, q') \wedge (p, V) \xrightarrow[\Phi', A']{*} (p', V')$$

De cette définition découle immédiatement la propriété :

Propriété 5 *L'émulation préserve la relation binaire d'accessibilité, puisque l'on a :*

$$\xrightarrow[q, q', \Phi, A]{*} = \bigcup_{(p,p') \in \Theta^{-1}(q,q')} \xrightarrow[p, p', \Phi', A']{*}$$

Pour ce qui suit, on s'intéresse plus particulièrement aux modèles interprétés sur des domaines numériques, et désormais les relations $\Phi(w)$ seront des formules interprétées sur ces domaines.

2.2.2 Automates à Compteurs

Pour une théorie $\mathcal{T}h$ interprétée sur un domaine $\mathcal{D}$, si X est un ensemble de variables, on note $\mathcal{F}(X)$ un sous ensemble des formules de $\mathcal{T}h$ qui ont leurs variables libres dans X.

Les théories que l'on considère par la suite sont les théories additives $\mathcal{T}h = (+, <, \leq, Entier)$. Ses termes sont construits sur un ensemble de variables, avec les constantes 0 et 1 et le symbole de fonction $+$. Les formules sont construites en effectuant des quantifications du premier ordre, et en utilisant les connecteurs logiques sur les formules atomiques. Celles ci sont de la forme $u < v, u \leq v$ ou $Entier(u)$ pour des termes u, v. Les domaines d'interprétation que l'on considère sont $\mathcal{D} = \mathbb{N}, \mathbb{Z}, \mathbb{Q}$ ou $\mathbb{R}$, avec les interprétations habituelles pour $+, <, \leq$. Le prédicat $Entier(x)$ est vérifié ssi x est un entier.

On définit le fragment $\mathcal{F}_c(X)$ comme étant le plus petit ensemble de formules de $\mathcal{T}h$ clos par conjonction, et qui $\forall x, y \in X, \forall c \in \mathbb{Z}$, contienne $x \leq y + c, x < y + c, x < c, x \leq c$. Puisque cette théorie est close par conjonction, on pourra utiliser le symbole $=$ au lieu d'une double inégalité.

Définition 8 (Automate à compteurs)
Un automate étendu (Q, X, Σ, Δ) *interprété dans* $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ *ou* $\mathbb{R}$*, dont la fonction de sémantique est à valeurs dans* $\mathcal{F}_c(X \cup X')$ *est un automate à compteurs.*

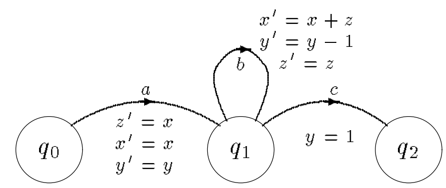


FIG. 2.1: *Représentation d'un automate à compteurs*

Exemple 4 *La figure 2.1 donne un exemple d'automate à compteurs* $\mathcal{A} = (Q = \{q_0, q_1, q_2\}, X = \{x, y, z\}, \Sigma = \{a, b, c\}, \Delta = \{t_1, t_2, t_3\})$ *avec* $t_1 = (q_0, a, q_1)$, $t_2 = (q_1, b, q_1)$, *et* $t_3 = (q_1, c, q_2)$ *et* Φ *est définie par :*

$$\Phi(a) \stackrel{def}{=} (z' = x \wedge x' = x \wedge y' = y)$$
$$\Phi(b) \stackrel{def}{=} (x' = x + z \wedge y' = y - 1 \wedge z' = z)$$
$$\Phi(c) \stackrel{def}{=} (y = 1)$$

Si l'on note les valuations par un vecteur $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$, *un exemple de calcul est donné par :*

$$(q_0, \begin{pmatrix} 4 \\ 3 \\ 10 \end{pmatrix}) \xrightarrow[\mathcal{A}]{a} (q_1, \begin{pmatrix} 4 \\ 3 \\ 4 \end{pmatrix}) \xrightarrow[\mathcal{A}]{b} (q_1, \begin{pmatrix} 8 \\ 2 \\ 4 \end{pmatrix}) \xrightarrow[\mathcal{A}]{b} (q_1, \begin{pmatrix} 12 \\ 1 \\ 4 \end{pmatrix}) \xrightarrow[\mathcal{A}]{c} (q_2, \begin{pmatrix} 12 \\ 1 \\ 7 \end{pmatrix})$$

2.2.3 Expressivité des Automates à Compteurs

Les automates à compteurs présentés ici sont une extention de ceux étudiés par Minsky [Min61] ou Fisher [FMR68]. Lorsque l'on considère des automates étendus de domaines d'interprétations numériques, et même si les fonctions de sémantique se définissent dans des fragments très réduits de $\mathcal{T}h$, les automates ont en général le même pouvoir d'expression que les machines de Minsky, elles-mêmes équivalentes aux machines de Türing. Ainsi, pour tout fragment $\mathcal{F}(\mathcal{X})$ permettant d'encoder les machines de Minsky, le théorème de Rice [Ric53] s'applique. Celui ci énonce qu'aucune propriété non triviale sur un ensemble récursivement énumérable n'est décidable. C'est donc également le cas pour les automates à compteurs en général.

Dans cette section nous présentons la machine de Minsky, puis nous envisageons deux sortes de restrictions sur ce modèle pour espérer obtenir une analyse automatique : une réduction du fragment utilisé les transformations sur les variables, et une restriction sur la structure de l'automate.

Définition 9 *Une Machine de Minsky est un uplet* (Q, C, q_0, Q_F, Δ). $q_0 \in Q$ *est apelé* l'état initial, $Q_F \subset Q$ *est l'ensemble des* états finaux, Δ *est une relation* *sur* $Q \times \mathcal{F}_M(X, X') \times Q$ *où* $\mathcal{F}_M(C, C')$ *est l'ensemble des formules :*

$$\mathcal{F}_M(C, C') = \bigcup_{x \in C} \begin{cases} (\bigwedge_{y \in C} y' = y) \wedge (x = 0), & \textit{test à zéro de } x \\ (\bigwedge_{y \in C} y' = y) \wedge (x > 0), & \textit{test de } x \\ (\bigwedge_{y \in C \setminus \{x\}} y' = y), \wedge (x' = x - 1) & \textit{décrément de } x \\ (\bigwedge_{y \in C \setminus \{x\}} y' = y), \wedge (x' = x + 1) & \textit{incrément de } x \end{cases}$$

Une exécution d'une machine de Minsky est une suite d'étapes $(q, V) \overset{f}{\mapsto} (q', V')$ avec $(q, f, q') \in \Delta$ et $(V, V') \models f$.

Ces machines ne modifient donc les valeurs qu'au plus d'un compteur par franchissement de transition, soit en l'incrémentant de 1, soit en lui retranchant 1. Une transition peut aussi simplement tester si la valeur d'un compteur est nulle ou non avant de changer d'état.

Il est clair que les automates à compteurs codent les machines de Minsky puisque $\mathcal{F}_M \subset \mathcal{F}_c$. Nous envisageons deux approches orthogonales pour permettre de choisir un modèle qui soit candidat à une étude possible. La première est de considérer un fragment $\mathcal{F}_c \not\subset \mathcal{F}_M$ qui, puisqu'il est moins expressif, pourrait ne pas encoder toutes les machines de Turing. Par exemple on peut interdire l'opération d'incrément, de décrément ou de test à zéro, mais il faut aussi éviter qu'un encodage de ces opérations ne soit possible par combinaisons. On peut également restreindre le nombre de variables utilisées. On obtient alors des modèles pour lesquels on arrive à vérifier des classes de propriétés.

La seconde possibilité est de faire des restrictions sur l'ensemble des transitions de l'automate étendu, de sorte que là encore seule une sous classe des machines de Minsky ne soit encodable. On peut alors accepter un fragment $\mathcal{F}_c \supset \mathcal{F}_M$. C'est cette approche que nous allons illustrer dans le chapitre 4 avec les automates à compteurs plats.

Définition 10 *Une boucle simple sur un état* q_0 *d'un automate* $\mathcal{A} = (Q, X, \Sigma, \Delta)$*, est une suite de transitions* $((q_0, a_1, q_1), (q_1, a_2, q_2) \ldots, (q_n, a_{n+1}, q_{n+1}))$ *avec* $q_{n+1} = q_0$ *et* $\forall i \neq j \in [0..n], q_i \neq q_j$*.*

Définition 11 (Automate plat) *Un automate* $\mathcal{A} = (Q, X, \Sigma, \Delta)$ *est dit* plat *s'il ne contient au plus qu'une boucle simple par état.*

Un exemple d'automate plat est donné par la figure 2.2.

Le langage de ces automates de la figure 2.2 peut se caractériser par une expression rationnelle de la forme $\Sigma_{i \in I} \Pi_{j \in J_i} u_i \cdot (v_i)^*$. Ainsi le langage de l'automate entre les états q et q' s'exprime par l'expression $L_{q,q'} = a \cdot (b \cdot c \cdot d)^* \cdot e \cdot g^* + a \cdot (b \cdot c \cdot d)^* \cdot b \cdot c \cdot f \cdot g^*$.

La classe des automates plats a de nombreux échos dans la littérature, depuis que récemment le point de vue de l'accélération de convergence lui a donné un nouveau souffle. Dans [BW94b] les auteurs soulignent l'intéret qu'apporte la possibilité de calculer en une étape l'effet de l'itération d'une boucle un nombre arbitraire de fois. Cette idée est utilisée par la suite dans de nombreux papiers

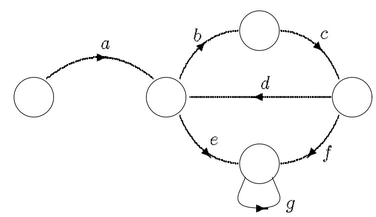


FIG. 2.2: *Exemple d'automate plat*

qui examinent son application sur des modèles à domaine numérique ou régulier [Boi98, BGP97, BH97]. Leurs résultats sont présentés sur des modèles sans la restriction de platitude, mais dans le cadre de semi-algorithmes dont la terminaison n'est pas garantie. La classe pour laquelle ils s'appliquent en toute certitude n'est ni plus ni moins que celle des automates plats.

L'une des thèses de cet exposé est que ce critère syntaxique très fort, ne s'accompagne pas d'une perte d'expressivité excessive. Elle est appuyé par les deux sentiments donnés d'un côté par l'intéret pratique des méthodes d'accélération de convergence [BW94a], et la difficulté de trouver une autre opération d'accélération de convergence qui soit plus forte que l'itération d'une séquence. Mais aussi de l'autre coté par deux résultat de réduction : il existe deux modèles non triviaux dont l'étude peut se ramener à celle des automates à compteurs plats. Le premier est celui des BPP, qui sont des restrictions des réseaux de Petri [FO97], le second est celui des automates temporisés présentés dans le chapitre 6

Parmis les fragments qui ont été étudiés, on peut citer le fragment "gap-order" [Rev90]: les contraintes autorisées sont de la forme $y_1 < y_2+c$, $y_1 \leq y_2+c$ pour des constantes c positives. Il a été étendu par exemple en introduisant des contraintes de périodicité [DCR94], et une addition faible [Rev98]. Ces restrictions très fortes assurent en particulier que quelques soient les compositions de ces contraintes on ne puisse pas exprimer à la fois l'increment et le décrément. Il n'est pas nécessaire alors de faire d'hypothèses sur la structure de l'automate.

Un résultat très fort de calcul d'itérations a été obtenu par B.Boigelot pour des transformations déterministes sur les entiers. La transformation est présentée à l'aide de matrices [Boi98], elle est composée d'une précondition arbitraire de la forme $MX \leq P$ qui si elle vérifiée transforme les valeurs des variables en $f(X) = AX + B$. Il existe bien entendu des conditions sur la matrice A pour que cette l'iteration soit expressible dansune logique décidable. En comparaison, notre fragment ne permet pas autant de liberté dans les préconditions. En ce qui concerne les transformations, la condition donnée sur la matrice A est que ses valeurs propres soient egale à zéro ou à l'unité [1] ainsi modulo un changement de base ces transformations non gardées sont alors dans notre fragment.

1. le critère requis est en fait plus précis, il faut qu'une puissance de la matrice A ait pour valeurs propres non nulle des puissances de la base dans laquelle on représente les entiers

On peut encore citer un résultat [KPRS94] où les transformations de x en x' sont de la forme $a \leq x' - x \leq b$ et $x' - x \equiv_a b$ et qui suggèrent à nouveau l'introduire future de contraintes de périodicité dans notre fragment.

2.3 Automates Temporisés

Nous étudierons également une autre classe d'automates que nous présentons ici avec leur sémantique originale. Nous allons montrer que l'étude de ces automates peut se ramener à celle d'automates étendus, et par un changement de variables appropriées, elle peut se ramener à celle d'automates à compteurs. En fait le résultat principal du chapitre 6 permet aussi de les observer, en un certain sens, comme des automates à compteurs plats.

2.3.1 Définition

Soit X un ensemble de variables. Soit $\mathcal{H}(X)$ le plus petit ensemble de formules clos par conjonction et qui $\forall x \in X, \forall c \in \mathbb{N}$ contienne les formules $x \leq c, x \geq c, x < c, x > c$. On note aussi $\mathcal{P}(X)$ l'ensemble des parties de X.

Définition 12 (Automate temporisé) *Un automate temporisé est un élément de la forme $\mathcal{A}_t = (Q, q_0, Q_F, H, \Delta, I)$, où Q est un ensemble fini d'états, $q_0 \in Q$ est un état initial, $Q_F \subset Q$ est un ensemble d'états finaux, H est un ensemble fini de variables interprétées dans $\mathbb{R}$ aussi appelées horloges, Δ est une relation de transition dans $Q \times \mathcal{H}(H) \times \mathcal{P}(H) \times Q$, et I une fonction d'invariants de Q dans $\mathcal{H}(H)$.*

Comme dans le cas des automates à variables, une configuration d'un automate temporisé $\mathcal{A}$ est un élément de $Q \times \mathbb{R}^{+|H|}$, constitué d'un état de contrôle et d'une valuation des horloges.

Nous présentons la sémantique classique des automates temporisés.

Définition 13 *Une étape de calcul de $\mathcal{A}$, notée $(q, V) \underset{\mathcal{A}}{\overset{a}{\longmapsto}} (q', V')$ correspond à l'un des deux cas suivants :*

- *$a \in \mathbb{R}^+, q = q'$ et $V' = V + a * \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ est tel que $V' \models I(q)$*
- *a est un élément (q, g, R, q') de Δ, tel que $V \models g, V' \models I(q')$, et il existe $t \in \mathbb{R}^+$ tel que $\forall i, v_i \in H \setminus R, v_i' = v_i + t$, et $\forall i, v_i \in R, v_i' = t$*

Un calcul est une suite d'étapes de calculs, et une exécution est un calcul entre $(q_0, \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix})$ et (q, V) pour $q \in Q_F$

Une étape de calcul est donc, soit un simple passage du temps uniforme pour chacune des horloges, soit un franchissement de transitions lorsque la garde correspondante est vérifiée. La configuration obtenue après le changement d'états et un éventuel passage du temps doit respecter l'invariant de l'état d'arrivée, les horloges spécifiées par le champs R de la transition ont été remises à zéro au moment du franchissement puis toutes les horloges ont progressé uniformément.

Les deux propriétés suivantes permettent de caractériser les calculs des automates temporisés d'une manière qui se prête à une transformation sous forme d'automates à compteurs. Elles sont des conséquences de l'associativité de l'addition dans $\mathbb{R}$.

Propriété 6 $\forall a, a_1, a_2 \in \mathbb{R}^+$ *tel que* $a = a_1 + a_2$, $\forall q, q', V, V'$ *on a :*

$$(q, V) \overset{a}{\longmapsto} (q, V') \ ssi\ \exists V_1, (q, V) \overset{a_1}{\longmapsto} (q, V_1) \overset{a_2}{\longmapsto} (q, V')$$

Propriété 7 $\forall a \in \Delta, \forall q, q', V, V'$, *on a :*

$$\exists V_1, \exists \delta \in \mathbb{R}^+, (q, V) \overset{a}{\longmapsto} (q', V_1) \overset{\delta}{\longmapsto} (q', V') \ ssi\ (q, V) \overset{a}{\longmapsto} (q', V')$$

Ainsi, l'existence d'un calcul pour un automate temporisé se ramène à l'existence d'un calcul de la forme :

$$(q_1, V_0) \overset{\delta_0}{\longmapsto} (q_1, V_1) \overset{a_1}{\longmapsto} (q_2, V_2) \overset{a_2}{\longmapsto} \dots \mapsto (q_i, V_i) \overset{a_i}{\longmapsto} (q_{i+1}, V_{i+1}) \dots$$

avec $\delta_0 \in \mathbb{R}^+$, et chaque $a_i \in \Delta$.

Nous exploitons alors cette forme normalisée pour encoder les exécutions des automates temporisés en exécutions d'automates à compteurs. Cette transformation est due à L.Fribourg et est présentée dans [Fri98].

2.3.2 Réduction en automates à compteurs

Soit $\mathcal{A}_t = (Q, q_0, Q_F, H, \Delta_t, I)$ un automate temporisé qui possède une horloge particulière τ qui n'est jamais remise à zéro, ni testée dans les gardes ou les invariants. Cette condition n'est pas restrictive puisqu'elle préserve fidèlement les exécutions. On associe à $\mathcal{A}_t$ l'automate à compteurs $\mathcal{A}_c = (Q', C, \Sigma, \Delta_c)$, tel que $Q' = Q$ et $\Sigma = \Delta_t$, si $H = \{h_1 \cdots h_n, \tau\}$, C est de la forme $\{c_1 \cdots c_n, \tau\}$, Δ_c est constitué des transitions (q, t, q') lorsque $t \in \Delta_t$ est une transition de q à q'. La fonction de sémantique Φ associée à cet automate est définie de sorte que si $t \in \Delta_t$ est égale à (q, g, R, q'), $\Phi(t)$ est égale à :

$$\begin{cases} \bigwedge_{h_i \in R} c'_i = \tau \\ \bigwedge_{h_i \in H \setminus R \setminus \{\tau\}} c'_i = c_i \\ \tau \leq \tau' \\ \bigwedge \tau' \ \#\ c'_i + c, \text{ lorsque } h_i \ \#\ c \text{ est une contrainte de } I(q) \\ \bigwedge (\tau \ \#\ c_i + c) \text{ lorsque } h_i \ \#\ c \text{ est une contrainte de } g \end{cases}$$

($\#$ représente indifféremment $<, >, =, \leq, \geq$)

Propriété 8 *Soit le changement de variables f défini sur H par $f(\tau) = \tau$, $f(h_i) = \tau - h_i$, et $a \in \Delta_t$, on a :*

$$(q, V) \underset{\mathcal{A}_t}{\overset{a}{\longmapsto}} (q', V') \textit{ ssi } (f(V), f(V')) \models \Phi(a)$$

Preuve : soit $a = (q, g, R, q') \in \Delta_t$, on rappelle qu'un calcul $(q, W) \underset{\mathcal{A}_t}{\overset{a}{\longmapsto}} (q', W')$ est défini ssi $W \models g, W' \models I(q)$, et $\exists \delta \in \mathbb{R}^+$ tel que $\forall h_i \in R, w'_i = \delta, \forall h_i \in H \setminus R, w'_i = w_i + \delta$. Puisque τ n'est remise à zéro par aucune des transitions de Δ_t, on a $\tau' - \tau = \delta$. Ainsi en remplacant δ par $\tau' - \tau$ les conditions précédentes s'écrivent :

- $(\tau - c) \ \# \ (\tau - w_i)$ lorsque $h_i \ \# \ c$ est une contrainte de g
- $\tau' \ \# \ c'_i + c$, lorsque $h_i \ \# \ c$ est une contrainte de $I(q)$
- $\tau \leq \tau'$
- $\forall w_i \in R, (\tau' - w'_i) = \tau$
- $\forall w_i \in H \setminus R \setminus \{\tau\}, (\tau' - w'_i) = (\tau - w_i)$

Ce qui correspond bien à la definition de $\Phi(a)$.
□

Nous avons effectué une normalisation des exécutions des automates temporisés en considérant celles qui se composent d'une transition de temps, puis d'une suite de transitions de Δ_t. Cette normalisation préserve la relation binaire d'accessibilité, et il existe un automate à compteurs qui modulo un changement de variables a le même comportement. Cette transformation justifie le point de vue que l'étude des automates temporisés puisse se ramener à l'étude des automates à compteurs.

Chapitre 3

Graphes de Contraintes

Nous avons vu lors des remarques 2 et 3 que le corps du problème de l'expression de la relation binaire d'accessibilité réside dans l'expression des formules $\Phi(w)$ ou $\Phi(L_{q,q'})$.

Dans le cas des automates à compteurs, où Φ est construite sur des formules du fragment $\mathcal{F}_c$, on se ramène à un problème de résolution de contraintes qui sont binaires. Il n'est pas original [CLR90] de représenter ces contraintes sous forme de graphes et ce chapitre ne fait que présenter le problème sous un autre angle. En particulier nous observerons que la procédure d'élimination des quantificateurs que nous utilisons pour résoudre les contraintes $\Phi(w)$, s'exprime de manière très élégante en terme de chemins.

Ce chapitre se décompose donc ainsi : nous faisons un rappel sur la technique de l'élimination de quantificateurs, puis nous définissons les graphes associés aux contraintes sémantiques d'un automate et la correspondance des deux représentations du problème. Alors nous montrons comment résoudre les contraintes sur les graphes. Nous concluons ce chapitre en présentant une famille particulière de graphes, les Tgraphes qui représentent les contraintes des automates temporisés.

3.1 Rappels sur la résolution de contraintes

Pour un mot $w = a_1 \cdots a_n$ sur un alphabet Σ^* d'un automate étendu, de fonction de sémantique Φ, la formule $\Phi(w)$ est de la forme :

$$\exists X_0 \ldots X_n, X_0 = X \wedge X_n = X' \wedge \bigwedge_{i=0..n-1} \Phi(a_{i+1})(X_i, X_{i+1})$$

Il s'agit d'une formule purement existentielle. De plus dans le cadre des automates à compteurs les sous formules de $\Phi(w_i)$ sont des contraintes binaires entre variables (dans $\mathcal{F}_c$). Ces particularités simplifient la résolution des contraintes $\Phi(w)$ puisqu'il existe une procédure, celle d'élimination des quantificateurs existentiels, qui donne une formule sans quantificateur qui a les mêmes modèles.

Cette procédure est également quasi indépendante du domaine d'interprétation des variables, et permet une résolution uniforme sur $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ ou $\mathbb{R}$.

Considérons dans un premier temps le cas où le domaine d'interprétation est $\mathbb{Q}$ ou $\mathbb{R}$. La base de cette procédure est donnée par le lemme 1 :

Lemme 1 *Soient ϕ, ϕ' des formules interprétées dans $\mathbb{R}$ ou $\mathbb{Q}$ de la forme*

$$\phi = \exists x \bigwedge_{i \in I_1} x \leq \psi_i \wedge \bigwedge_{i \in I_2} x < \psi_i \bigwedge_{j \in J_1} \varphi_j \leq x \wedge \bigwedge_{j \in J_2} \varphi_j < x \wedge \phi'$$

avec x non libre dans ψ_i, φ_j, ϕ'. Si $I_1 \uplus I_2, J_1 \uplus J_2$ sont finis, alors on a $\phi \models\!\!| \theta$ avec :

$$\theta \stackrel{def}{=} \bigwedge_{(i,j) \in I_1 \times J_1} \varphi_j \leq \psi_i \wedge \bigwedge_{(i,j) \in I_1 \times J_2 \cup I_2 \times J_1 \cup I_2 \times J_2} \varphi_j < \psi_i \wedge \phi'$$

Preuve :

Le sens $\models$ se déduit simplement de l'associativité de $\leq$ et de $<$. Pour la réciproque il faut exhiber un x convenable. Soit $\mathcal{M}$ un modèle pour θ, et σ la substitution correspondante. Lorsque $I = I_1 \uplus I_2$ et $J = J_1 \uplus J_2$ sont non vides, on pose $M = min_{i \in I}\{\psi_i.\sigma\}$ et $m = max_{j \in J}\{\varphi_j.\sigma\}$. Soit $v = (m + M)/2$, on peut vérifier qu'alors :

$$\mathcal{M}.\{x = v\} \models \bigwedge_{i \in I_1} x \leq \psi_i \wedge \bigwedge_{i \in I_2} x < \psi_i \bigwedge_{j \in J_1} \varphi_j \leq x \wedge \bigwedge_{j \in J_2} \varphi_j < x \wedge \phi'$$

Si m n'est pas défini, il suffit de considérer $v = M - 1$, et $v = m + 1$ lorsque c'est M qui n'est pas défini. On a encore le même résultat.

□

Ce lemme 1 peut se combiner avec d'autres opérations qui préservent les modèles. Dans le cas des réels ou des rationnels, on peut en particulier diviser ou multiplier des termes atomiques par des constantes non nulles, justifiant ainsi une élimination par pivot, comme celle de Gauss.

Exemple 5 *On considère ici deux exemples d'élimination dans $\mathbb{Q}$ ou $\mathbb{R}$:*

- *Soit $\phi(x) \stackrel{def}{=} \exists y, x < y \wedge y < 2x - 4$.*
  Le lemme 1 permet d'écrire $\phi(x) \models\!\!| x < 2x - 4$ qui se simplifie en $x > 4$
- *Soit $\phi(x) \stackrel{def}{=} \exists y, 3y \leq x \wedge x \leq 3y + 1$.*
  Puisque l'on interprète les variables dans un domaine dense, et que $\forall y, 3y \leq 3y + 1$, on a $\phi(x) \models\!\!| True$

Considérons à présent le cas de contraintes interprétées dans $\mathbb{N}$ ou $\mathbb{Z}$. On remarque qu'alors le prédicat $<$ est une conséquence du prédicat $\leq$: pour toutes formules ϕ et ψ on a $\phi < \psi \models\!\!| \phi + 1 \leq \psi$. Ainsi une version simplifiée du lemme 1 est suffisante:

Lemme 2 *Soient ϕ, ϕ' des formules interprétées dans $\mathbb{N}$ ou $\mathbb{Z}$ de la forme*

$$\phi = \exists x \bigwedge_{i \in I} x \leq \psi_i \wedge \bigwedge_{j \in J} \varphi_j \leq x \wedge \phi'$$

avec x non libre dans ψ_i, φ_j, ϕ' et I, J finis, on a $\phi \models\!\!\mid \theta$ avec :

$$\theta \stackrel{def}{=} \bigwedge_{(i,j) \in I \times J} \varphi_j \leq \psi_i \wedge \phi'$$

Preuve :

De la même façon que pour le lemme 1 on définit $M = min_{i \in I}\{\psi_i.\sigma\}$ et $m = max_{j \in J}\{\varphi_j.\sigma\}$, on peut alors prendre indifféremment $v = m$ ou $v = M$ pour conclure.

□

Exemple 6 *Pour illustrer les différences selon le domaine d'interprétation, reprenons les deux exemples précédents*

- $\phi(x) \stackrel{def}{=} \exists y, x < y \wedge y < 2x - 4$.
  On procède d'abord au remplacement des $<$ par des $\leq$. On obtient : $\phi(x) \models\!\!\mid x + 1 \leq y \wedge y \leq 2x - 5$. *Le lemme 2 permet d'écrire* $\phi(x) \models\!\!\mid x + 1 \leq 2x - 5 \models\!\!\mid 6 \leq x$. *En particulier, $x = 5$ qui était bien une solution entière à un système de contraintes interprétées sur des réels n'est plus une solution.*

- $\phi(x) \stackrel{def}{=} \exists y, 3y \leq x \wedge x \leq 3y + 1$.
  Ici, c'est la division par 3 qui ne préserve pas les modèles car $\mathbb{Q}$ ou $\mathbb{R}$ sont denses, mais pas $\mathbb{N}$ ou $\mathbb{Z}$. Ici encore l'ensemble des modèles ne se déduit pas de l'ensemble des modèles lorsque le domaine est dense. Une solution x doit avoir 0 ou 1 comme reste d'une division par 3.

Nous avons donc deux exemples d'élimination non uniforme selon les domaines d'interprétation. On peut toutefois isoler les causes de ces problèmes : les formules ψ_i et ϕ_i des lemmes 1 et 2 contiennent le prédicat $<$ ou ont des coefficients sur des variables à éliminer. Ainsi, si l'on se restreint à des formules de $\mathcal{F}_c$, et que l'on remplace éventuellement les inégalités strictes par de formules équivalente qui n'ont que des inégalités larges, comme nous l'avons indiqué pour les domaines d'interprétation entiers, on a alors un fragment pour lequel l'élimination est uniforme. Dans la suite on ne supposera d'ailleurs que cette transformation est toujours réalisée. Comme simple application des lemmes 1 et 2 ce fragment est également stable par composition :

Corollaire 9 *Soient ϕ, ψ deux formules dans $\mathcal{F}_c$, il existe une formule χ dans $\mathcal{F}_c$ telle que $\chi \models\!\!\mid \exists x, \phi \wedge \psi$.*

3.2 Graphes d'entrées-sorties

Nous allons utiliser le caractère uniforme selon les domaines de l'élimination pour modéliser les contraintes sur $\mathcal{F}_c(X)$ sous forme de graphes d'entrées-sorties. Sur cette nouvelle représentation aurons des outils synthétiques pour itérer des éliminations.

3.2.1 Définitions de $\Theta(g)$

Soit $V = \{v_1 \dots v_p\}$ un ensemble de sommets, pour tout entier i on définit $V_i = \{v_1^i \dots v_p^i\}$ un ensemble de sommets, et $\mathcal{V} = \bigcup_{i=0}^{\infty} V_i$.

Définition 14 (Graphe d'entrées-sorties) *Un graphe d'entrées-sorties sur V est un graphe de la forme $g = (\bigcup_{i=0}^{n} V_i, E)$ avec $E \subset \bigcup_{i=0}^{n} V_i \times \mathbb{Z} \times \{'<',' \leq'\} \times \bigcup_{i=0}^{n} V_i$.*

Les fonctions poids et ope sont les fonctions de projections de $\mathcal{V} \times \mathbb{Z} \times \{'<',' \leq'\} \times \mathcal{V})$ respectivement dans $\mathbb{Z}$ et $\{'<',' \leq'\}$.

La longueur d'un graphe d'entrées-sorties $g = (\bigcup_{i=0}^{n} V_i, E)$ notée $|g|$ est n.

Les graphes d'entrées-sorties sur V sont munis de la loi de composition $\circ$ définie par :

Définition 15 (composition de graphes) *Soit $g_1 = (\bigcup_{i=0..n} V_i, E)$ et $g_2 = (\bigcup_{i=0..m} V_i, E')$ deux graphes d'entrées-sorties sur V. Le graphe $g_1 \circ g_2 = (\bigcup_{i=0..n+m} V_i, E'')$ est tel que $E'' = E \bigcup_{(v_i^k, p, \#, v_j^l) \in E'} \{(v_i^{k+n}, p, \#, v_j^{l+n}\}$.*

Exemple 7 *La figure 3.1 représente deux graphes $G(f_1)$, $G(f_2)$, ainsi que leur composé $G(f_1 \cdot f_2) = G(f_1) \circ G(f_2)$.*

Contraintes de graphe $\Theta(g)$

Soit $\mathcal{X} = \bigcup_{i=0}^{\infty} X_i$ un ensemble de variables, où les X_i ont autant de composantes que les V_i. La fonction χ définie sur $\mathcal{V}$ à valeurs dans $\mathcal{X}$ associe à chaque sommet v_i^j le nom de variable x_i^j.

Définition 16 *La contrainte associée à un graphe d'entrées-sorties $g = (W, E)$ est la formule de variable libre X, X' définie par :*

$$\Theta(g) \stackrel{def}{=} \exists X_0 \dots X_{|g|}, \left\{ \begin{array}{l} X = X_0 \wedge X' = X_{|g|} \\ \bigwedge_{e=(s_1,p,\#,s_2)\in E} \chi(s_1) \# \chi(s_2) + p \end{array} \right.$$

On dira que le graphe g est satisfaisable si $\Theta(g) \not\models\!\mid \perp$.

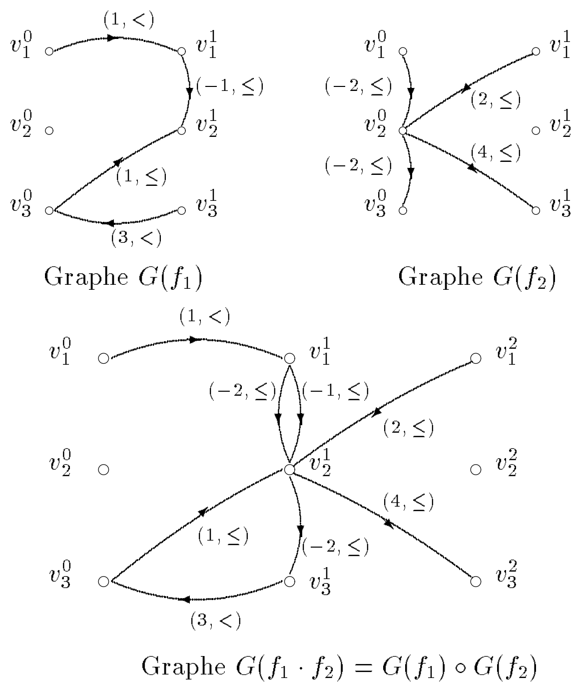


FIG. 3.1: *Composition de graphes d'entrées-sorties*

Liens entre graphes et contraintes sémantiques

Définition 17 *Soit $f \in \mathcal{F}_c(X \cup X')$. Le graphe $G(f) = (V_0 \cup V_1, E)$ est tel que E est le plus petit ensemble qui pour chaque contrainte atomique de f de la forme* $\begin{bmatrix} x_i \# x_j + c \\ x'_i \# x'_j + c \\ x'_i \# x_j + c \\ x_i \# x'_j + c \end{bmatrix}$ *contienne respectivement un arc* $\begin{bmatrix} (v_i^0, c, \#, v_j^0) \\ (v_i^1, c, \#, v_j^1) \\ (v_i^1, c, \#, v_j^0) \\ (v_i^0, c, \#, v_j^1) \end{bmatrix}$

Définition 18 *Soient $f_1 \ldots f_n$ dans $\mathcal{F}_c(X, X')$, le graphe $G(f_1 \cdots f_n)$ se définit par récurrence sur n par $G(f_1 \cdots f_n) = G(f_1 \cdots f_{n-1}) \circ G(f_n)$.*

Exemple 8 *On considère les deux formules de variables libres X et X', avec $X = \{x_1, x_2, x_3\}$, interprétées sur les réels :*

$$f_1 \stackrel{def}{=} \begin{cases} x_1 < x'_1 + 1 \\ x'_1 \leq x'_2 - 1 \\ x_3 \leq x'_2 + 1 \\ x'_3 < x_3 + 3 \end{cases} \quad et\ f_2 \stackrel{def}{=} \begin{cases} x_1 \leq x_2 - 2 \\ x'_1 \leq x_2 + 2 \\ x_2 \leq x'_3 + 4 \\ x_2 < x_3 - 2 \end{cases}$$

Les graphes $G(f_1)$, $G(f_2)$, et $G(f_1 \cdot f_2)$ sont ceux de la figure 3.1

Remarque 10 (Équivalence contraintes de graphes - contraintes sémantiques) *Soit A un automate à compteurs d'alphabet Σ et de sémantique Φ. Le jeu de réécriture définit par Φ, G et Θ est tel que :*

$$\forall a_1 \ldots a_n \in \Sigma, \Phi(a_1 \cdots a_n) =_{syntaxique} \Theta(G(\Phi(a_1) \cdots \Phi(a_n)))$$

3.2.2 Résolution des contraintes $\Theta(g)$

Le résultat principal de cette section, le lemme 3 permet de montrer une correspondance entre les contraintes et les chemins dans les graphes de $\mathcal{G}$.

Sur l'ensemble des chemins d'un graphe $g = (W, E)$, on définit les ensembles et opérations suivantes :

Définition 19 *Les fonctions source et dest de l'ensemble des chemins d'un graphe dans ses sommets qui désignent respectivement le sommet de départ et d'arrivée d'un chemin.*

Définition 20 *$\Gamma_g(W)$ est l'ensemble des chemins dans g dont les extrémités sont des sommets dans W.*

On étend aussi les fonctions *poids* et *ope* de sorte que :

Définition 21 *Le poids d ún chemin est la somme des poids de ses arêtes.*

L'opération associée à un chemin $\gamma \cdot e$ est : $ope(\gamma \cdot e) = ope(\gamma)$ si $ope(e) = \leq$ et $<$ sinon. L'opération associée au chemin vide étant $\leq$

On montre ici deux lemmes d'élimination, qui fournissent des formules équivalentes à $\Theta(g)$.

Lemme 3 *Soit $g = (W, E)$ un graphe d'entrées-sorties, on a :*

$\Theta(g) \models\!\!\!\mid \bot$ ssi $\exists \sigma$ cycle dans g tel que $\neg(0 \# poids(\sigma))$ avec $\#_\sigma = ope(\sigma)$

Si $\Theta(g) \not\models\!\!\!\mid \bot$ alors :

$$\Theta(g) \models\!\!\!\mid \exists X_0, X_{|g|}, X = X_0, X' = X_{|g|} \bigwedge_{\gamma \in \Gamma(V_0 \cup V_{|g|})} \chi(source(\gamma)) \#_\gamma \chi(dest(\gamma)) + poids(\gamma)$$

Preuve : Cette démonstration repose sur une "élimination" des arcs intermédiaires par analogie aux lemmes 1 et 2 d'élimination sur les variables.

Soit $W = \bigcup_{0..n} V_i$ un ensemble de sommets fixés avec $n = |g|$. Sur l'ensemble des graphes de sommets W, on raisonne par récurrence sur l'ordre lexicographique (p, q) sur les sommets v_q^p de $\bigcup_{i=1..n-1} V_i$. On montre que

$$\Theta(g) \models\!\!\!\mid HR(g) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \bigwedge_\sigma \text{ cycle dans }_g (0 \#_\sigma poids(\sigma)) \\ \exists X_0, \exists X_n, X = X_0, X' = X_n \\ \bigwedge_{\gamma \in \Gamma_g(V_0 \cup V_{|g|})} \chi(source(\gamma)) \#_\gamma \chi(dest(\gamma)) + poids(\gamma) \end{array} \right.$$

En effet, soit s le dernier sommet de $\bigcup_{i=1..n-1} V_i$ qui soit aussi une extrémité d'une arête de E, E se partitionne alors en $E_1 \uplus E_2 \uplus E_3$ ou E_2 est l'ensemble des arcs de g qui passent exactement une fois par s, et E_3 est l'ensemble des arêtes de g dont les extrémités sont s. Soit $x_q^p = \chi(s)$. Par définition on a :

$$\Theta(g) \models\!\mid \begin{cases} \exists X_0 \ldots \exists X_{q-1} \exists x_q^1 \ldots \exists x_q^{p-1} \exists x_q^{p+1} \ldots \exists X_n, X = X_0 \wedge X' = X_n \\ \bigwedge_{e \in E_1} (\chi(source(e)) \ \#_e \ \chi(dest(e)) + poids(e)) \exists x_m^p \\ \bigwedge_{e \in E_2} (\chi(source(e)) \ \#_e \ \chi(dest(e)) + poids(e)) \\ \bigwedge_{e \in E_3} (0 \ \#_e \ poids(e)) \end{cases}$$

On peut alors appliquer les lemmes 1 ou 2 pour éliminer la variable x_q^p. Il est aisé de vérifier que l'on a :

$$\Theta(g) \models\!\mid \Theta(g') \bigwedge_{e \in E_3} (0 \ \# \ poids(e))$$

où g' est le graphe (W, E') de mêmes sommets que g, et où

$$E' = E_1 \bigcup_{(s_1,p_1,\#_1,s),(s,p_2,\#_2,s_2) \in E_2} \{(s_1, p_1 + p_2, \#_3, s_2)\}$$

où $\#_3$ est le plus strict des opérateurs $\#_1$ et $\#_2$. Alors au vu de la définition de E', aucune arête n'a d'extrémités sur un sommet $v_{q'}^{p'}$ pour $(p', q') \geq_{lex} (p, q)$, et l'hypothèse de récurrence s'applique sur g', on a donc :

$$\Theta(g) \models\!\mid \begin{cases} \begin{cases} \bigwedge_{\sigma \text{ cycle dans } g'} (0 \ \# \ poids(\sigma)) \\ \bigwedge_{e \in E_3} (0 \ \# \ poids(e)) \end{cases} \\ \begin{cases} \exists X_0, X_n, X = X_0, X' = X_n \\ \bigwedge_{\gamma \in \Gamma_{g'}(V_0 \cup V_{|g|})} \chi(source(\gamma)) \ \# \ \chi(dest(\gamma)) + poids(\gamma) \end{cases} \end{cases}$$

On montre alors que $\theta(g) \models\!\mid HR(g)$:

On distingue parmi les chemins et les cycles de g ceux qui passent ou non par s. Ceux qui ne passent pas par s sont également des chemins de g'. Pour un chemin γ de g qui passe par une arête e de E_3, on a soit $\neg(0 \ \#_e \ poids(e))$ soit le chemin γ' obtenu à partir de γ en lui ôtant les arcs de E_3 est tel que $\chi(source(\gamma')) \leq \chi(dest(\gamma')) \Rightarrow \chi(source(\gamma)) \leq \chi(dest(\gamma))$. On peut donc se concentrer sur les chemins de g sans arc dans E_3. Pour les chemins qui contiennent des facteurs $e_1 \cdot e_2$ avec $e_1, e_2 \in E_2$, on peut les remplacer par des arêtes de E' de même poids et opérateurs. C'est bien que $\Theta(g) \models HR(g)$.

Réciproquement on distingue parmi les chemins et les cycles de g' ceux qui ne passent pas par des arêtes de $E' \setminus E_1$, qui sont déjà dans g, et ceux qui passent par des arêtes de $E' \setminus E_1$. Pour ceux là, il existe un chemin dans g qui induit la même contrainte : il s'obtient en remplaçant chaque arête e de $E' \setminus E_1$ par une paire d'arêtes $e_1 \cdot e_2$, respectivement de destination et de source s, et tel que la source de e_1 et la destination de e_2 soient celles de e. Par construction de E' il existe une telle paire d'arêtes de E qui vérifie $poids(e_1 \cdot e_2) = poids(e)$, $ope(e_1 \cdot e_2) = ope(e)$. On a donc également $HR(g) \models \Theta(g)$.

L'essentiel de la preuve est alors démontré. En effet, s'il existe un cycle σ qui vérifie $\neg(0 \#_\sigma poids(\sigma))$ alors $\Theta(g) \models\!\mid \bot$. Donc si l'on sait que $\Theta(g) \not\models\!\mid \bot$, c'est que $\bigwedge_{\sigma \text{ cycle dans } g}(0 \#_\sigma poids(\sigma)) \models\!\mid True$, et la seconde partie du lemme est vérifiée.

Si l'on s'intéresse à la satisfaisabilité, il reste à éliminer de $HR(g)$ les variables X_0, X_n et X, X' quantifiées existentiellement. Dans ce cas la démarche reste la même, et si l'on poursuit l'élimination il ne reste que les contraintes sur les cycles. Nous le montrons ici en utilisant un simple renommage. Par définition de $\Theta(g)$ on peut écrire $\exists X, \exists X', \Theta(g)$ sous la forme $\exists X, \exists X', \exists X_0, \exists X_n, X = X_0 \wedge X' = X_n \wedge \psi(X_0, X_n)$. Puisque X et X' sont non libres dans ψ, on a même :

$$\exists X, \exists X', \Theta(g) \models\!\mid \exists X, \exists X', \exists X_0, \exists X_n, \psi(X_0, X_n)$$

ou encore :

$$\exists X, \exists X', \Theta(g) \models\!\mid \exists X, \exists X', \exists X_{-1} \exists X_{n+1} \exists X_0, \exists X_n, \psi(X_0, X_n)$$

C'est à dire, à un renommage près $\exists X, \exists X', \Theta(g) \models\!\mid \exists X, \exists X', \Theta(g')$ pour un graphe g' de sommets $\cup_{i=0..n+2} V_i$ et d'ensemble d'arêtes E' qui ne contient aucun arc d'extrémités dans $V_0 \cup V_{n+2}$. g' est obtenu à partir de g en translatant les noms des sommets de v_i^j a v_i^{j+1}, et en propageant cette translation aux arêtes. Ainsi on a :

$$\theta(g') \models\!\mid \exists X_0, \exists X_{n+2}, X = X_0 \wedge X' = X_{n+2} \bigwedge_{\sigma \text{ cycle de } g'} (0 \# poids(\sigma))$$

Et puisque les cycles de g' sont des translatés de cycles de g on $\exists X, \exists X', \Theta(g) \models\!\mid \bot$ ssi $\exists \sigma$ cycle de g avec $\neg(0 \# poids(\sigma))$.
□

Exemple 9 *L'application du lemme 3 sur le graphe $G(f_1 \cdot f_2)$ de la figure 3.1 donne :*

$$\Theta(G(f_1 \cdot f_2)) \models\!\mid \begin{cases} \exists X_0, \exists X_2, X_0 = X \wedge X_2 = X' \\ \wedge \begin{cases} x_1^0 < x_3^0 + 0 \wedge x_1^0 < x_3^0 + 1 \\ x_1^0 < x_3^2 + 3 \wedge x_1^0 < x_3^2 + 4 \\ x_3^0 < x_3^0 + 2 \wedge x_1^2 \le x_3^2 + 6 \wedge x_1^2 \le x_3^0 + 3 \end{cases} \\ \wedge 0 < 2 \wedge 0 < 2 \end{cases}$$

Le lemme 3 fait apparaître que seuls les chemins entre les extrémités d'un graphe g sont significatifs pour exprimer $\Theta(g)$, dès lors que $\Theta(g)$ est satisfaisable. On peut encore raffiner cette remarque, en considérant les chemins de poids minimaux :

Lemme 4 *Soit g un graphe satisfaisable de sommets $\bigcup_{i=0..n} V_i$ et Γ un sous ensemble de $\Gamma_g(V_0 \cup V_n)$ tel que pour chaque $\gamma \in \Gamma_g(V_0 \cup V_n)$ il existe un chemin μ dans Γ de mêmes extrémités, qui vérifie $poids(\mu) \le poids(\gamma)$, et si $poids(\gamma) = poids(\mu)$ alors on a aussi $ope(\gamma) = ope(\mu)$. Alors :*

$$\Theta(g) \models\!\mid \exists X_0 \exists X_n, \left\{ \begin{array}{l} X_0 = X \wedge X_n = X' \\ \bigwedge_{\gamma \in \Gamma} \chi(source(\gamma)) \ \#_\gamma \ \chi(dest(\gamma)) + poids(\gamma) \end{array} \right.$$

Exemple 10 *Ainsi sur la figure 3.1 le lemme 4 montre que :*

$$\Theta(G(f_1 \cdot f_2)) \models\!\mid \left\{ \begin{array}{l} \exists X_0, \exists X_2, X_0 = X \wedge X_2 = X' \\ \wedge \left\{ \begin{array}{l} x_1^0 < x_3^0 + 0 \wedge x_1^0 < x_3^2 + 3 \\ x_3^0 < x_3^0 + 2 \wedge x_1^2 \leq x_3^2 + 6 \\ \wedge x_1^2 \leq x_3^0 + 3 \end{array} \right. \end{array} \right.$$

3.3 Les Tgraphes

Nous avons vu dans le chapitre 2 qu'une classe particulière d'automates à compteurs était associée aux automates temporisés. La modélisation correspondante sous forme de graphes se fera à l'aide des Tgraphes qui possèdent des propriétés propres.

On distingue le nom d'un des sommets sur ces Tgraphes qui joue un rôle particulier : on pose $V = \{v_1, \ldots, v_n, \tau\}$ où τ est ce sommet distingué, et V_i se note $\{v_1^i, \ldots, v_n^i, \tau^i\}$, $\mathcal{V} = \bigcup_i V_i$. Le projeté d'un sommet v_i^j est le sommet v_i, et celui d'un τ^j est τ.

De la même façon on distingue une suite de variables τ^i dans l'ensemble $\mathcal{X} = \bigcup X_i$, chaque τ^i étant dans X^i. La fonction χ de V_i dans X_i associe à chaque sommet v_i^j la variable x_i^j, et au sommet τ^i la variable τ^i. (La propriété 10 de correspondance des contraintes de graphes et des contraintes sémantique justifie cet abus de notation).

Définition 22 *Un 1-Tgraphe(K, W) est un graphe d'entrées-sorties $G(f)$ pour une formule f de la forme :*

$$f = \left\{ \begin{array}{l} \tau \leq \tau' \\ \bigwedge_i (x_i' \leq \tau' \wedge x_i \leq \tau) \\ \bigwedge_{i \in I_1} (x_i' = x_i) \bigwedge_{i \notin I_1} (x_i' = \tau) \\ \bigwedge_i (x_i \ \#_1 \ \tau + a_1 \wedge x_i' \ \#_2 \ \tau' + a_2) \\ \bigwedge_i (\tau \ \#_3 \ x_i + a_3 \wedge \tau' \ \#_4 \ x_i + a_4) \end{array} \right.$$

Dont la fonction poids est à valeur dans $[-K, K]$. Un Tgraphe(K, V) est un composé par $\circ$ de 1-Tgraphe(K, V).

La propriété suivante précise l'ossature que forme l'ensemble $\bigcup_i \tau^i$ dans les Tgraphes. Elle est représentée par la figure 3.2, il s'agit d'une simple relecture de la définition des Tgraphes.

Propriété 11 *Soient u, v deux Tgraphes(K, V). Pour tout $s \in V_{|u|}$ et $i \in [|u|, |u| + |v|$ il existe un chemin γ dans $u \circ v$ de s a τ^i, qui vérifie $poids(\gamma) = 0$ et $ope(\gamma) =' \leq'$, dont aucune arête n'est dans u.*

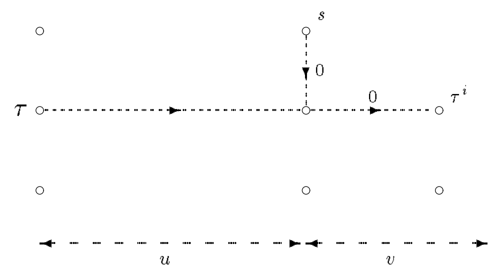


FIG. 3.2: *Chemin implicite*

La contrainte sur les formules génératrices d'un Tgraphe, qui assure que deux variables ne sont comparées que si l'une des deux est τ ou τ', permet de montrer la propriété :

Propriété 12 *Tout chemin d'un Tgraphe(K, V) entre deux sommets v_i^k et v_j^l pour $i \neq j$ passe par un sommet τ^m.*

Il est également aisé de vérifier que :

Propriété 13 *Tout chemin d'un Tgraphe(K, V) entre deux sommets v_i^j et v_i^k, qui ne passe pas par un sommet de $\bigcup_l \tau^l$, est de poids nul et d'opérateur $' \leq '$.*

Nous présentons encore ici deux propriétés des Tgraphes qui précisent des conditions sur les poids des chemins, encadrés par la satisfaisabilité des Tgraphes et la recherche de représentants de poids minimaux pour les chemins.

Propriété 14 *Soient u, v deux Tgraphes, γ et $\mu = \mu_0 \cdot \mu_1$ des chemins de $u \circ v$ tels que γ est d'extrémités dans u, μ est un chemin dont aucune arête n'est dans u et tel que μ_1 contienne exactement un sommet de $\bigcup \tau^k$. (Voir figure 3.3). Si $\mu \cdot \gamma$ est un chemin dans $u \circ v$, alors on a $poids(\mu_1 \cdot \gamma) < 0 \Rightarrow u \circ v$ insatisfaisable et donc $u \circ v$ satisfaisable $\Rightarrow poids(\gamma) \geq -K$.*

Preuve :

Soit τ^i le dernier passage dans $\bigcup \tau^k$ de μ et x, y respectivement les sommets source et destination de γ. Le chemin μ_1 est un chemin de τ^i à x. Par définition de μ_1, et par la propriété 13 le poids de μ_1 est celui de sa première arête, et donc $poids(\mu_1) \in [0, K]$. D'après la propriété 11, il existe aussi un chemin ν de poids nul entre y et τ^i. On a alors un cycle $\gamma \cdot \nu \cdot \mu_1$ dans $u \circ v$. Puisque $u \circ v$ est supposé satisfaisable, le lemme 3 assure que ce poids est positif. Avoir $poids(\gamma) < -K$ serait donc absurde.

□

Propriété 15 *Soient u, v deux Tgraphes, γ un chemin dans $u \circ v$ d'extrémités dans u et μ un chemin dont aucune arête n'est dans u et qui contienne un sommet de $\bigcup \tau^k$. Si $\gamma \cdot \mu$ est un chemin dans $u \circ v$ (voir figure 3.4), alors on a $poids(\gamma) \leq K \vee \exists \eta$ de mêmes extrémités avec $poids(\eta) < poids(\gamma \cdot \mu)$.*

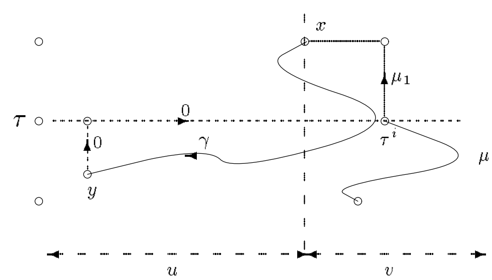


FIG. 3.3: $poids(\gamma) \not< -K$

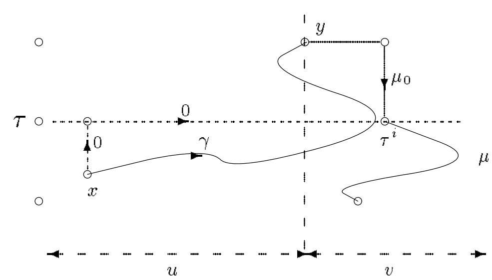


FIG. 3.4: $poids(\gamma) \not> -K$

Preuve :

Soit x le sommet source de γ et τ^i le premier passage de μ dans $\bigcup \tau^k$. On peut décomposer μ en $\mu_0 \cdot \mu_1$ de sorte que μ_0 soit le plus cours préfixe de destination τ^i. La propriété 13 assure que $poids(\mu_0) \in [-K, 0]$. La propriété 11 affirme qu'il existe un chemin μ'_0 de poids nul de x à τ^i. Si $poids(\gamma) > K$, alors on a $poids(\mu'_0) < poids(\gamma \cdot \mu_0)$ et le chemin $\eta = \mu'_0 \cdot \mu_1$ est de poids inférieur à celui de $\gamma \cdot \mu$.

$\square$

Chapitre 4

Relation Binaire d'Accessibilité pour les Automates à Compteurs Plats

Ce chapitre vise à présenter un résultat d'expressibilité de la relation binaire d'accessibilité pour les automates à compteurs plats. Il s'obtient en composant les expressions de relations binaires pour des relations $\Theta(g^n)$ sur des graphes d'entrées-sorties. On s'attache donc dans un premier temps à résoudre un problème de graphes, on montrera en conclusion de ce chapitre comment résoudre alors le problème sur les automates à compteurs.

Le problème intermédiaire que nous considérons est le suivant :

> Soit g un graphe d'entrées-sorties sur V de taille 1,
> comment exprimer la relation ternaire $\Theta(g^n)$
> de variables libres X, n, X' dans $\mathcal{T}h$?

La première difficulté provient de la nature de la définition de $\Theta(g)$:

$$\Theta(g) \stackrel{\text{def}}{=} \exists X_0 \dots \exists X_{|g|}, \begin{cases} X = X_0 \wedge X' = X_{|g|} \\ \bigwedge_{e=(s_1,s_2)\in E} \chi(s_1) \# \chi(s_2) + poids(e) \text{ où } \# = ope(e) \end{cases}$$

Celle-ci fait intervenir n quantifications, et dés lors que n est un paramètre, $\Theta(g^n)$ n'est pas du premier ordre. Le lemme 3 déplace la difficulté puisque dans le cas où $\Theta(g^n)$ est satisfaisable, il donne une formule équivalente qui ne contient plus que deux quantifications. Mais il réduit le problème à une caractérisation des chemins dans g^n, qui reste un graphe paramétré par n.

Nous allons introduire un graphe $K(g)$, replié de la famille des graphes g^n, qui sera indépendant de n et pour lequel, dans une certaine mesure, il y aura correspondance entre chemins de g^n et chemins de $K(g)$.

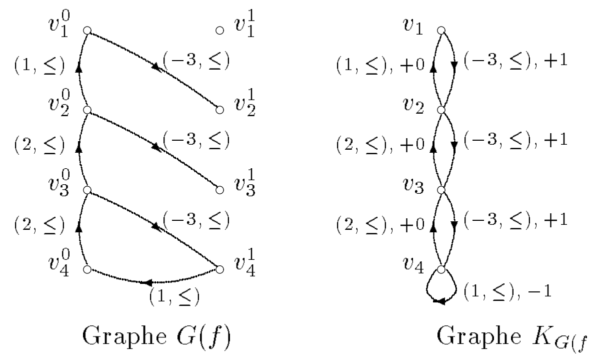


FIG. 4.1: *Graphes replié - Exemple*

L'étude de $K(g)$ permet en particulier de résoudre le problème de la satisfaisabilité de la famille des g^n, qui nous occupera dans un premier temps. Puis d'exprimer dans $\mathcal{T}h$ un ensemble représentatif des chemins de g^n qui permet d'appliquer le théorème 4 pour exprimer $\Theta(g^n)$ linéairement en fonction de n.

4.1 Graphe replié

Définition 23 *Soit $g = (W, E)$ un graphe d'entrées-sorties sur V. Le graphe replié de g est $K(g) = (V, E')$ avec $E' \subset V \times \mathbb{Z} \times \{'<', \leq\} \times \mathbb{Z} \times V$ pour ensemble d'aretes, sur lequel on définit respectivement pour chacune des composantes les projections sources, poids, ope, prog et dest. E' est défini par $E' = \bigcup_{(v_i^j, p, \#, v_k^l) \in E} \{(v_i, p, \#, l - j, v_k)\}$.*

L'arête correspondante à $e = (v_i^j, p, \#, v_k^l)$ est aussi appelée son arête projetée. On la note $proj(e)$. Cette fonction de projection s'étend naturellement aux chemins, le projeté d'un chemin étant le chemin constitué des projetés des arêtes.

On étend aussi la fonction de progression aux chemins, la progression d'un chemin étant la somme des progressions de ses arêtes.

On peut remarquer que pour tout n, $K(g) = K(g^n)$.

Exemple 11 *Les graphes $G(f)$ et $K(G(f))$ associés à la formule*

$$f = \begin{cases} x_1 \leq x_2' - 3 \wedge x_2 \leq x_1 + 1 \\ x_2 \leq x_3' - 3 \wedge x_3 \leq x_2 + 2 \\ x_3 \leq x_4' - 3 \wedge x_4 \leq x_3 + 2 \\ x_4' \leq x_4 + 1 \end{cases} \quad \textit{sont représentés par la figure4.1}$$

Définition 24 *Soit g un graphe d'entrées-sorties, on définit une application $Expand_n(\gamma, p)$ qui associe lorsque c'est possible, à un chemin γ de $K(g)$ et à* *un entier* p*, un chemin dans* g^n*. Cette application se définit par récurrence sur la taille du chemin* γ*.*

- *Lorsque* γ *est réduit à un arc* e *de* $K(g)$ *d'un sommet* v_i *à un sommet* v_j *:* $Expand_n(e,p)$ *est l'arc qui relie* v_i^p *à* $v_j^{p+prog(e)}$ *de mêmes poids et opérateur que* e*.*
- $Expand_n(\gamma \cdot e, p) = Expand_n(\gamma, p) \cdot Expand_n(e, p + prog(\gamma))$

Les fonctions de projection et d'expansion sont inverses l'une de l'autre :

Propriété 16 *Soit* γ *un chemin dans* g^n *de source dans* V_i*, on a:*

$$Expand_n(proj(\gamma), i) = \gamma$$

Propriété 17 *Pour tout chemin* γ *dans* $K(g)$*, il existe* $n, p \in \mathbb{N}$ *tels que* $Expand_n(\gamma, p)$ *soit défini et* $proj(Expand_n(\gamma, p)) = \gamma$*.*

Si g *est un graphe d'entrées-sorties de longueur 1, on peut en particulier prendre* $p = |\gamma| + 1$ *et* $n = 2 * p$*. Dans le cas particulier où* γ *est un cycle tel que* $prog(\gamma) = 0$*, il existe une permutation* γ' *de* γ *telle que* $Expand_n(\gamma', 1)$ *existe avec* $n = \lfloor |\gamma|/2 \rceil + 2$*.*

Preuve :

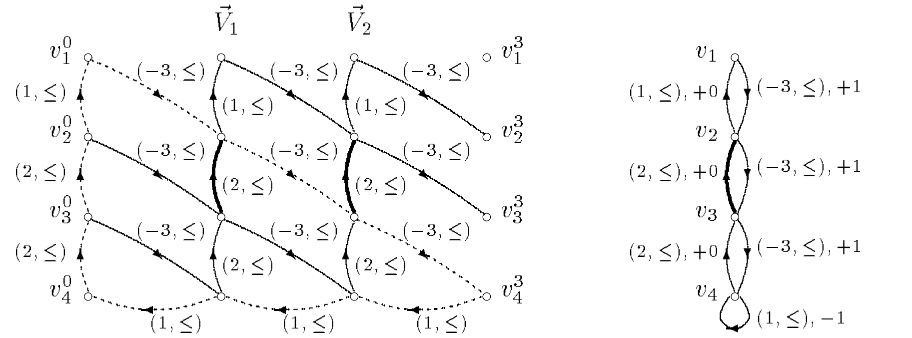
$Expand_n$ n'est définie à chaque étape de manière certaine que si les indices i et $i + prog(e)$ sont dans $[1..n-1]$. En effet, il faut d'une part que chaque sommet soit dans $\cup V_i$, mais aussi que l'arc soit défini dans g^n. Or le repliage de g en $K(g)$ a associé une progression nulle à la fois aux arcs entre sommets de V_0, et à ceux entre sommets de V_1. L'expansion est alors soumise à un effet de bord lorsqu'elle prend pour valeur un sommet de V_0 ou de V_n. Il faut donc ajouter une marge de sécurité en bloquant l'accès à ces sommets. Dans le cas des graphes de longueur 1, une borne triviale pour l'amplitude de la progression d'un chemin γ est $|\gamma|$. Il suffit de prendre $n = 2 * |\gamma| + 2$, et $p = |\gamma| + 1$ pour que $Expand_n(\gamma, p)$ soit bien un chemin dans g^n. $\square$

Exemple 12 *Les arêtes en gras de* $G(f \cdot f \cdot f)$ *de la figure 4.2 sont des expansions de l'arête en gras de* $K(G(f))$*.*

On s'intéresse à présent à la caractérisation dans $\mathcal{T}h$ des poids et des opérateurs des chemins de $K(g)$. Dans un premier temps, si l'on associe aux chemins le nombre d'occurrence de leurs arêtes, on peut en déduire immédiatement leurs poids et leurs opérateurs par une formule de $\mathcal{T}h$.

Réciproquement on peut définir une formule sur les vecteurs d'occurrence d'arêtes qui est satisfaite ssi il existe un chemin constitué exactement de ces arêtes. Nous allons commencer par établir ce résultat pour tout graphe.

Soit $G = (V, E)$ un graphe quelconque. On dénote les multi-ensembles d'arêtes de G par des vecteurs $\vec{M} = (m_1 \ldots m_{|E|}) \subset \mathbb{N}^{|E|}$ tels que chaque m_i corresponde à un nombre d'arêtes $e_i \subset E$. Le vecteur d'occurrence d'arêtes d'un



Graphe $G(f\cdot f\cdot f)$ Graphe $K(G(f))$

FIG. 4.2: *Projections Expansions - Exemple*

chemin γ est noté $\vec{M}(\gamma)$. On utilise les fonctions et prédicats suivants définissables dans $\mathcal{T}h$:

$Entrant_v(\vec{M}) \stackrel{\text{def}}{=} \Sigma_{e_i \text{ de destination } v}\, m_i$, qui associe à chaque sommet v le nombre d'arcs de $\vec{M}$ de destination v.

$Sortant_v(\vec{M}) \stackrel{\text{def}}{=} \Sigma_{e_i \text{ de source } v}\, m_i$, qui associe à chaque sommet v le nombre d'arcs de $\vec{M}$ de source v

$Connexe_v(\vec{M})$ qui est vérifié ssi pour toute arête e dans $\vec{M}$, il existe un chemin constitué d'arêtes de $\vec{M}$ qui soit de source v et qui termine par e.

Pour définir le prédicat $Connexe_v$ on utilise l'ensemble des chemins simples entre v et v' : $Chemin\_simple_{v,v'}$ est l'ensemble des vecteurs d'occurrence d'arêtes des chemins sans cycle de G entre les sommets v et v'. Ces ensembles sont finis, on peut donc définir :

$$Connexe_v(\vec{M}) \stackrel{\text{def}}{=} \bigwedge_{e_i \text{ de destination } v_i} m_i > 0 \Rightarrow \left(\bigvee_{\vec{M}' \in Chemin\_simple(v,v_i)} \bigwedge_j m_j \geq m'_j\right)$$

Définition 25 *Le prédicat $Chemin_{v,v'}(\vec{M})$ est défini par :*

$$Chemin_{v,v'}(\vec{M}) \stackrel{def}{=} \begin{cases} \bigwedge_i Entier(m_i) \\ \bigwedge Connexe_v(\vec{M}) \\ \\ \bigwedge_{v'' \in V \setminus \{v,v'\}} Entrant_{v''}(\vec{M}) = Sortant_{v''}(\vec{M}) \\ \\ \left(\begin{array}{l} v = v' \wedge Entrant_v(\vec{M}) = Sortant_v(\vec{M}) \\ \vee \\ v \neq v' \wedge \begin{cases} Sortant_v(\vec{M}) = Entrant_v(\vec{M}) + 1 \\ Entrant_{v'}(\vec{M}) = Sortant_v(\vec{M}) + 1 \end{cases} \end{array} \right) \end{cases}$$

Lemme 5 *$\vec{M} \models Chemin_{v,v'}$ ssi il existe un chemin γ de G d'extrémités v et v' dont $\vec{M}$ est le vecteur d'occurrence d'arêtes.*

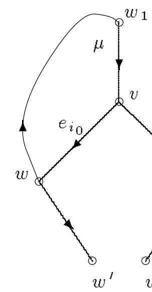


FIG. 4.3: $\vec{M}' \models Connexe_w$

Preuve :

On peut vérifier directement que le multi-ensemble $\vec{M}$ des arêtes d'un chemin qui va de v à v' vérifie $Chemin_{v,v'}$.

Pour la réciproque, on raisonne par récurrence sur la somme des m_i. Si elle est réduite à 1, le chemin constitué de cette seule arête convient. Si $Sortant_v(\vec{M}) = 1$ on considère e_{i_0} l'arête de source v pour laquelle $m_{i_0} = 1$, et sa destination w. On peut vérifier sans difficulté que $\vec{M}' \models Chemin_{w,v'}$ est vérifié avec $\forall i \neq i_0, m'_i = m_i$ et $m'_{i_0} = 0$. Par hypothèse de récurrence il existe alors un chemin γ de w à v' avec $\vec{M}(\gamma) = \vec{M}'$, et donc $e_{i_0} \cdot \gamma$ est un chemin de v à v' avec $\vec{M}(e_{i_0} \cdot \gamma) = \vec{M}$.

Si $Sortant_v(\vec{M}) \geq 2$ alors $Entrant_v(\vec{M}) \geq 1$. On peut suivre la preuve sur la figure 4.3. Soit w_1 un sommet tel qu'il existe un arc e_i de w_1 à v avec $m_i > 0$. Puisque l'on a $connexe_v(\vec{M})$, il existe un chemin formé d'arêtes de $\vec{M}$ de v à w_1. On le note $e_{i_0} \cdot \mu$, où e_{i_0} est sa première arête, soit w le sommet destination de e_{i_0}. On définit $\vec{M}'$ par $\forall i \neq i_0, m'_i = m_i$ et $m'_{i_0} = m_{i_0} - 1$. On montre que l'on a alors $\vec{M}' \models Chemin_{w,v'}$. Les relations sur le nombre d'arcs entrants et sortants sont sans difficulté. Pour montrer que l'on a $\vec{M}' \models Connexe_w$ on considère un sommet w' quelconque pour lequel $Entrant_{w'}(\vec{M}') > 0$. Alors $Entrant_{w'}(\vec{M}) > 0$ et on peut utiliser que $\vec{M} \models Connexe_v$: il existe un chemin γ sans cycle dont les arêtes sont dans $\vec{M}$ qui relie v à w'.

Si γ passe par e_1, puisqu'il est sans cycle alors e_1 est sa première arête, et on peut écrire $\gamma = e_1 \cdot \gamma'$, où γ' est un chemin de w à w' dont les arêtes sont dans $\vec{M}'$.

Si γ ne passe pas par e_1, alors on peut extraire de $\mu \cdot \gamma$, qui relie w à w', un sous chemin de mêmes extrémités, sans cycle, dont les arêtes sont dans $\vec{M}'$.

On a donc $\vec{M}' \models Chemin_{w,v'}$. Par hypothèse de récurrence il existe un chemin δ de w à v' avec $\vec{M}(\delta) = \vec{M}'$, et donc $e_{i_0} \cdot \delta$ est un chemin de v à v' avec $\vec{M}(e_{i_0} \cdot \gamma) = \vec{M}$.

□

Exemple 13 *Ici nous reprenons l'exemple de la figure 4.2, pour exprimer le prédicat* $Chemin_{v_2,v_3}(\vec{M})$ *de* $K(G(f))$*. Le multi-ensemble* $\vec{M} = (m_1, \ldots, m_7)$ *d'arêtes indique respectivement le nombre d'arêtes* $e_1 = (v_1, v_2), e_2 = (v_2, v_3), e_3 = (v_3, v_4), e_4 = (v_4, v_4), e_5 = (v_4, v_3), e_6 = (v_3, v_2), e_7 = (v_2, v_1)$*.*

On a :

$$Connexe_{v_2}(\vec{M}) \stackrel{def}{=} \begin{cases} (m_1 > 0 \vee m_7 > 0) \Rightarrow m_7 > 0 \\ (m_1 > 0 \vee m_2 > 0 \vee m_6 > 0 \vee m_7 > 0) \Rightarrow m_2 > 0 \\ (m_2 > 0 \vee m_3 > 0 \vee m_5 > 0 \vee m_6 > 0) \Rightarrow m_2 > 0 \\ (m_3 > 0 \vee m_4 > 0 \vee m_5 > 0) \Rightarrow m_2 > 0 \wedge m_3 > 0 \end{cases}$$

Et les contraintes sur le nombre d'arcs entrants et sortants :

$$m_1 = m_7 \wedge m_3 + m_4 = m_4 + m_5$$

Et:

$$m_2 + m_7 = m_1 + m_6 + 1 \wedge m_2 + m_5 = m_3 + m_6 + 1$$

4.2 Caractère borné de la famille des $\Theta(g^n)$

Nous avons vu lors du lemme 3 que la satisfaisabilité de $\Theta(g^n)$ est liée à l'existence de cycles qui induisent une contrainte insatisfaisable. Puisque les graphes g^n sont des sous graphes des g^m pour $m > n$, la non satisfaisabilité est un caractère monotone avec n de la suite des formules $\Theta(g^n)$.

D'autre part, nous avons vu (propriété 17 et 16) qu'à un chemin de g^n correspond un chemin de $K(g)$ et inversement dans une certaine mesure. Cela permet de décider si la relation $\Theta(g^*) = \bigcup_i^\infty \Theta(g^i)$ est une union finie de $\Theta(g^i)$, et dans ce cas de l'exprimer dans $\mathcal{T}h$

Comme nous l'avons fait pour les chemins, on peut étendre les fonctions $poids, prog$ et introduire le prédicat ope_strict sur les multi-ensembles d'arêtes de $K(g)$: on pose $poids(\vec{M}) \stackrel{\text{def}}{=} \Sigma_i m_i * poids(e_i)$, $prog(\vec{M}) \stackrel{\text{def}}{=} \Sigma_i m_i * prog(e_i)$ et $ope\_strict(\vec{M}) \stackrel{\text{def}}{=} \bigvee_{i, ope(e_i) = '<'} (m_i > 0)$.

Définition 26 *Soit* g *un graphe d'entrées-sorties sur* V *de longueur 1. On définit :*

$$Borne_g(n) \stackrel{def}{=} \begin{cases} Entier(n) \\ \exists \vec{M}, \bigwedge_i Entier(m_i) \end{cases} \wedge \bigvee_{v \in V} \begin{cases} Chemin_{v,v}(\vec{M}) \\ prog(\vec{M}) = 0 \\ ope\_strict(\vec{M}) \Rightarrow \neg(0 < poid(\vec{M})) \\ \neg ope\_strict(\vec{M}) \Rightarrow \neg(0 \leq poid(\vec{M})) \\ n \geq (\Sigma_i m_i)/2 + 2 \end{cases}$$

Lemme 6 *Soit* g *un graphe d'entrées-sorties sur* V *de longueur 1, on a :*

$$\begin{cases} n \models Borne \Rightarrow \Theta(g^*) = \bigvee_{i=0..n-1} \Theta(g^i) \\ \exists n, \Theta(g^n) \models\!\mid \bot \Rightarrow \exists n' \geq n, Borne(n') \end{cases}$$

Preuve :

En effet, si $n \models Borne_g$ on est dans le cadre de la propriété 17 il existe un cycle γ avec $\vec{M} = \vec{M}(\gamma)$ de longueur Σm_i tel que $Expand_n(\gamma', 1)$ existe.

Réciproquement si γ est un cycle de g^n qui rende $\Theta(g^n)$ insatisfaisable. Le multi-ensemble $\vec{M} = \vec{M}(proj(\gamma))$ vérifie toutes les conditions de $borne_g(n')$ dés lors que $n' \geq (\Sigma_i m_i)/2 + 2$.

□

Exemple 14 *Dans le cas de l'étude de la figure 4.2, Il existe un cycle de poids négatif sur v_1 et de progression nulle, il est représenté par son expansion en pointillés dans $G(f)^3$. Ce chemin a une longueur de 9, par la propriété 17 il existe une expansion de ce chemin dans $G(f)^6$, (en fait il en existe une ici dans $G(F)^3$).*

4.3 Représentants des chemins de g^n

On se restreint à présent à des graphes d'entrées-sorties tels que $\forall n, \neg Borne(n)$. On cherche alors à caractériser une famille de chemins représentatifs dans g^n, au sens du théorème 4, pour n fixé suffisamment grand. Ceux ci seront également des expansions de chemins très réguliers dans $K(g)$. Ce sont ces derniers que l'on pourra caractériser dans $\mathcal{T}h$, ce qui permettra alors d'exprimer $\Theta(g^n)$.

On introduit une relation d'ordre partiel sur les chemins des g^n :

Définition 27 *Sur les chemins des graphes g^n on définit l'ordre $\trianglelefteq$ tel que*

$\gamma \trianglelefteq \gamma'$ ssi $(poids(\gamma), ope(\gamma), |\gamma|) <_{lex} (poids(\gamma'), ope(\gamma'), |\gamma'|)$ (avec pour convention que "$<$" $<$ "$\leq$")

On note $\triangleleft$ l'ordre pris au sens large, et $\sim$ la relation réflexive de $\trianglelefteq$.

Définition 28 *Pour un chemin γ on note :*

$$\gamma \in [V_i, V_j] \text{ ssi les sommets de } \gamma \text{ sont dans } \bigcup_{k \in [i,j]} V_k$$

Nous allons montrer que pour les chemins contenus dans les parties centrales des graphes g^n, c'est à dire les chemins $\gamma \in [V_b, V_{n-b}]$ pour un certain b, il existe des chemins γ' plus petits pour $\trianglelefteq$ qui soient très réguliers, dans un sens que nous allons préciser.

Définition 29 *Sur le graphe replié $K(g)$ on définit les ensembles :*

- $Cycles_g^+(v) = \{$ *cycles élémentaires σ de $K(g)$ sur v tels que $prog(\sigma) > 0\}$*
- $Cycles_g^+ = \bigcup_v Cycles_g^+(v)$
- $Cycles_g^-(v) = \{$ *cycles élémentaire σ de $K(g)$ sur v tels que $prog(\sigma) < 0\}$*

– $Cycles_g^- = \bigcup_v Cycles_(^- v)$

– $Cycles_g = Cycles_g^+ \cup Cycles_g^-$

Ces cycles seront les constituants de base de la régularité recherchée. Ainsi dans un premier temps, sur un chemin suffisamment long, on peut exhiber un certain nombre de sous chemins dont les projections sont un même cycle.

Propriété 18 *$\forall\alpha, \forall n, \forall\gamma$ chemin dans g^n on a :*

$$|\gamma| \geq (|V|+1) * (||Cycles_g||+1) * \alpha_0$$

$$\Rightarrow \begin{cases} \exists\gamma_1, \exists\sigma_1, \dots, \exists\gamma_\alpha, \exists\sigma_\alpha, \exists\gamma_{\alpha+1}, \gamma = \gamma_1 \cdot \sigma_1 \cdots \gamma_\alpha \cdot \sigma_\alpha \cdot \gamma_{\alpha+1} \\ proj(\sigma_1) = \cdots = proj(\sigma_\alpha) \in Cycles_g \end{cases}$$

Preuve :

En effet, dans chaque sous chemin de longueur $|V|+1$ de γ, il existe au moins un sous chemin σ_1 dont la projection est dans $Cycles_g$. Si le chemin est suffisamment long, on peut faire apparaître deux cycles. De la même façon on peut en extraire un nombre α arbitraire si le chemin considéré est encore plus long. Ainsi on peut imposer que $|\gamma_1|$ ou $|\gamma_{\alpha+1}|$ soit inférieur à $(|V|+1) * (||Cycles_g||+1) * \alpha$.

□

Pour montrer qu'il existe des chemins réguliers minimaux pour $\trianglelefteq$, on cherche à regrouper ces cycles pour qu'ils apparaissent consécutivement. Cette opération est généralement possible grâce à la régularité des graphes g^n qui sont composés d'une juxtaposition de graphes g. Elle consiste en une translation des chemins. Cette opération nécessite toutefois quelques précautions pour s'assurer que l'image de la translation existe. C'est pourquoi nous ne l'effectuons que pour des chemins centraux.

Définition 30 *Soit γ un chemin de l'un des g^n de source dans V_i. Le translaté de γ de $k \in \mathbb{Z}$ noté $Translate_m(\gamma, k)$ est le chemin $Expand_m(proj(\gamma), i+k)$, s'il est défini.*

Propriété 19 *Soit g un graphe d'entrées-sorties de longueur 1, $\forall b, \forall n, \forall\gamma$ chemin de $g^n \in [V_b, V_{n-b}]$ qui relie $v_i^{n_1}$ à $v_j^{n_2}$, pour tout c tel que $|c| < b$, le chemin $\gamma' = Translate_n(\gamma, c)$ existe et relie $v_i^{n_1+c}$ à $v_j^{n_2+c}$. Ce chemin vérifie $poids(\gamma') = poids(\gamma)$,$ope(\gamma') = ope(\gamma)$, $|\gamma'| = |\gamma|$*

Propriété 20 *Soit g un graphe d'entrées-sorties de longueur 1, $\alpha \in \mathbb{N}$, et $b = \alpha * max\{|\sigma|, \sigma \in Cycles_g\}$. Pour tout chemin $\gamma = \sigma_1 \cdot \gamma_1 \cdot \sigma_2 \cdots \gamma_{\alpha-1} \cdot \sigma_\alpha$ de g^n dans $[V_b, V_{n-b}]$ tel que $\forall i, j, proj(\sigma_i) = proj(\sigma_j) \in Cycles_g$, on a :*

$$\exists\delta, \exists\sigma'_2 \dots \sigma'_\alpha, proj(\sigma'_i) = proj(\sigma_1)$$

avec

$$\sigma_1 \cdot \sigma'_2 \cdots \sigma'_\alpha \cdot \delta \sim \gamma \text{ est un chemin de } g^n$$

Preuve :

On applique la propriété 19 avec $\sigma'_i = Translate_n(\sigma_1, prog(proj(\sigma_1)) * (i-1))$, et $\gamma'_{\alpha-i} = Translate_n(\gamma_i, prog(proj(\sigma_1)) * i)$

□

Nous avons donc un premier résultat qui permet d'exhiber une régularité locale à un chemin. On peut encore aller plus loin, car si un chemin contient deux morceaux suffisamment longs, on peut leur appliquer chacun la propriété 20, c'est à dire qu'il existe deux chemins comparables à ces morceaux, et auxquels on peut associer deux cycles σ et θ de $K(g)$. Nous allons à présent préférer l'un de ces cycles à l'autre pour obtenir soit un chemin plus petit pour $\lhd$, soit un chemin *comparable* et plus régulier encore.

Pour comparer deux cycles σ, θ nous utiliserons les constantes $k(\sigma, \theta) = ppcm(|prog(\sigma)|, |prog(\theta)|)/|prog(\sigma)|$. Selon les signes relatifs de $prog(\sigma)$ et $prog(\theta)$, nous allons pouvoir ajouter ou retirer un nombre $k(\sigma, \theta)$ de cycles σ ou $k(\theta, \sigma)$ de cycles θ pour obtenir un nouveau chemin. Plus précisément :

Propriété 21 *Soit g un graphe d'entrées-sorties de longueur 1, $\alpha_0 = max\{k(\sigma,\theta)$ pour $\sigma, \theta \in Cycles_g\}$ et $b = \alpha_0 * max\{|\sigma|, \sigma \in Cycles_g\}$. Pour tout chemin $\gamma = \gamma_1 \cdot \gamma_2 \cdot \gamma_3 \in [V_b, V_{n-b}]$ tel que $proj(\gamma_1)$ et $proj(\gamma_3)$ soient de la forme σ^α et θ^β pour $\sigma, \theta \in Cycles_g$, et $\alpha, \beta \geq \alpha_0$. On pose $u = k(\sigma, \theta)$ et $v = k(\theta, \sigma)$, alors on peut construire les chemins suivants dans g^n de mêmes extrémités que γ :*

1. *Si $prog(\sigma)$ et $prog(\theta)$ sont de même signe :*

   (a) $\gamma'_1 \cdot Translate_n(\gamma_2, u * prog(\sigma)) \cdot \gamma'_3$ *avec* $\left\{ \begin{array}{l} prog(\gamma'_1) = \sigma^{\alpha+u} \\ proj(\gamma'_3) = \theta^{\beta-v} \end{array} \right.$

   (b) $\gamma'_1 \cdot Translate_n(\gamma_2, -u * prog(\sigma)) \cdot \gamma'_3$ *avec* $\left\{ \begin{array}{l} prog(\gamma'_1) = \sigma^{\alpha-u} \\ proj(\gamma'_3) = \theta^{\beta+v} \end{array} \right.$

2. *Si $prog(\sigma)$ et $prog(\theta)$ sont de signes différents :*

   (a) $\gamma'_1 \cdot Translate_n(\gamma_2, u * prog(\sigma)) \cdot \gamma'_3$ *avec* $\left\{ \begin{array}{l} prog(\gamma'_1) = \sigma^{\alpha+u} \\ proj(\gamma'_3) = \theta^{\beta+v} \end{array} \right.$

   (b) $\gamma'_1 \cdot Translate_n(\gamma_2, -u * prog(\sigma)) \cdot \gamma'_3$ *avec* $\left\{ \begin{array}{l} prog(\gamma'_1) = \sigma^{\alpha-u} \\ proj(\gamma'_3) = \theta^{\beta-v} \end{array} \right.$

Cette propriété de régularisation va permettre de caractériser un ensemble de représentants des chemins de g^n. Pour le définir nous allons utiliser les ensembles :

$$Chemin\_basic_g(v, v') = \{ \text{ chemins sans cycle de } g \text{ entre } v \text{ et } v' \}$$

$$Chemin\_basic_g(W, W') = \bigcup_{(v,v') \in W \times W'} Chemin\_basic_g(v, v')$$

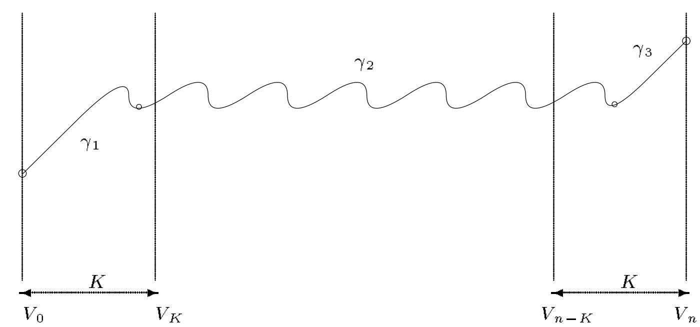


FIG. 4.4: *Un élément de* $Chemin\_transverse_{g,K}(n)$

Pour $K \in \mathbb{N}$, on pose :

$$Chemin\_gauche\_droite_{g,K}(n) \stackrel{\text{def}}{=} \{ \text{ chemin } \gamma_1 \cdot \gamma_2 \cdot \gamma_3 \text{ de } g^n \} \text{ tels que :}$$

$$\begin{cases} \gamma_1 \in Chemin\_basic_{g^K}(V_0, \bigcup_{i=0..K} V_i) \\ \exists \sigma \in Cycles_g^+, \exists \alpha \in \mathbb{N}, proj(\gamma_2) = \sigma^\alpha \\ \gamma_3 \in \bigcup_{i=0..K} Translate_n(Chemin\_basic_{g^K}(V_i, V_K), n - K + i) \end{cases}$$

Et :

$$Chemin\_droite\_gauche_{g,K}(n) \stackrel{\text{def}}{=} \{ \text{ chemin } \gamma_1 \cdot \gamma_2 \cdot \gamma_3 \text{ de } g^n \} \text{ tels que :}$$

$$\begin{cases} \gamma_1 \in Translate_n(Chemin\_basic_{g^K}(V_K, \bigcup_{i=0..K} V_i), n - K) \\ \exists \sigma \in Cycles_g^-, \exists \alpha \in \mathbb{N}, proj(\gamma_2) = \sigma^\alpha \\ \gamma_3 \in Chemin\_basic_{g^K}(\bigcup_{i=0..K} V_i, V_0) \end{cases}$$

On regroupe ces deux notions en :

$$Chemin\_transverse_{g,K}(n) \stackrel{\text{def}}{=} Chemin\_gauche\_droite_{g,K}(n) \bigcup Chemin\_droite\_gauche_{g,K}(n)$$

Un exemple de chemin transverse est donné par la figure 4.4.

Définition 31 *Soit* $\alpha = max\{k(\sigma, \theta), \sigma, \theta \in Cycles_g\}$, $b = \alpha * max\{|\sigma|, \sigma \in Cycles_g\}$ *et* $K = |V| * \alpha * ||Cycles(g)||$. *On définit l'ensemble* $Regulier_g(n)$ *des chemins* μ *de* g^n *d'extrémités dans* $V_0 \cup V_n$ *qui se décomposent en* $\mu = \Pi \mu_i$ *et où chaque* μ_i *vérifie l'une des conditions suivantes :*

$$\begin{array}{l} \mu_i \in [V_0, V_b] \vee \mu_i \in [V_{n-b}, V_n] \\ Translate_n(\mu_i, -b) \in Chemin\_transverse_{g,K}(n - 2b) \\ Translate_n(\mu_i, -b) \in chemin\_basic_{g,K}(V_0, V_0) \\ Translate_n(\mu_i, K + b - n) \in chemin\_basic_{g,K}(V_K, V_K) \end{array}$$

L'aspect des chemins de $Regulier_g(n)$ est donné par la figure 4.5. Le lemme 7 établit que les chemins réguliers de g^n forment un ensemble de représentants conforme au lemme 4.

Lemme 7 *Pour tout chemin* γ *de* g^n *d'extrémités dans* $V_0 \cup V_n$, *il existe un chemin* μ *dans* $Regulier_g(n)$ *qui vérifie* $\mu \trianglelefteq \gamma$

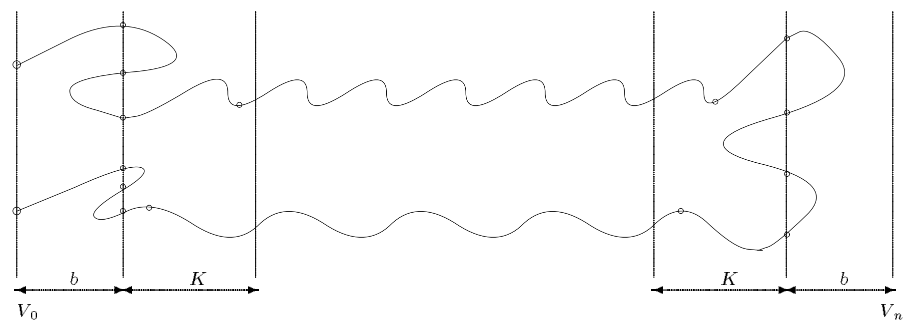


FIG. 4.5: *Un chemin régulier de g^n*

Preuve : Dans ce chapitre on a supposé que pour tout n on a $\neg Borne_g(n)$. Cela implique que les cycles de g^n sont de poids positifs. Ainsi un chemin μ obtenu à partir de γ en lui retirant ces cycles vérifie $\mu \trianglelefteq \gamma$. Les minimums pour $\trianglelefteq$ sont donc bien définis puisqu'ils sont dans l'ensemble des chemins sans cycle.

Soit $\mu = \Pi_{i \in I} \mu_i \trianglelefteq \gamma$ un minimum pour $\trianglelefteq$, avec $\forall i, \mu_i \in [V_0, V_b] \cup [V_b, V_{n-b}] \cup [V_{n-b}, V_n]$.

Nous allons démontrer le lemme en effectuant une récurrence double sur μ.

La première mesure de la récurrence est $\Sigma_{I_0} |\mu_i|$, où I_0 est l'ensemble des indices i pour lesquels μ_i ne vérifie pas les conditions de régularité. Si cette somme est nulle, la propriété est clairement vérifiée.

Nous allons par la suite effectuer des regroupements de cycles comme nous l'avons vu dans les propriétés 20 et 21 pour des chemins dans $[V_b, V_{n-b}]$. Si un chemin $\mu'_{i_0} \sim \mu_{i_0}$ obtenu ne vérifie pas $\mu'_{i_0} \in [V_b, V_{n-b}]$, alors le chemin $\Pi_{i<i_0} \mu_i \cdot \mu'_{i_0} \cdot \Pi_{i>i_0} \mu_i$ sera plus petit pour la première mesure de la récurrence. On suppose donc par la suite que chaque application des propriétés 20 et 21 donne un chemin dans $[V_b, V_{n-b}]$.

Dans ce cadre on montre par récurrence sur la taille des chemins que l'on peut régulariser les chemins. Pour cela on considère la propriété (P) :

$$(P) : \left\{ \begin{array}{l} \text{si } \gamma \text{ est un chemin dont la projection d'un préfixe} \\ \text{est une puissance } \beta \geq \alpha \text{ d'un } \sigma \in Cycle_g\text{, } alors \\ \exists \gamma' \trianglelefteq \gamma \text{ avec soit } \gamma' \lhd \gamma \\ \text{soit } \gamma' = \nu'_1 \cdot \nu'_2 \text{ avec } |\nu'_2| < K \text{ et } \exists \delta, proj(\nu'_1) = \sigma^\delta \\ \text{et } \gamma' \text{ est de même poids et opérateur que } \gamma \end{array} \right.$$

En effet, si $\beta > \alpha$, on peut écrire $\nu_1 = \sigma_1 \cdot \nu'_1$ avec $proj(\sigma_1) = \sigma$ et appliquer l'hypothèse de récurrence de (P) sur $\nu'_1 \cdot \nu_2$.

Si $\beta = \alpha$ et $|\nu_2| > K$, alors par les propriétés 18, 20 il existe $\theta \in Cycles(g)$ et $\gamma_1, \gamma_2, \gamma_3$ tels que le chemin $\gamma' = \nu_1 \cdot \gamma_1 \cdot \gamma_2 \cdot \gamma_3$ vérifie $\gamma' \sim \gamma$, et $proj(\gamma_2) = \theta^\alpha$.

On pose $c_\sigma = poid(\sigma)/|prog(\sigma)|$ et $c_\theta = poid(\theta)/|prog(\theta)|$. Si $c_\sigma \neq c_\theta$, alors l'un des chemins γ'', transformés de γ' par la propriété 21, vérifie $\gamma'' \lhd \gamma'$.

Si $c_\sigma = c_\theta$ alors selon le signe relatif de $prog(\sigma)$ et $prog(\theta)$, on peut encore appliquer la propriété21. Et selon les signes relatifs de $prog(\sigma)$ et $prog(\theta)$, ou bien le chemin donné par 1)a) permet de se ramener à $\beta > \alpha$, ou bien le chemin donné par 2)b) est strictement plus court, ce qui prouve la propriété (P).

On peut alors conclure le lemme car chaque $\mu_i, i \in I_0$ est de longueur supérieure à K, on peut donc lui appliquer les propriétés 18 puis 20. On obtient un chemin μ'_i qui s'il n'est pas dans $[V_b, V_{n-b}]$ fait décroître le premier champs de la récurrence, et sinon est régularisable d'après (P).

□

4.4 Expression de $\Theta(g^n)$ dans $\mathcal{T}h$

Les chemins qui sont dans $Regulier_g(n)$ se caractérisent par des sous chemins bornés et des chemins transverses dépendants de n. Nous montrons tout d'abord que l'on peut caractériser les chemins transverses dans $\mathcal{T}h$. Ce sont des composés de chemins $\gamma_1 \cdot \gamma_2 \cdot \gamma_3$ ou γ_1 et γ_3 sont dans des ensembles finis, et les γ_2 sont des expansions de chemins de $K(g)$ de la forme σ^α.

Soit K fixé, on pose

$$\begin{array}{l}
E_1(v,w) = \{\vec{M}(\gamma), \gamma \in \bigcup_{i=0..K} Chemin_basic(v^0, w^i)\} \\
E_2(v,w) = \{\vec{M}(\gamma), \gamma \in \bigcup_{i=0..K} Chemin_basic(v^i, w^K)\} \\
E_3(v,w) = \{\vec{M}(\gamma), \gamma \in \bigcup_{i=0..K} Chemin_basic(v^K, w^i)\} \\
E_4(v,w) = \{\vec{M}(\gamma), \gamma \in \bigcup_{i=0..K} Chemin_basic(v^i, w^0)\}
\end{array}$$

Ces ensembles sont finis, et permettent de définir les formules $Chemin\_gauche\_droite_{g,K}$ et $Chemin\_droite\_gauche_{g;K}$ sur les multi-ensembles d'arêtes de $K(g)$:

$$Chemin\_gauche\_droite_{g,K,v,v'}(\vec{M}, n) \stackrel{\text{def}}{=} \begin{cases} \bigvee_{v'' \in V} \bigvee_{\substack{\vec{M}_1 \in E_1(v,v'') \\ \vec{M}_3 \in E_2(v'',v')}}, \exists \vec{M}_2, \exists \alpha, \\ \bigvee_{\vec{M}' \in Cycles^+_g(v'')} \vec{M}_2 = \vec{M}' * \alpha \\ n = prog(\vec{M}_1) + prog(\vec{M}_2) + prog(\vec{M}_3) \\ \vec{M} = \vec{M}_1 + \vec{M}_2 + \vec{M}_3 \end{cases}$$

$$chemin\_droite\_gauche_{g,K,v,v'}(\vec{M}, n) \stackrel{\text{def}}{=} \begin{cases} \bigvee_{v'' \in V} \bigvee_{\substack{\vec{M}_1 \in E_3(v,v'') \\ \vec{M}_3 \in E_4(v'',v')}}, \exists \vec{M}_2, \exists \alpha, \\ \bigvee_{\vec{M}' \in Cycles^-_g(v'')} \vec{M}_2 = \vec{M}' * \alpha \\ n = prog(\vec{M}_1) + prog(\vec{M}_2) + prog(\vec{M}_3) \\ \vec{M} = \vec{M}_1 + \vec{M}_2 + \vec{M}_3 \end{cases}$$

On a clairement :

$$\left[\begin{array}{l} \gamma \in Chemin\_gauche\_droite_{g,K}(n) \\ proj(source(\gamma)) = v, proj(dest(\gamma)) = v' \end{array} \right. \Leftrightarrow \vec{M}(\gamma) \models Chemin\_gauche\_droite_{g,K,v,v'}(\vec{M})$$

$$\left[\begin{array}{l} \gamma \in Chemin\_droite\_gauche_{g,K}(n) \\ proj(source(\gamma)) = v, proj(dest(\gamma)) = v' \end{array} \right. \Leftrightarrow \vec{M}(\gamma) \models Chemin\_droite\_gauche_{g,K,v,v'}(\vec{M})$$

Théorème 1 *La formule $\Theta(g^n)$ de variables libres n, X et X' est expressible dans $\mathcal{T}h$.*

Si n_0 est tel que $Borne_g(n_0)$ on a :

$$\Theta(g^n) \models\!\mid \bigvee_{i=0..n_0-1} (n = i \wedge \Theta(g^i))$$

Si $\forall n, \neg Borne_g(n)$ on a :

$$\Theta(g^n)(n, X, X') \models\!\mid \left\{ \begin{array}{l} \bigvee_{i=0..2b}(n = i \wedge \Theta(g^i)) \\ \\ n \geq 2b \Rightarrow \left\{ \begin{array}{l} \exists X_1, X_2, \Theta(g^b)(X, X_1) \wedge \Theta(g^b)(X_2, X') \\ \forall \vec{M}, \bigvee_{v_i, v_j \in V} \\ Chemin\_gauche\_droite_{g,K,v_i,v_j}(\vec{M}, n) \Rightarrow \\ \quad \left(\begin{array}{l} ope\_strict(\vec{M}) \Rightarrow x_i^1 < x_j^2 + poids(\vec{M}) \\ \neg ope\_strict(\vec{M}) \Rightarrow x_i^1 \leq x_j^2 + poids(\vec{M}) \end{array} \right) \\ \\ Chemin\_droite\_gauche_{g,K,v_i,v_j}(\vec{M}, n) \Rightarrow \\ \quad \left(\begin{array}{l} ope\_strict(\vec{M}) \Rightarrow x_i^2 < x_j^1 + poids(\vec{M}) \\ \neg ope\_strict(\vec{M}) \Rightarrow x_i^2 \leq x_j^1 + poids(\vec{M}) \end{array} \right) \\ \\ \exists X_3, \Theta(g^K)(X_1, X_3) \wedge \exists X_4, \Theta(g^K)(X_4, X_2) \end{array} \right. \end{array} \right.$$

On peut passer à présent d'un résultat sur les graphes d'entrées-sorties à un résultat sur les automates plats. En effet le langage des mots d'un automate plat est de la forme $\Pi_{i \in I} w_i \cdot (u_i)^*$. L'ensemble de ces mots est de la forme $\Pi_{i \in I} w_i \cdot (u_i)^{n_i}$. On peut donc exprimer la relation binaire d'accessibilité comme composition des relations $\exists n_i, \Phi(u_i^{n_i})$ et $\Phi(w_i)$. Par la propriété 9, il existe un graphe d'entrées-sorties g_i de longueur 1 tel que $\Phi(u_i) \models\!\mid \Theta(g_i)$, et puisque $\exists n_i, \theta(g_i^{n_i})$ est justement exprimable dans $\mathcal{T}h$ (Théorème 1), on a le résultat :

Théorème 2 *La relation binaire d'accessibilité pour les automates à compteurs plats est expressible dans $\mathcal{T}h$.*

On peut utiliser plus particulièrement le théorème 1 pour décider des propriétés sur une séquence fixée de transitions d'un automate à compteurs. On peut s'interroger sur son caractère itérable ou non. Prenons le cas de la transformations $f : x \geq 0 \wedge x' = x - 1$. Si l'on applique n fois successivement cette transformation, la relation de variable libre n, x, x' correspondante est $\psi = x \geq 0 \wedge x' = x - n \wedge x \geq n - 1$. Ainsi il existe toujours une valeur initiale de la variable x qui permette d'atteindre un nombre donné d'itérations. Par contre lorsque la variable x est fixée, le nombre d'itération est borné.

Corollaire 22 *Soit w une séquence de transitions d'un automate à compteurs, le problème de savoir si l'exécution infinie w^ω est possible, est décidable.*

Le problème de savoir si w est itérable un nombre arbitraire de fois est décidable.

Preuve :

En effet, on considère g le graphe d'entrées-sorties tel que $\Theta(g) \models\!\mid \Phi(w)$. Les réponses aux problèmes sont données respectivement par les modèles des formules :

$$\exists X, \forall n, \exists X', \Theta(g^n)(n, X, X')$$

$$\forall n, \exists X, \exists X', \Theta(g^n)(n, X, X')$$

qui sont décidables puisqu'ils sont exprimés dans $\mathcal{T}_h$ qui est décidable. $\square$

Chapitre 5

Condition Suffisante pour un Aplatissement

Dans le chapitre concernant les automates à compteurs, nous avons justifié les raisons de l'étude du fragment $\mathcal{F}_c$: à la fois pour son expressivité, et pour sa nouveauté par rapport à d'autres fragments déjà étudiés. Toutefois il est difficile d'appréhender son intérêt pratique, qui peut être ressenti comme réduit à l'étude d'un automate à un seul état et à une seule transition.

Même si les propriétés de compositions des relations permettent d'utiliser ce résultat comme pré-processing dans des outils plus complets, et d'imaginer un peu mieux son intérêt, il est plus démonstratif d'isoler une classe significative d'automates étendus dont l'expressivité se réduit à celle d'automates plats.

Nous présentons ici un résultat de langage formel qui est un moyen d'effectuer ce genre de réduction. Il s'inspire de l'intuition suivante : si une exécution contient deux suites consécutives d'actions qui individuellement ont le même effet de transformation sur les variables, alors l'ordre de ces exécutions n'importe pas.

Il n'est donc plus nécessaire d'étudier toutes les exécutions, il est suffisant de s'intéresser à un sous ensemble représentatif. Pour nous, deux exécutions sont équivalentes si elles induisent la même contrainte de transformation sur les variables. Nous donnons ici une condition pour qu'un ensemble d'exécutions représentatives soit donné par un automate plat. Si les transformations de cet automate sont des formules dans $\mathcal{F}_c$, on peut alors utiliser le résultat du chapitre précédent, et exprimer la relation binaire d'accessibilité. Mais ce procédé n'est pas propre aux automates à compteurs, et on peut imaginer qu'il s'appliquera avec succès à d'autres modèles.

Pour montrer cette réduction finale aux automates plats, on utilise une relation sur les séquences de transitions qui permet de commuter deux boucles en relation. Ainsi, si l'on peut permuter les transitions a et b en préservant la représentativité des exécutions sur $(a|b)^*$, alors on aura des représentants de chaque exécution dans a^*b^* qui est un langage d'automate plat.

Nous exprimons ce résultat sous forme d'un théorème de langage formel.

Théorème 3 *Soit Σ un alphabet fini, et $\sim, \approx$ deux relations sur Σ^*. Si $\sim$ est une congruence droite d'indice fini, $\approx$ est une congruence, telles que lorsque $u \sim v \sim w$ on a $u \cdot v \cdot w \approx v \cdot u \cdot w$, alors il existe un automate plat qui reconnaît un ensemble de représentants pour $\approx$.*

On peut s'interroger sur la règle de commutation choisie dans ce théorème, et se demander si le résultat tient encore pour $u \cdot v \approx v \cdot u$. C'est le cas puisqu'alors $\forall w$ (et en particulier $w \sim u \sim v$) on a $u \cdot v \cdot w \approx v \cdot u \cdot w$

Corollaire 23 *Soit Σ un alphabet fini, et $\sim, \approx$ deux relations sur Σ^*. Si $\sim$ est une congruence droite d'indice fini, $\approx$ est une congruence, telles que lorsque $u \sim v$ on a $u \cdot v \approx v \cdot u$, alors il existe un automate plat qui reconnaît un ensemble de représentants pour $\approx$.*

La preuve du théorème se décompose en plusieurs étapes. La première utilise le caractère fini de $\sim$ pour exhiber une séquence candidate pour une commutation dans un mot de Σ^* suffisamment long. La seconde consiste en une réécriture de ce mot, par application de la règle de commutation, pour obtenir un mot équivalent pour $\approx$ mais plus petit pour l'ordre lexicographique. Cette réécriture termine et le langage obtenu est régulier. Un examen de l'expression rationnelle de ce langage, et l'utilisation de critère combinatoire sur les boucles de cette expression, permet de la transformer en une expression rationnelle caractéristique d'un automate plat.

Définition 32 *Soit $w \in \Sigma^*$. Pour tout $i \leq |w|$ on note $w(i)$ l'élément de Σ de position i dans w.*

Pour $1 \leq i < j \leq |w|$, on note $w[i,j]$ le facteur de w qui part de la position i de w jusqu'à la position j de w, et qui inclut les extrémités.

Propriété 24 *Soit k l'indice de $\sim$. Il existe une constante $n(k)$ telle que pour tout $w \in \Sigma^*$, pour toute suite strictement croissante $p_1 \ldots p_{n(k)}$ de positions dans w, il existe une décomposition $w = w_0 \cdot w_1 \cdot w_2 \cdot w_3 \cdot w_4$ avec $w_1 \sim w_2 \sim w_3$ et des entiers $i < j < k < l$ tels que $w_1 = w[p_i, p_j - 1], w_2 = [p_j, p_k - 1], w_3 = [p_k, p_l - 1]$.*

Preuve :

A chaque position p_i on associe l'ensemble $S(i)$ formé par les classes des facteurs $w[p_i, p_j - 1]$ pour$p_j > p_i$, et l'ensemble $P(i)$ des classes des facteurs $w[p_j, p_i - 1]$ pour $p_j < p_i$:

$$S(i) = \{c \in \Sigma^*_{/\sim}, \exists j > i, w[p_i, p_j - 1] \in c\}$$

$$P(i) = \{c \in \Sigma^*_{/\sim}, \exists j < i, w[p_j, p_i - 1] \in c\}$$

Les paires $S(i), P(i)$ prennent au plus 2^{2k} valeurs distinctes. Ainsi si $n(k) = 2^{2k} + 1$, il existe deux positions $i < j$ telles que $(S(i), P(i)) = (S(j), P(j))$. Soit $w_2 = w[p_i, p_j - 1]$, et c sa classe pour $\sim$. On a $c \in S(i) = S(j)$. Il existe donc

$k > j$ tel que $w_3 = w[p_j, p_k - 1] \sim w_2$. Mais on a également $c \in P(j) = P(i)$. Il existe donc $l < i$ tel que $w[p_l, p_i - 1] = w_1 \sim w_2 \sim w_3$. $\square$

Nous sommes donc capable d'exhiber des candidats à la commutation lorsque les mots sont suffisamment longs. Il reste à présenter une stratégie ordonnée d'application de ces règles pour obtenir un langage plat, ou dans un premier temps un langage pour lequel l'application d'une règle laisse le mot invariant. Pour cela on considère un système de réécriture.

Définition 33 *Le système de réécriture $R(p)$ est défini par l'ensemble infini de règles :*

$$R(p) \stackrel{def}{=} \{(u{\cdot}x){\cdot}(v{\cdot}y){\cdot}w \rightarrow (v{\cdot}y){\cdot}(u{\cdot}x){\cdot}w,\ si\ u{\cdot}x \sim v{\cdot}y \sim w; u, v \in \Sigma^p\ \ et\ u >_{lex} v\}$$

où $>_{lex}$ est l'ordre lexicographique.

Pour chaque mot w, la réécriture de w par des règles de $R(p)$ termine : en effet, chaque réécriture fait décroître strictement l'ordre lexicographique, qui est bien fondé. Même si le système $R(p)$ a un nombre infini de règles, nous pouvons caractériser l'ensemble des mots irréductibles pour $R(p)$. Plus précisément, le fait que $\sim$ soit une congruence droite d'indice fini assure que cet ensemble est régulier.

Propriété 25 *Pour tout p, le système de réécriture $R(p)$ est convergent sur Σ^*. L'ensemble de ses formes normales est régulier, et lorsqu'un mot u se réécrit en un mot v par $R(p)$, alors $u \approx v$.*

Preuve : Soit $\mathcal{A} = (\Sigma^*_{/\sim}, q_0, \Sigma, \Delta)$ l'automate d'alphabet Σ dont les états sont les classes de $\sim$ (qui est d'indice fini), défini par $\forall x \in \Sigma, q_0 \xrightarrow[\mathcal{A}]{x} c \in \Delta$ lorsque c est la classe de x, et $c_1 \xrightarrow[\mathcal{A}]{x} c_2$ lorsque c_2 est la classe de $c_1 \cdot x$. Par construction cet automate vérifie $\forall w \in \Sigma^*, q_0 \xrightarrow[\mathcal{A}]{w} c$ ssi c est la classe de w.

On peut aussi définir un automate $\mathcal{A}_{u,c}$ pour chaque $u \in \Sigma^p$ qui reconnaît le langage $L(u, c)$ des mots de classe c qui s'écrivent $u.x$. On pose $\mathcal{A}_{u,c} = (\Sigma^*_{/\sim} \times \{0..|u|\}, (q_0, 0), \Sigma, \Delta_u, (c, |u|))$ où l'état final est $(c, |u|)$. L'ensemble de ses transitions Δ_u est composé des transitions $\forall i \leq |u|, \forall a \in \Sigma_{/\sim}, (a, i-1) \xrightarrow[\mathcal{A}_{u,c}]{u(i)} (b, i)$ si $a \xrightarrow[\mathcal{A}]{u(i)} b$, et $\forall x \in \Sigma, (a, |u|) \xrightarrow[\mathcal{A}_{u,c}]{x} (b, |u|)$ ssi $a \xrightarrow[\mathcal{A}]{x} b$.

Ces ensembles permettent de définir par des opérations booléennes l'ensemble des mots pour lesquels une règle de $R(p)$ s'applique, cet ensemble est donc régulier, il s'agit de :

$$\Sigma^* \cdot [\bigcup_{u,v \in \Sigma^p, u >_{lex} v, c \in (\Sigma/\sim)} Lang(A_{u,c}) \cdot Lang(A_{v,c}) \cdot Lang(A_{\epsilon,c})] \cdot \Sigma^*$$

L'ensemble des mots irréductibles pour $R(p)$ est le complémentaire de cet ensemble, il est donc régulier et calculable de manière effective.

Pour la seconde partie de la propriété, si $u \xrightarrow[R(p)]{*} v$, et puisque chaque règle utilisée vérifie $(u_1 \cdot x) \cdot (v_1 \cdot y) \cdot w \approx (v_1 \cdot y) \cdot (u_1 \cdot x) \cdot w$, par récurrence immédiate sur la longueur de la réduction on a encore $u \approx v$.

□

Voici à présent une suite de lemmes de combinatoire des mots qui va nous permettre de trouver une puissance commune aux mots qui forment des boucles dans le langage des mots irréductibles pour $R(p)$. Ainsi le langage d'une expression non plate $(u+v)^*$ irréductible sera le même que le langage d'une expression w^*.

Lemme 8 ([],[]) *Si u est un mot infini qui contient au plus n facteurs distincts de longueur n, alors u est ultimement périodique avec une période de longueur inférieure à n.*

Propriété 26 *Si un mot infini m est égal à u^ω et à v^ω, alors il existe un mot w de longueur $pgcd(|u|, |v|)$ tel que u, v sont des puissances de w.*

Preuve : En effet, si $|u|$ et $|v|$ sont des périodes de m on a :

$$\forall i, \left[\begin{array}{l} m(i+|u|) = m(i) \\ m(i+|v|) = m(i) \end{array} \right.$$

Soit p le plus grand diviseur commun de $|u|$ et de $|v|$.

$|u|/p$ et $|v|/p$ sont premiers entre eux. Il existe donc des entiers α, β tels que $\alpha * |u|/p - \beta * |v|/p = 1$. Ainsi :

$$\begin{aligned} \forall i, m(i+p) \quad &= m(i + \alpha * |u| - \beta * |v|) \\ &= m(i + \alpha * |u|) \\ &= m(i) \end{aligned}$$

On pose donc $w = m[1, p]$. Par définition p divise $|u|$ et $|v|$, c'est donc que u et v sont des puissances de w. □

Propriété 27 *Soit $w \in \Sigma^*$ tel que w^ω contiennent au plus n facteurs distincts de longueur n. Alors il existe un mot u de longueur inférieure à n tel que w soit une puissance de u.*

Preuve : Si $|w| < n$, alors w convient. Sinon, d'après le lemme 8, il existe un mot v de longueur inférieure à n tel que $w^\omega = v^\omega$. Par la propriété 26, il existe un mot u de taille $pgcd(|w|, |v|)$ tel que w et v soient des puissances de u. Puisque $pgcd(|w|, |v|) \leq |v|$, on a $|u| \leq n$ □

Propriété 28 *Soit $w \in \Sigma^*$ tel que tout mot de w^* soit irréductible pour $R(n(k))$. Alors il existe un mot u de taille au plus $n(k) - 1$ tel que w soit une puissance de u.*

Preuve : On pose $N = n(k) - 1$. Considérons l'ensemble des facteurs de w^ω de longueur N.

$$Fact(N) = \{f_i, i \in \mathbb{N}\} \text{ où } f_i = w^\omega[i, n + (i-1)]$$

Si $|Fact(N)| \leq N$, alors la propriété 27 s'applique, et la propriété 28 est acquise. Supposons que $|Fact(N)| > N$, alors il existe une suite de $N+1$ facteurs distincts de longueur N, $f_{i_0} \dots f_{i_N}$ décroissante strictement pour l'ordre lexicographique. Nous montrons que cela est contraire à l'hypothèse w^* irréductible pour $R(n(k))$.

En effet, vu la régularité de w^ω, il existe une puissance suffisamment grande α telle que dans w^α les facteurs f_{i_j} se trouvent ordonnés selon : $v_0 \cdot f_{i_0} \cdot v_1 \cdot f_{i_1} \cdot v_2 \cdot f_{i_2} \cdots v_N \cdot f_{i_N} \cdot v_{N+1}$. On isole ainsi des positions croissantes $p_0, \dots, p_N$ qui renvoient respectivement aux facteurs $f_{i_0}, \dots, f_{i_N}$. Par le lemme 24, w^α contient un facteur $w_1 \cdot w_2 \cdot w_3$ avec $w_1 \sim w_2 \sim w_3$ et $w_1 >_{lex} w_2 >_{lex} w_3$ puisque leurs préfixes de longueur N sont déjà dans cet ordre. Cela signifie que w^α n'est pas irréductible pour $R(n(k))$, ce qui est contraire à l'hypothèse. C'est donc que $|Fact(N)| \leq N$. La propriété 28 est donc vérifiée.

□

Propriété 29 *Soit $w_1 \dots w_n \in \Sigma^*$ tel que tout mot de $(w_1 + \dots + w_n)^*$ est irréductible pour $R(n(k))$. Alors il existe un mot u de taille au plus $n(k) - 1$ tel que chaque w_i soit une puissance de u.*

Preuve : En particulier, chaque mot de w_i^* est irréductible pour $R(n(k))$. Par la propriété 28 il existe donc des u_i de longueur au plus $n(k) - 1$, que l'on peut supposer minimale, et des entiers α_i tels que $w_i = (u_i)^{\alpha_i}$.

Nous allons montrer que $\forall i, j, u_i = u_j$. Pour cela nous allons montrer qu'il existe des entiers α, β tels que $u_i^\alpha = u_j^\beta$. C'est-à-dire que u_i et u_j sont des périodes de w_i^ω. Alors la propriété 26 et le fait que u_i est de longueur minimale imposera que u_j est une puissance de u_i, et symétriquement que $u_i = u_j$.

Pour tout entier q, les mots de $(w_i \cdot w_j^q)^*$ sont irréductibles pour $R(n(k))$. Il existe donc (propriété 28) un mot v_q, que l'on suppose de longueur minimale, tel que $w_i \cdot w_j^q$ soit une puissance de v_q, et $|v_q| \leq n(k) - 1$. Cette condition sur la longueur assure que les v_q prennent leurs valeurs dans un domaine fini, et donc qu'il existe des entiers $q_1 > q_0$ pour lesquels on peut prendre $v_{q_1} = v_{q_0} = v$. Il existe donc des puissances $\beta_1 > \beta_0$ pour lesquelles $w_i \cdot w_j^{q_1} = v^{\beta_1}$ et $w_i \cdot w_j^{q_0} = v^{\beta_0}$. Alors :

$$v^{\beta_1} = w_i \cdot w_j^{q_1} = (w_i \cdot w_j^{q_0}) \cdot w_j^{q_1 - q_0} = v^{\beta_0} \cdot u_j^{\alpha_j (q_1 - q_0)}$$

Et donc $v^{\beta_1 - \beta_0} = u_j^{\alpha_j (q_1 - q_0)}$. C'est que $v = u_j$, mais aussi :

$$w_i \cdot w_j^{q_1} = u_j^{\beta_1} = (u_i^{\alpha_i}) \cdot (u_j^{q_1 * \alpha_j})$$

Et à nouveau $u_j^{\beta_1 - q_1 * \alpha_j} = u_i^{\alpha_i}$ permet d'affirmer que $u_i = u_j$.

□

Nous avons examiné avec les propriétés 28 et29 le cas des boucles simples, et des composées de boucles simples, il reste le cas où ces boucles sont imbriquées, c'est le cas général :

Propriété 30 *Si L est un langage régulier tel que L^* est irréductible pour $R(n(k))$, alors il existe un nombre fini de paires de mots (u_i, v_i) telles que $L^* = \Sigma_{i=1}^n u_i \cdot v_i^*$.*

Preuve : En effet, la propriété 29 montre qu'il existe un mot u tel que tout mot de L^* soit une puissance de u. Si l'on regarde l'ensemble des longueurs des mots de L^*, on a un sous ensemble reconnaissable de $\mathbb{N}$, qui est donc une union finie de progression arithmétique $S = \bigcup_{i=1}^n \{a_i + b_i * z, z \in \mathbb{N}\}$ pour $a_i, b_i \in \mathbb{N}$. Chacune de ces longueurs est divisible par $|u|$ et en particulier chaque a_i (si $z = 0$). C'est donc que les b_i le sont aussi. On pose $a'_i = a_i/|u|, b'_i = b_i/|u|$. On a alors :

$$L^* = \{\bigcup_{i=1}^n u^{a'_i + z b'_i}, z \in \mathbb{N}\} = \Sigma_{i=1}^n u^{a'_i} \cdot (u^{b'_i})^*$$

□

On peut alors rassembler ces résultats pour démontrer le théorème 3 :

Preuve : (Théorème 3) D'après la propriété 25, il existe une expression régulière e irréductible pour $R(n(k))$ qui contienne un ensemble de représentants pour $\approx$. En particulier chacune de ces sous expressions est aussi irréductible pour $R(n(k))$. La propriété 30 permet de remplacer les sous expressions s^* de e par des combinaisons linéaires de mots. Il existe donc une expression rationnelle de la forme $\Sigma_{i=1}^n u_{i,1} \cdot v_{i,1}^* \ldots v_{i,n_i}^* \cdot u_{i,n_i+1}$ qui reconnaît le même langage que e. Cette expression est typique des automates plats. □

Théorème 4 *Soit $A = (Q, \Sigma, \Delta)$ un automate d'alphabet Σ et d'ensemble de transitions Δ. Soit $\sim$ une congruence droite d'indice fini sur Σ^*, $\approx$ une congruence telle que si $u \sim v \sim w$ alors $u \cdot v \cdot w \approx v \cdot u \cdot w$. Alors il existe un automate plat $A_p(P, p_0, \Delta^*, \Omega)$ et une fonction f telle que $q \xrightarrow[A]{*} q' \Leftrightarrow \exists p, f(p) = (q, q') \wedge p_0 \xrightarrow[A_p]{*} p$.*

Preuve : Soit $\sim_A$ la relation sur Δ^* telle que $u \sim_A v$ ssi u et v forment des chemins (et pas nécessairement des exécutions) dans l'automate A. L'indice de cette relation est en $O(|Q|^2)$ et il s'agit clairement d'une congruence droite.

On définit alors $\sim'$ sur Δ^* par $u \sim' v$ ssi $u \sim_A v$ et $trace(u) \sim trace(v)$, où la trace d'une suite de transitions de A est donnée par la succession de ses étiquettes. On définit également $\approx'$ sur Δ^* par $u \approx v$ ssi $u \sim_A v$ et $trace(u) \approx trace(v)$.

On a encore $u \sim' v \sim' w \Rightarrow u \cdot v \cdot w \approx' v \cdot u \cdot w$, car $u \sim_A v \sim_A w$ impose que les états sources et destinations de ces exécutions sont identiques pour u, v et w. Ainsi si $u \cdot v \cdot w$ n'est pas un chemin possible dans l'automate, ce sera encore le cas pour $v \cdot u \cdot w$. Dans le cas contraire, on aura de plus que la source et la destination de u sont les mêmes, et sont égales à celles de v, w, la commutation préserve donc $\sim_A$, et par hypothèse elle préserve aussi $\approx$.

On peut donc appliquer le théorème 3 sur Δ^*, $\sim'$ et $\approx'$, qui nous donne un langage plat de représentants pour $\approx'$ reconnu par un automate $A_{p_1} = (P_1, p_0, \Delta^*, \Omega_1)$.

On transforme alors A_{p_1} en supprimant les transitions incompatibles pour $\sim_A$. Puisque A_{p_1} est plat, chaque transition est impliquée dans au plus une boucle. On itère le processus suivant jusqu'à ce que le nombre de transitions se stabilise :

- s'il existe deux transitions (p_1, δ_1, p_2) et (p_2, δ_2, p_3) de Ω_1, dont la première n'est pas impliquée dans une boucle, telle que $\delta_1 = (q_1, a, q_2)$ et $\delta_2 = (q_3, b, q_4)$ avec $q_2 \neq q_3$, alors on supprime la transition (p_2, δ_2, p_3).

- Lorsque l'étape précédente n'est plus applicable, s'il existe deux transitions (p_1, δ_1, p_2) et (p_2, δ_2, p_3), dont la première est impliquée dans une boucle, telle que $\delta_1 = (q_1, a, q_2)$ et $\delta_2 = (q_3, b, q_4)$ avec $q_2 \neq q_3$, alors on supprime celles des transitions qui sont dans une boucle.

On obtient alors un automate $A_p = (P, p_0, \Delta^*, \Omega)$, nécessairement plat car c'est un sous-automate de A_{p_1}, sur lequel on définit la fonction $f : P \setminus \{p_0\} \to Q \times Q$ telle que $f(p) = (q, q')$ ssi $\forall w, p_0 \xrightarrow[A_p]{w} p \Leftrightarrow trace(w)$ est un chemin de q à q' dans A. On a alors montré que la fonction f est celle de la propriété :

Soit $q \xrightarrow[A]{w} q'$ une exécution dans A, il existe une suite de transitions w_Δ de trace w qui correspond à l'une des exécutions. Par construction de A_{p_1}, il existe w' reconnu par A_{p_1} avec $w' \approx' w_\Delta$. En particulier $w' \sim_A w_\Delta$ assure que w' est aussi reconnu par A_p donc $p_0 \xrightarrow[A_p]{w'} p$ avec $f(p) = (q, q')$

Réciproquement si $p_0 \xrightarrow[A_p]{w} p$ alors $q \xrightarrow[A]{trace(w)} q'$ si $f(p) = (q, q')$ par définition de f. $\square$

Corollaire 31 *Si A est un automate étendu, dont la fonction de sémantique est Φ, et que $\approx$ est défini par $u \approx v$ ssi $\Phi(u) \models\!\mid \Phi(v)$, alors A_p muni de la sémantique $\Phi_p((q, a, q')) = \Phi(a)$ émule A.*

Preuve : L'application d'émulation est $\Theta(p, p') = f(p')$ si $p = p_0$, et elle n'est pas définie pour $p \neq p_0$. La preuve consiste au diagramme suivant, sur lequel on passe de la sémantique de mot à une sémantique utilisant Φ ou Φ_p en utilisant la remarque 2.

$$
\begin{array}{ll|ll}
(q,V) & \xrightarrow[A]{w} (q',V') & (q,V) & \xrightarrow[A]{trace(w)} (q',V') \\
 & \Downarrow \quad (V,V') \models \Phi(w) & & \Uparrow \quad (V,V') \models \Phi(trace(w)) \\
q & \xrightarrow[A]{w_\Delta} q' & q & \xrightarrow[A]{trace(w)} q' \\
 & \Downarrow \quad \exists w' \approx' w_\Delta & & \Uparrow \quad f(p) = (q,q') \\
p_0 & \xrightarrow[A_p]{w'} p & p_0 & \xrightarrow[A_p]{w} p \\
 & \Downarrow \quad \Phi(trace(w')) \models\!\!\!\dashv \Phi(w) & & \Uparrow \quad (V,V') \models \Phi(trace(w)) \\
(p_0,V) & \xrightarrow[A_p]{w'} (p,V') & (p_0,V) & \xrightarrow[A_p]{w} (p,V')
\end{array}
$$

□

Chapitre 6

Relation Binaire d'Accessibilité pour les Automates Temporisés

Dans les chapitres précédents, nous avons étudié les automates à compteurs plats, puis nous avons donné un moyen général d'aplatissement d'automates à variables. Il reste à présenter une classe satisfaisante pour laquelle ce résultat s'applique. Nous allons le faire avec les automates temporisés. Ces automates ont été abondamment étudiés et utilisés depuis leur introduction, à la fois parce que l'on sait leur appliquer de nombreux processus de décision, et parce qu'ils se sont révélés suffisamment expressifs pour modéliser nombre de problèmes réels. Outre le fait qu'exprimer la relation binaire d'accessibilité pour les automates temporisés n'est pas un processus de décision usuel, cette approche permet de décider de nouvelles classes de problèmes.

6.1 Introduction

La preuve de ce résultat se décompose en plusieurs étapes. La première est la réduction des automates temporisés aux automates à compteurs, qui a été présentée dans la section 2.3.2. La seconde consiste à utiliser le lien entre les exécutions et les graphes de contraintes, plus particulièrement les Tgraphes conformément au chapitre 3. Alors nous pouvons définir, et c'est l'objet de ce chapitre, une congruence droite d'indice fini sur les Tgraphes, qui par homomorphisme définit une congruence droite sur les exécutions, qui commute conformément au théorème 3. Puisqu'elle préserve la relation binaire d'accessibilité par aplatissement, nous pouvons conclure en utilisant le théorème 4 qui permet déxprimer la relation binaire d'accessibilité pour les automates plats.

D'ordinaire l'étude des automates temporisés se fait également à l'aide d'une congruence droite d'indice fini, appelée "régions". Elle se définit sur les configurations, c'est-à-dire la paire (état,valeur des horloges) et non sur les sequences de transformations. Il s'agit d'une relation R telle que $\forall w, \forall cRc', \forall c_1, (c, c_1) \models \Phi(w) \Rightarrow \exists c'_1 Rc_1, (c'c'_1) \models \Phi'(w)$. La relation $\sim$ que nous définissons sur les sequences de transformations n'a pas de correspondance claire avec les régions. Il existe cependant des intéractions : par exemple si $(c, c') \models \Phi(w)$ et $(c_1, c'_1) \models \Phi(w')$ alors cRc_1 et $c'Rc'_1$. Mais à contrario si $(c, c') \models \Phi(w)$, on n' a pas nécessairement $(c, c') \models \Phi(w')$. Ces deux approches sont donc différentes.

Il y a une autre différence majeure : l'utilisation des régions dans des processus de décision repose sur le fait que les valeurs initiales des horloges sont connues à priori. Il n'est donc nécessaire que de conserver le résultat d'une exécution. Celui-ci est de plus approximé par la région sur laquelle l'exécution aboutit, cette approximation préserve les relations entre horloges finales lorsque celles-ci ont des valeurs suffisamment petites, et tronque les autres sans récupération possible.

Lorsque l'on s'intéresse à la relation induite par une exécution avec des variables initiales libres, d'une part il reste à exprimer les relations entre toutes les variables, à la fois initiales et finales, et d'autre part il ne faut plus que les approximations masquent des relations entre horloges.

Nous allons montrer (propriété 38) qu'étant donné un automate temporisé, les relations possibles entre variables initiales sont en nombre fini. Mais ce n'est plus le cas pour les relations faisant intervenir des variables finales. Nous allons donc également utiliser des approximations dans la définition de $\sim$.

Cette congruence sera utilisée pour effectuer un aplatissement par échange de boucles en relations pour $\sim$. Pour réparer l'effet masquant des approximations, nous conditionnons cet échange par l'utilisation d'une règle plus fine de commutation, la règle $u \sim v \sim w : u \cdot v \cdot w \rightarrow v \cdot u \cdot w$. Celle-ci permet de préserver de manière exacte les relations malgré des approximations faites de manière intermédiaire. Nous rentrons alors dans le cadre du théorème 3.

Notre relation d'équivalence se décompose en relations plus grossières utilisées chacune pour la préservation d'un type de contraintes après commutation, $\sim_1$ et $\sim_2$ préservent les contraintes sur les valeurs initiales et jouent un rôle important pour la préservation des autres relations, $\sim_3$ et $\sim_4$ ont pour objet de préserver les contraintes entre valeurs initiales et finales, et respectivement finales et initiales, tandis que $\sim_5$ préserve les relations entre valeurs finales.

La suite de ce chapitre se décompose ainsi :
Dans un premier temps nous définissons nos notations, puis les deux premières sous relations $\sim_1$ et $\sim_2$ qui déjà forment une première congruence droite et qui vérifient un ensemble de propriétés utiles pour la suite. Alors nous introduisons chacun des raffinements $\sim_i$ en montrant à chaque étape qu'ils forment ensemble une congruence droite. Finalement nous vérifions que pour des graphes équivalents la commutation $u \circ v \circ w \rightarrow v \circ u \circ w$ préserve la relation binaire d'accessibilité, ce qui nous permettra de conclure à l'expressibilité de la relation binaire d'accessibilité.

6.2 Définitions préalables

Dans tout ce chapitre nous considérons toujours le même automate à compteurs $\mathcal{A}(Q, C, \Delta)$, pour lequel on cherche à résoudre le problème de l'expression de la relation binaire d'accessibilité. Nous ne ferons donc pas apparaître les dépendances des définitions avec l'automate. En particulier la constante K, qui représente la plus grande valeur absolue des constantes qui interviennent dans Δ, joue le rôle de valeur maximale pour les informations préservées par les équivalences. Elle apparaît de manière implicite lorsque l'on parle de Tgraphes, puisqu'il s'agit en fait de Tgraphe(K, C).

D'après le théorème 3, lorsqu'un graphe est satisfaisable, seuls les chemins entre les extrémités de ce graphe sont significatifs, c'est pourquoi nous utiliserons les notations :

Définition 34 *Pour tout sommet s_1, s_2 d'un Tgraphe g satisfaisable de longueur n, on note $(s_1 s_2)_g$ l'ensemble des chemins sans cycle de g entre s_1 et s_2 de plus faible poids et d'opérateur le plus strict.*

Pour tous sommets $x, y \in V$, on utilise les abréviations $(xy)_g$, $(x'y')_g$, $(xy)_g$, $(x'y)_g$ pour les ensembles $(x^0 y^0)_g$, $(x^n y^n)_g$, $(x^0 y^n)_g$, $(x^n y^0)_g$, où $x^0, y^0 \in V_0$ et $x^n, y^n \in V_n$ sont les sommets de projection x et y.

Définition 35 *Sur l'ensemble des chemins des Tgraphes, on définit les relations $\lhd, \unlhd, \lhd\!\rhd$ par :*

$$\gamma \lhd \mu \text{ ssi } poids(\gamma) < poids(\mu) \vee \begin{cases} poids(\gamma) = poids(\mu) \\ ope(\gamma) = '<' \neq ope(\mu) \end{cases}$$

$$\gamma \unlhd \mu \text{ ssi } poids(\gamma) < poids(\mu) \vee \begin{cases} poids(\gamma) = poids(\mu) \\ ope(\mu) = '<' \Rightarrow ope(\gamma) = '<' \end{cases}$$

$$\gamma \lhd\!\rhd \mu \text{ ssi } poids(\gamma) = poids(\mu) \wedge ope(\gamma) = ope(\mu)$$

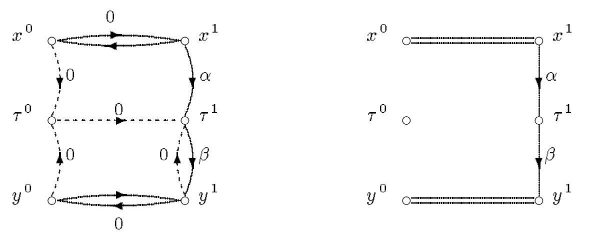
On étend ces relations de façon similaire sur les ensembles de chemins en posant, par exemple dans le cas de $\lhd$:

$$E \lhd F \text{ ssi } \forall(\gamma, \mu) \in E \times F, \gamma \lhd \mu$$

Ainsi elles prennent aussi un sens sur les ensembles vides de chemins.

Ces relations permettent de préciser respectivement qu'un chemin est strictement meilleur, meilleur au sens large ou comparable à un autre chemin, au sens de l'inclusion des ensembles des modèles de leurs contraintes.

Pour définir notre congruence droite, il faut en particulier préserver les contraintes sur les valeurs initiales par composition d'une même exécution. Nous cherchons donc à anticiper ce que peuvent être les conséquences d'exécutions futures sur ces valeurs. On peut montrer que les contraintes possibles sur les valeurs initiales sont en nombre fini, c'est donc que les prédictions possibles, au regard de leurs conséquences sur ces valeurs, sont finitaires.



Représentation complète   Représentation simplifiée

FIG. 6.1: *un élément de $\mathcal{F}(x, y)$*

Plus précisément, on peut caractériser ces anticipations en examinant l'effet de composition par des graphes canoniques. Il s'agit de graphes d'exécutions pour lesquelles aucune horloge n'est remise à zéro, et qui contiennent chacune une contrainte particulière.

Le 1-Tgraphe g_ϵ sert de trame à ces ensembles de graphes canoniques.

Définition 36 *g_ϵ est le 1-Tgraphe associé à la transformation :*

$$(\tau \leq \tau') \bigwedge_{x_k \in C \setminus \{\tau\}} (x'_k = x_k) \wedge (x'_k \leq \tau') \wedge (x_k \leq \tau)$$

Définition 37 *$\forall x', y' \in V_1 \times V_1 \setminus \{\tau^1\}$ de projection x et y, on définit $\mathcal{F}(x, y)$ par la réunion, pour (α, β) entiers dans $[-K, 0] \times [0, K]$, des graphes g_ϵ auxquels on ajoute deux arcs (x', τ^1) de poids α, et (τ^1, y') de poids β, et d'opérateur dans $\{`\leq', <\}$.*

$\forall x \in V_0$, $\mathcal{F}(proj(x), \tau)$ est la réunion, pour α entiers dans $[-K, 0]$, des graphes g_ϵ auxquels on ajoute l'arc (x, τ^0) de poids α, et d'opérateur dans $\{`\leq', <\}$.

L'ensemble de ces prédictions est $\mathcal{F} = g_\epsilon \cup \bigcup_{x,y \in V} \mathcal{F}(x, y)$.

On a $|\mathcal{F}| = O(|C|^2 * K^2)$.

Un exemple d'élément de $\mathcal{F}(x, y)$ est donné par la figure 6.1. Elle donne aussi une représentation simplifiée en ne représentant que les éléments significatifs de ce Tgraphe, les arêtes représentées en pointillés sont considérées comme étant implicites, et on représente par un double trait les contraintes d'égalité de compteurs.

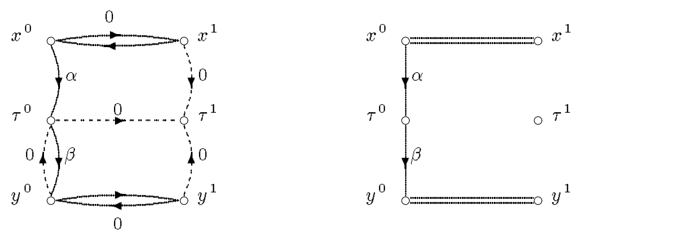
Symétriquement on définit les prédictions sur les passés possibles en utilisant les ensembles $\mathcal{P}(x, y)$ (Une représentation en est donnée par la figure 6.2) :

Définition 38 *$\forall x, y \in V_0 \setminus \{\tau^0\} \times V_0$, $\mathcal{P}(proj(x), proj(y))$ est la réunion, pour (α, β) entiers dans $[-K, 0] \times [0, K]$, des graphes g_ϵ auxquels on ajoute les deux arcs (x, τ^0) de poids α, et (τ^0, y) de poids β, et d'opérateur dans $\{\leq, <\}$.*

$\forall y \in V_1$, $\mathcal{P}(\tau, proj(y))$ est la réunion pour β entier dans $[0, K]$ des graphes g_ϵ auxquels on ajoute l'arc $(\tau^1, proj(y))$ de poids β, et d'opérateur dans $\{\leq, <\}$.

L'ensemble de ces graphes est $\mathcal{P} = \bigcup_{x,y \in V} \mathcal{P}(x, y)$.

On a $|\mathcal{P}| = O(|C|^2 * K^2)$.



Représentation complète Représentation simplifiée

FIG. 6.2: *Un élément de $\mathcal{P}(x, y)$*

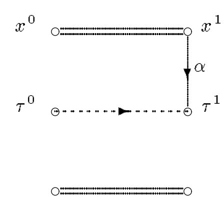


FIG. 6.3: *Un élément de $\mathcal{Q}(x)$*

Pour définir les relations $\sim_i$, nous utiliserons aussi des représentants des ensembles $\mathcal{Q}$ et $\mathcal{R}$ représentés sur les figures 6.3 et 6.4.

Définition 39 *$\forall x' \in V_1 \setminus \{\tau^1\}$ de projection x, on définit $\mathcal{Q}(x)$ par la réunion pour α entier dans $[-K, 0]$ des graphes g_ϵ auxquels on ajoute un arc (x', τ^1) de poids α et d'opérateur dans $\{`\leq', <\}$. Et $\mathcal{Q}(\tau) = g_\epsilon$.*

L'ensemble de ces graphes est $\mathcal{Q} = \bigcup_{x \in V} \mathcal{Q}(x)$. On a $|\mathcal{Q}| = O(|C| * K)$.

Définition 40 *$\forall x' \in V_1 \setminus \{\tau^1\}$ de projection x, on définit $\mathcal{R}(x)$ (voir figure 6.4) par la réunion pour α entier dans $[0, K]$ des graphes g_ϵ auxquels on ajoute un arc (τ^1, x') de poids α et d'opérateur dans $\{`\leq', <\}$. Et $\mathcal{R}(\tau)$ est le graphe vide.*

L'ensemble de ces graphes est $\mathcal{R} = \bigcup_{x \in V} \mathcal{R}(x)$. On a $|\mathcal{R}| = O(|C| * K)$.

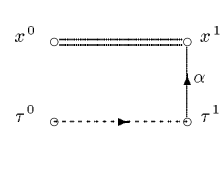


FIG. 6.4: *Un élément de $\mathcal{R}(x)$*

6.3 Une première congruence droite

L'ensemble $\mathcal{F}$ permet de définir les deux premières relations non triviales $\sim_1$ et $\sim_2$ qui ensemble formeront une première congruence droite.

Définition 41 *La relation d'équivalence $\sim_1$ sur les Tgraphes se définit par :*

$$u \sim_1 v \text{ ssi } \forall x, y \in V, \forall g \in \mathcal{F}, \begin{cases} u \circ g \text{ satisfaisable } \Leftrightarrow v \circ g \text{ satisfaisable} \\ \text{et si c'est le cas, } (xy)_{u \circ g} \lhd\!\rhd (xy)_{v \circ g} \end{cases}$$

Définition 42 *La relation d'équivalence $\sim_2$ sur les Tgraphes se définit par $u \sim_2 v$ ssi :*

$$\forall x, y \in V, \forall g \in \mathcal{F}, \begin{cases} u \circ g \text{ satisfaisable } \Leftrightarrow v \circ g \text{ satisfaisable, et si c'est le cas} \\ poids((x'y')_{u \circ g}) < -K \Leftrightarrow poids((x'y')_{v \circ g}) < -K \\ poids((x'y')_{u \circ g}) > K \Leftrightarrow poids((x'y')_{v \circ g}) > K \\ poids((x'y')_{u \circ g}) \in [-K, K] \Rightarrow (x'y')_{u \circ g} \lhd\!\rhd (x'y')_{v \circ t} \end{cases}$$

Remarque 32 *Dans le cas où $g = g_\epsilon$ on a :*

$$\begin{cases} u \sim_1 v \Rightarrow \forall x, y \in V, (xy)_u \lhd\!\rhd (xy)_v \\ u \sim_2 v \Rightarrow (\forall x, y \in V, poids((x'y')_u) \in [-K, K] \Rightarrow (x'y')_u \lhd\!\rhd (x'y')_v) \end{cases}$$

C'est à dire que $\sim_1$ préserve les chemins (xy), alors que la relation $\sim_2$ tronque les poids des chemins $(x'y')$ dans $[-K, K]$.

Ces deux relations permettent déjà un certain nombre de manipulations sur les Tgraphes que nous allons détailler ici.

6.3.1 Propriétés de $\sim_1, \sim_2$

On montre ici que sous certaines conditions deux Tgraphes qui sont en relation pour $\sim_1$ ou $\sim_2$ sont interchangeables.

Propriété 33 *Soient u, v, w des Tgraphes tels que $v \sim_1 w$. Soit γ un chemin de $u \circ v$ qui se décompose en $\gamma_1 \cdot \mu_1 \cdot \gamma_2 \cdots \mu_p \cdot \gamma_{p+1}$ selon u et v, tels que les γ_i soient dans u, et les μ_k soient des chemins minimaux. Alors pour tout k il existe $\mu'_k \lhd\!\rhd \mu_k$ dans $u \circ w$, tels que $\gamma_1 \cdot \mu'_1 \cdot \gamma_2 \cdots \mu'_p \cdot \gamma_{p+1}$ soit un chemin dans $u \cdot w$ de mêmes extrémités que γ et comparable.*

Preuve : Il s'agit simplement de l'application de la remarque 32 avec $v \sim_1 w$, la preuve se résume alors à la figure 6.5

□

Propriété 34 *Soient u, v, w des Tgraphes tels que $u \sim_2 w$. Soit γ un chemin dans $u \circ v$ d'extrémités $x \in V_i$, $y \in V_j$ qui se décompose en $\gamma_1 \cdot \mu_1 \cdot \gamma_2 \cdots \mu_p \cdot \gamma_{p+1}$ selon u et v, tels que les μ_k soient des chemins minimaux dans u de poids dans $[-K, K]$. On considère les translatés γ'_k des γ_k de $|w| - |u|$ dans $w \circ v$. Alors pour tout k il existe $\mu'_k \lhd\!\rhd \mu_k$, tels que $\gamma'_1 \cdot \mu'_1 \cdot \gamma'_2 \cdots \mu'_p \cdot \gamma'_{p+1}$ soit un chemin de x' à y' dans $w \circ v$ comparable à γ, où x' est l'élément de $V_{i-|u|+|w|}$ de même projection que x et y' celui de $V_{j-|u|+|w|}$ de même projection que y.*

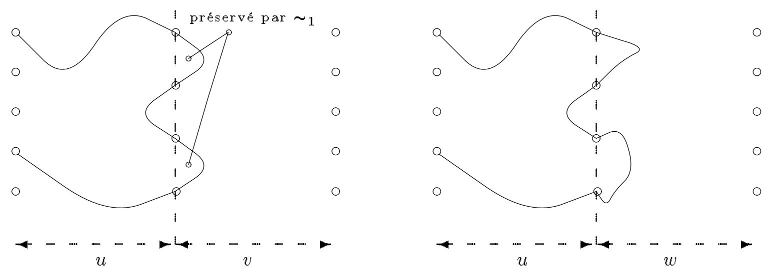


FIG. 6.5: *exemple de preservation par $\sim_1$*

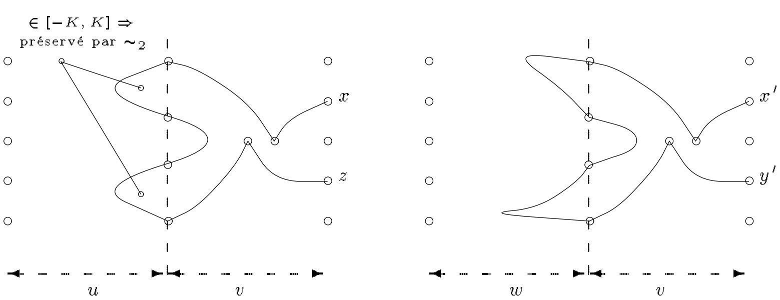


FIG. 6.6: *exemple de preservation par $\sim_2$*

Preuve :

La représentation de cette propriété est donnée par la figure 6.6. Elle découle simplement de l'application de $u \sim_2 w$ dans le cadre de la remarque 32. □

Remarque 35 *On peut en particulier appliquer la propriété 34 dans le cas où $u \circ v$ est satisfaisable, où γ est un minimum parmi les chemins de x à y, et γ_1 et γ_{p+1} passent chacun par des éléments de $\cup\tau^k$. En effet, dans ce cas les propriétés 14 et 15 assurent que $\forall n, poids(\mu_n) \in [-K, K]$.*

Propriété 36 *Soient u, v deux Tgraphes équivalents pour $\sim_2$. Pour tout Tgraphe w, on a $u \circ w$ satisfaisable ssi $v \circ w$ satisfaisable.*

Si u ou v est insatisfaisable, alors $u \sim_2 v$ implique qu'ils sont insatisfaisables tous les deux. Alors $u \circ w$ et $v \circ w$ sont tous deux insatisfaisables, c'est encore le cas si w est insatisfaisable.

Sinon, et puisque u et v jouent des rôles symétriques, il est suffisant de montrer que $u \circ w$ insatisfaisable implique que $v \circ w$ est insatisfaisable.

Soit $\gamma = \gamma_1 \cdot \mu_1 \cdots \gamma_p \cdot \mu_p$ la décomposition selon u et w d'un cycle insatisfaisable de $u \circ w$, où les μ_i sont des chemins dans u (Voir figure 6.7). On peut éventuellement remplacer les γ_i par leurs minimaux sans cycle de mêmes extrémités dans w puisqu'ils existent dès lors que w est satisfaisable. Cela assure que chaque γ_i passe par un sommet de $\bigcup \tau^k$.

On raisonne par récurrence sur p. Le cas $p = 0$ est celui où u ou w est insatisfaisable. On suppose donc que $p \geq 1$.

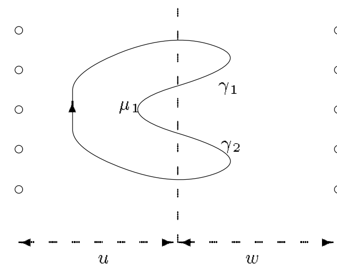


FIG. 6.7: *$u \sim_2 v$ et $u \circ w$ satisfaisable $\Leftrightarrow$ $v \circ w$ satisfaisable*

Si $\exists j, poids(\mu_j) < -K$, alors on est dans le cadre de la propriété 14 puisque γ_j passe par un élement de $\bigcup \tau^k$. Soient x et y les projetés de la source et de la destination de μ_i. Puisque $u \sim_2 v$, il existe $\mu'_j \in (x'y')_v$ de poids inférieur à $-K$. On peut encore appliquer la propriété 14 en utilisant le translaté de γ_j dans $w \circ v$, qui montre que $w \circ v$ est aussi insatisfaisable.

Si $\exists j, poids(\mu_j) > K$ alors la propriété 15 permet, en choisissant η sans arête de u de faire décroitre p.

Si $\forall j, poids(\mu_j) \in [-K, K]$ alors on peut appliquer la propriété 34, ce qui permet d'exhiber un cycle comparable dans $w \circ v$, et donc $w \circ v$ est insatisfaisable. $\square$

6.3.2 Anticipation

La propriété suivante montre que l'on peut prédire l'apparition d'une contrainte à partir d'éléments de $\mathcal{F}$. Une représentation de cette propriété est présentée par la figure 6.8. C'est principalement elle qui justifiera la préservation de $\sim_1$ par composition droite.

Propriété 37 *Soient u, v deux Tgraphes tels que $u \circ v$ soit satisfaisable. Pour tout sommet s_1, s_2 de u, il existe $g \in \mathcal{F}$ tel que $(s_1 s_2)_{u \circ v} \lhd\!\rhd (s_1 s_2)_{u \circ g}$.*

De plus pour tout w tel que $w \sim_2 u$, tout chemin entre des sommets de w dans $w \circ g$ a un chemin comparable de mêmes extrémités dans $w \circ v$.

Preuve : Soient u, v deux Tgraphes tels que $u \circ v$ satisfaisable, s_1, s_2 deux sommets de u, et γ un représentant de $(s_1 s_2)_{u \circ v}$.

Si γ est un chemin de u, il suffit de prendre $g = g_\epsilon$.

Sinon, on peut décomposer γ en $\gamma = \nu_1 \cdot \eta \cdot \nu_2$, où ν_1 et ν_2 sont dans u tandis que la première et dernière arêtes de η ne sont pas dans u. Soient y et y' les sommets respectivement de V_0 et V_1 dont la projection est égale à celle de la destination de ν_1.

η, comme sous chemin de $\gamma \in (s_1 s_2)_{u \circ w}$ est sans cycle, puisqu'il est aussi non vide, il passe par des sommets de $\bigcup_{k \geq |u|} \tau^k$. Soient τ^i et τ^j respectivement le premier et dernier passage de η par un τ^k, et e_1, e_2 les arcs correspondants. On décompose η en $\eta_0 \cdot e_1 \cdot \eta_1 \cdot e_2 \cdot \eta_2$ où $poids(\eta_0) = poids(\eta_2) = 0$ et $ope(\eta_0) = ope(\eta_2) =' \leq'$. Sur la figure 6.9 on distingue les cas où $i < j$ et $j < i$.

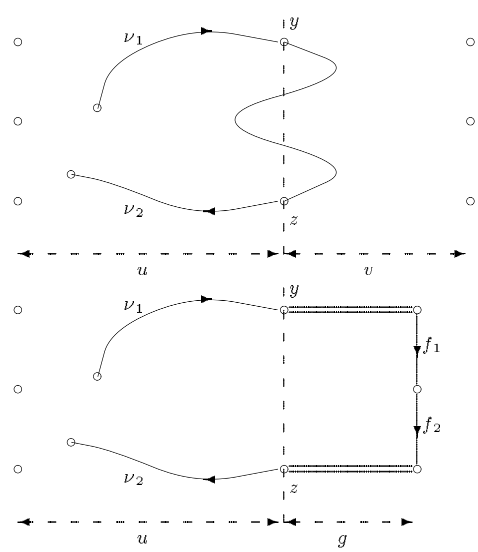


FIG. 6.8: *Illustration de la propriete 37*

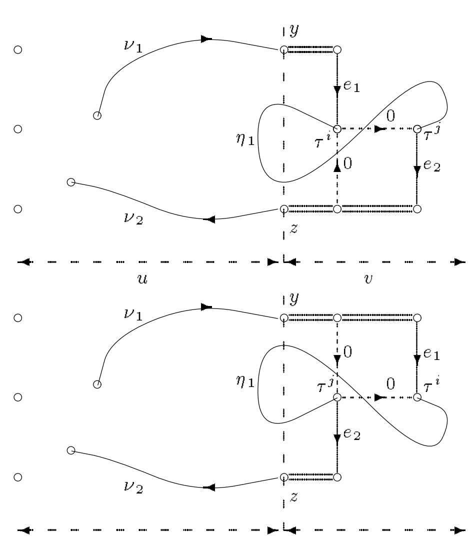


FIG. 6.9: *Illustration de la preuve de la propriete 37*

Soit $z \in V_1$ de même projection que la source de ν_2. Si $proj(z) = \tau$, on construit g en ajoutant à g_ϵ l'arc

$$f_1 : \text{de } y \text{ à } \tau^0, \text{ comparable à } e_1 \cdot \eta_1$$

Sinon, on construit g en ajoutant à g_ϵ les arcs :

$$\text{si } i < j \left[\begin{array}{l} f_1 : \text{de } y' \text{ à } \tau^1, \text{ comparable à } e_1 \\ f_2 : \text{de } \tau^1 \text{ à } v_k^1, \text{ comparable à } \eta_1 \cdot e_2 \end{array} \right.$$

$$\text{si } i > j \left[\begin{array}{l} f_1 : \text{de } y' \text{ à } \tau^1, \text{ comparable à } e_1 \cdot \eta_1 \\ f_2 : \text{de } \tau^1 \text{ à } v_k^1, \text{ comparable à } e_2 \end{array} \right.$$

On montre que g est bien un élément de $\mathcal{F}$:

- Si $i < j$: on a $poids(e_1) \in [-K, 0]$. On montre que $poids(\eta_1 \cdot e_2) \in [0, K]$. En effet d'après la propriété 11 il existe un chemin de poids nul de τ^i à τ^j, alors par minimalité de γ, on a $poids(\eta_1) \leq 0$, et donc $poids(\eta_1 \cdot e_2) \leq poids(e_2) \leq K$. D'autre part encore d'après la propriété 11 il existe un chemin de poids nul de y_2 à τ^i. Ce chemin forme avec $\eta_1 \cdot e_2$ un cycle de poids $poids(\eta_1 \cdot e_2)$ et, puisque $u \circ v$ satisfaisable, cela assure que $poids(\eta_1 \cdot e_2) \geq 0$.

- Si $i > j$, ou que $proj(z) = \tau$: on a $poids(e_2) \in [0, K]$. On montre que $poids(e_1 \cdot \eta_1) \in [-K, 0]$. En effet il existe un chemin de poids nul de y_1 à τ_j, alors par minimalité de γ on a $poids(e_1 \cdot \eta_1) \leq 0$. D'autre part puisqu'il existe un cycle de poids $poids(\eta_1)$ sur τ^i, on a $poids(\eta_1) \geq 0$, et donc $poids(e_1 \cdot \eta_1) \geq poids(e_1) \geq -K$.

Si l'on montre pour $w \sim_2 u$ qu'à tout chemin dans $w \circ g$ entre deux sommets de w on peut associer un chemin comparable dans $w \circ v$ de mêmes extrémités, on pourra conclure. En effet, dans le cas particulier où $w = u$ cette propriété assure que $u \circ g$ insatisfaisable $\Rightarrow u \circ v$ insatisfaisable, qui est contraire à l'hypothèse. Elle assure aussi que $(s_1 s_2)_{u \circ v} \trianglelefteq (s_1 s_2)_{u \circ g}$. Or par construction de g, on a l'inégalité inverse.

On note f'_1, f'_2 les translatés de f_1, f_2 dans $w \circ g$, et y^w, w^w les éléments de $V_{|w|}$ de même projection que y et z. Soit δ un chemin dans $w \circ g$ entre deux sommets z_1 et z_2 de w.

Si δ est dans w, alors il est aussi dans $w \circ v$, et le résultat est acquis.

Dans le cas contraire on décompose δ en $\alpha_1 \cdot \beta \cdot \alpha_2$, avec α_1 et α_2 les plus longs chemins strictement dans w.

- Si β ne passe ni par f'_1, ni par f'_2, alors il existe un chemin équivalent qui ne passe que par des arêtes de w.

- Si β passe par f'_1 alors il passe aussi par f'_2 et on peut caractériser exactement β qui ne peut être qu'un chemin entre y^w et z^w comparable à $f'_1 \cdot f'_2$. On a $\beta \lhd\!\rhd f'_1 \cdot f'_2 \lhd\!\rhd f_1 \cdot f_2 \lhd\!\rhd \eta$. Puisque γ est un minimum on est dans le cadre de la propriété 34. Il existe donc un chemin $\beta' \lhd\!\rhd \eta$ dans $w \circ v$ tel que $\alpha_1 \cdot \beta' \cdot \alpha_2 \lhd\!\rhd \delta$.

– Si β ne passe que par f_2' et non par f_1' on a $\beta \lhd\!\rhd f_2$. Reprenons alors la figure 6.9 par exemple dans le cas où $i < j$. D'une part il existe toujours un chemin μ de poids nul et d'opérateur $' \leq'$ de l'extrémité de α_1 à $\tau^{i+|w|-|u|}$, et d'autre part la propriété 34 s'applique pour $\eta_1 \cdot e_2$: il existe donc β' dans $w \circ v$ avec $\beta' \lhd\!\rhd \eta_1 \cdot e_2 \lhd\!\rhd f_2$ tel que $\alpha_1 \cdot \mu \cdot \beta' \cdot \alpha_2 \lhd\!\rhd \delta$ est dans $w \circ v$. Le cas $i > j$ est strictement similaire.

□

Remarque 38 *Cette construction reste valable si u est le graphe vide. Pour tout $x, y \in V$ elle donne alors un ensemble d'éléments de $\mathcal{F}(x, y)$ que l'on peut associer aux chemins de $(xy)_v$ quand v est satisfaisable. Alors on a $poids((xy)_v) \in [-K, K] \cap \mathbb{Z}$ puisque g ne contient au plus que deux arcs de poids non nuls qui sont aussi des entiers.*

Symétriquement il existe un résultat d'anticipation à gauche. Il diffère légèrement de la propriété 37 car il ne concerne que les chemins qui passent par $\bigcup \tau^k$ (voir figure 6.8) :

Propriété 39 *Soient u, v deux Tgraphes tels que $u \circ v$ soit satisfaisable, pour tout i, j il existe $g \in \mathcal{P}$ tel que $(\tau^{i+|u|}\tau^{j+|u|})_{u \circ v} \lhd\!\rhd (\tau^{i+1}\tau^{j+1})_{g \circ v}$. (Voir figure 6.10.)*

De plus pour tout $w \sim_1 v$, $\forall m, p$, pour tout chemin γ dans $g \circ w$ d'extrémités $x \in V_{m+1}$ et $y \in V_{p+1}$, il existe γ' dans $u \circ w$ comparable à γ d'extrémités $x' \in V_{m+|u|}$ et $y' \in V_{p+|u|}$ de même projection que x et y.

Preuve : La preuve est comparable à celle de la propriété 37, et est représentée sur la figure 6.11.

On décompose un chemin γ par ses premiers et derniers passages dans u par un élément de $\bigcup \tau^k$. On a $\gamma = \nu_1 \cdot \eta_0 \cdot e_1 \cdot \eta_1 \cdot e_2 \cdot \eta_2 \cdot \nu_2$ avec ν_1, ν_2 sans arête de u, η_0, η_2 dans u de poids nul et d'opérateur $' \leq'$, et e_1, e_2 qui sont les premiers et derniers arcs de γ dans u qui passent par des éléments de $\bigcup_k \tau^k$, respectivement τ^{k_1} et τ^{k_2}. On note $z \in V_0$ et $z' \in V_1$ les sommets dont le projeté est égal à celui de la source de η_2, et $t \in V_0$ dont le projeté est égal à celui de la source de η_0.

Si $t = \tau^0$, on définit g en ajoutant à g_ϵ l'arc :

$$f_2 : \text{ de } \tau^1 \text{ à } z' \text{ comparable à } \eta_1 \cdot e_2$$

Sinon, on définit g en ajoutant à g_ϵ les arcs :

$$\text{si } k_1 > k_2 : \left[\begin{array}{l} f_1 : \text{ de } t \text{ à } \tau^0 \text{ comparable à } e_1 \\ f_2 : \text{ de } \tau^0 \text{ à } z \text{ comparable à } \eta_1 \cdot e_2 \end{array} \right.$$

$$\text{si } k_1 < k_2 : \left[\begin{array}{l} f_1 : \text{ de } t \text{ à } \tau^0 \text{ comparable à } e_1 \cdot \eta_1 \\ f_2 : \text{ de } \tau^0 \text{ à } z \text{ comparable à } e_2 \end{array} \right.$$

Prenons le cas où $k_1 > k_2$. Puisqu'il existe un chemin de la destination de e_2 à τ^{k_1} de poids nul, et que $u \circ v$ est satisfaisable, c'est que $poids(f_2) \geq 0$. D'autre

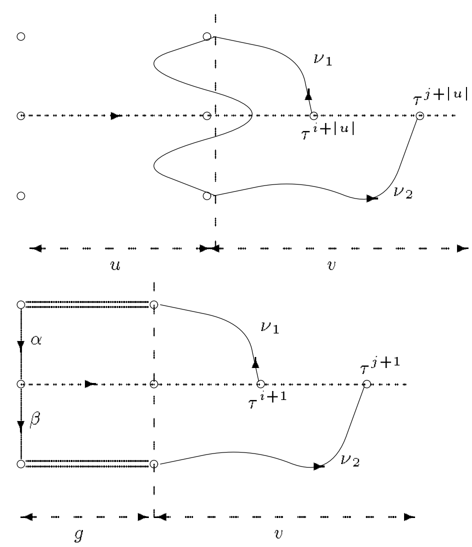


FIG. 6.10: *Illustration de la propriete 39*

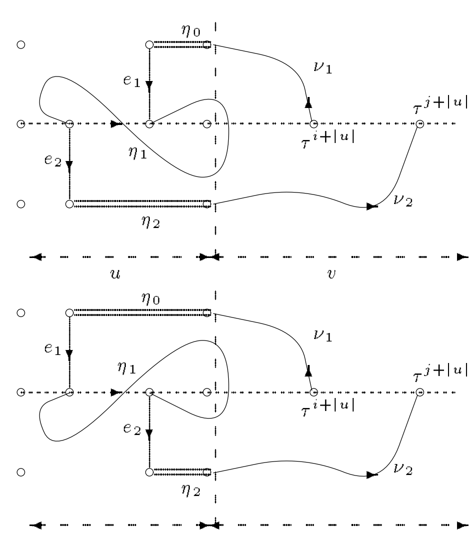


FIG. 6.11: *Illustration de la preuve de la propriete 39*

part, ν_2 passe par l'un des $\bigcup \tau^k$. Or il existe toujours un chemin de poids nul de τ^{k_1} à ce premier passage, ce qui met en concurrence deux chemins et assure par minimalité que $poids(\eta_1 \cdot e_2) \leq K$ par minimalité de γ.

Lorsque $k_1 < k_2$ on tient un raisonnement symétrique en considérant le dernier passage de ν_1 dans $\bigcup \tau^k$ pour justifier que $poids(f_1) \geq -K$. On a $poids(e_1) \leq 0$ et par minimalité $poids(\eta_1) \leq 0$ à cause du chemin concurrent de τ^{k_1} à τ^{k_2} de poids nul. Ces deux arguments assurent que $poids(e_1 \cdot \eta_1) \in [-K, 0]$.

La seconde partie de la propriété est similaire à la seconde partie de la propriéte 37. □

6.3.3 $\sim_1, \sim_2$ est une congruence droite

Propriété 40 *$\sim_1$ et $\sim_2$ sont d'indice $O(K^{|C|^4 * K^2})$*

Preuve : Par la propriété 38 les poids de chaque paire $(xy)_u$, s'ils sont définis, sont des entiers dans $[-K, K]$, tandis que $|\mathcal{F}| = O(|C|^2 * K^2)$. Pour $\sim_2$ les poids des paires $(x'y')$, s'ils sont définis, sont soit dans $[-K, K]$ soit tronqués supérieurement ou inférieurement. On garde donc le même ordre de grandeur. □

Lemme 9 *La relation d'équivalence conjonction de $\sim_1$ et $\sim_2$ est une congruence droite d'indice fini qui vérifie $\forall u, v, u \sim_1 v \wedge u \sim_2 v \Rightarrow u \circ v \sim_1 v \circ u$. (Et pas $u \circ v \sim_2 v \circ u$).*

Propriété 41 *Soient u, v deux Tgraphes équivalents pour $\sim_1$ et $\sim_2$. Pour tout Tgraphe w, on a $u \circ w \sim_1 v \circ w$*

Preuve :

La propriété 36 assure la condition $u \circ w \circ g$ satisfaisable $\Leftrightarrow$ $v \circ w \circ g$ satisfaisable pour tout $g \in \mathcal{F}$.

Soit $x, y \in V$, $g \in \mathcal{F}$, et γ un représentant de $(xy)_{u \circ w \circ g}$ lorsque $u \circ w \circ g$ est satisfaisable. Par la propriété 37 il existe un élément g' de $\mathcal{F}$ tel que $(xy)_{u \circ g'} \lhd \rhd (xy)_{u \circ w \circ g}$ et puisque $u \sim_2 v$, $(xy)_{v \circ w \circ g} \unlhd (xy)_{v \circ g'}$. Par définition de $u \sim_1 v$, on a $(xy)_{u \circ g'} \lhd\rhd (xy)_{v \circ g'}$ et finalement $(xy)_{v \circ w \circ g} \unlhd (xy)_{u \circ w \circ g}$. Alors puisque u et v jouent des rôles symétriques on a bien $u \circ w \sim_1 v \circ w$. □

Propriété 42 *Soient u, v deux Tgraphes équivalents pour $\sim_1$ et $\sim_2$. Pour tout 1-Tgraphe g, on a $u \circ g \sim_2 v \circ g$*

Preuve :

Soient $x, y \in V$, $t \in \mathcal{F}$, la propriété 36 nous permet de nous concentrer sur les cas où $u \circ g \circ t$ est satisfaisable. Soit alors γ un représentant de $(x'y')_{u \circ g \circ t}$. La preuve s'effectue par cas sur l'aspect de γ. Il suffit de montrer que l'on a chaque fois la propriété :

$$(P) : \begin{cases} poids((x'y')_{u \circ g \circ t}) < -K \Rightarrow poids((x'y')_{v \circ g \circ t}) < -K \\ poids((x'y')_{u \circ g \circ t}) \in [-K, K] \Rightarrow (x'y')_{v \circ g \circ t} \unlhd (x'y')_{u \circ g \circ t} \end{cases}$$

On pourra alors conclure que $u \circ g \sim_2 v \circ g$ puisque u et v jouent des rôles symétriques : en effet, l'étape 1 permet d'affirmer que $poids((x'z')_{uogot}) < -K \Leftrightarrow poids((x'z')_{vogot}) < -K$, et l'étape 2 exclut la possibilité d'avoir $poids((x'z')_{uogot}) \in [-K, K]$ et $poids((x'z')_{vogot}) > K$. Les ensembles $(x'z')_{uogot}$ et $(x'z')_{vogot}$ ont donc les mêmes types de troncatures. Dans le cas où leur poids est dans $[-K, K]$ l'étape 2 assure qu'ils sont comparables.

Lorsque γ n'a aucune arête dans u, sa représentation est celle de la figure 6.12a). Dans les autres cas, on peut décomposer γ en $\nu_1 \cdot \mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p \cdot \nu_2$ selon u et $g \circ t$ avec les μ_i dans u. Le cas b) de la figure 6.12 est celui où ν_1 et ν_2 passent par un τ^k. Les autres cas présentent les différentes combinaisons selon que ν_1 et ν_2 passent ou non par un τ^k. Le cas c) où $p \geq 2$ se distingue du cas d) où $p = 1$. Les cas $e), f)$ et $g), h)$ envisagent le cas où x ou y sont de projection τ.

a)b) : Que γ soit un chemin sans arête dans u, ou que ν_1 et ν_2 passent chacun par un τ^k, ces deux cas rentrent dans le cadre de la propriété 34. On a donc (P).

c)d) e)g) : Dans ces différents cas on exploite que dans les 1-Tgraphes tout sommet de $V_1 \setminus \tau^1$ est relié par des arcs d'égalité à un sommet de V_0. Le calcul de $(x'y')_{uogot}$ se ramène alors au calcul de chemins dans $u \circ g \cdot t$ minimums entre sommets de u. Par la propriété 37 il existe $t' \in \mathcal{F}$ qui se substitue à $g \circ t$. Par exemple dans le cas e), $(x's')_{uot'} \lhd\!\rhd (x'y')_{uogot}$ or $u \sim_2 v$ ainsi dans le cas où $poids((x's')_{uot'}) \in [-K, K]$, (respectivement inférieur à $-K$), on a $(x's')_{uot'} \lhd\!\rhd (x's')_{vot'}$, (respectivement $poids((x's')_{vot'}) < -K$) et puisque $v \sim_2 u$, la seconde partie de la propriété 37 prouve que $(x's')_{vogot} \trianglelefteq (x's')_{vot'}$. On a donc $(x'y')_{vogot} \trianglelefteq (x'y')_{uogot}$ (respectivement $poids((x'y')_{vogot}) < -K$), et donc la propriété (P).

g) : Si $Poids(\mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p) \in [-K, K]$, alors on est encore dans le cadre de l'utilisation de la propriété 34 qui permet d'obtenir un chemin comparable dans $v \circ g \circ t$.

La propriété 15 rend contradictoire les faits que ce poids soit plus grand que K, et que γ soit un minimum pour $\lhd$.

Si ce poids est $< -K$, on note z la source de ν_2. Il existe toujours un chemin de poids nul de z à $\tau^{|uogot|}$. C'est donc que $poids((x'y')_{uogot}) < -K$. On montre que l'on a aussi $poids((x'y')_{uogot}) < -K$. Soit $x^v, z^v \in V_{|v|}$ de même projection que x et z. Par $\sim_2$, et après avoir utilisé la propriété 37 entre x^v et z^v, on montre qu'il existe un chemin η de même troncature que $\mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p$ dans $v \circ g$ entre x^v et z^v. Ici encore il existe un chemin de poids 0 entre y^v et $\tau^{|vogot|}$, c'est donc que $poids((x'y')_{vogot}) < -K$. Et donc (P) est vérifiée.

h) : Dans ce cas de figure, puisque $u \circ g \circ t$ est satisfaisable, la propriété 14 assure que l'on ne peut avoir $poids(\mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p) < -K$.

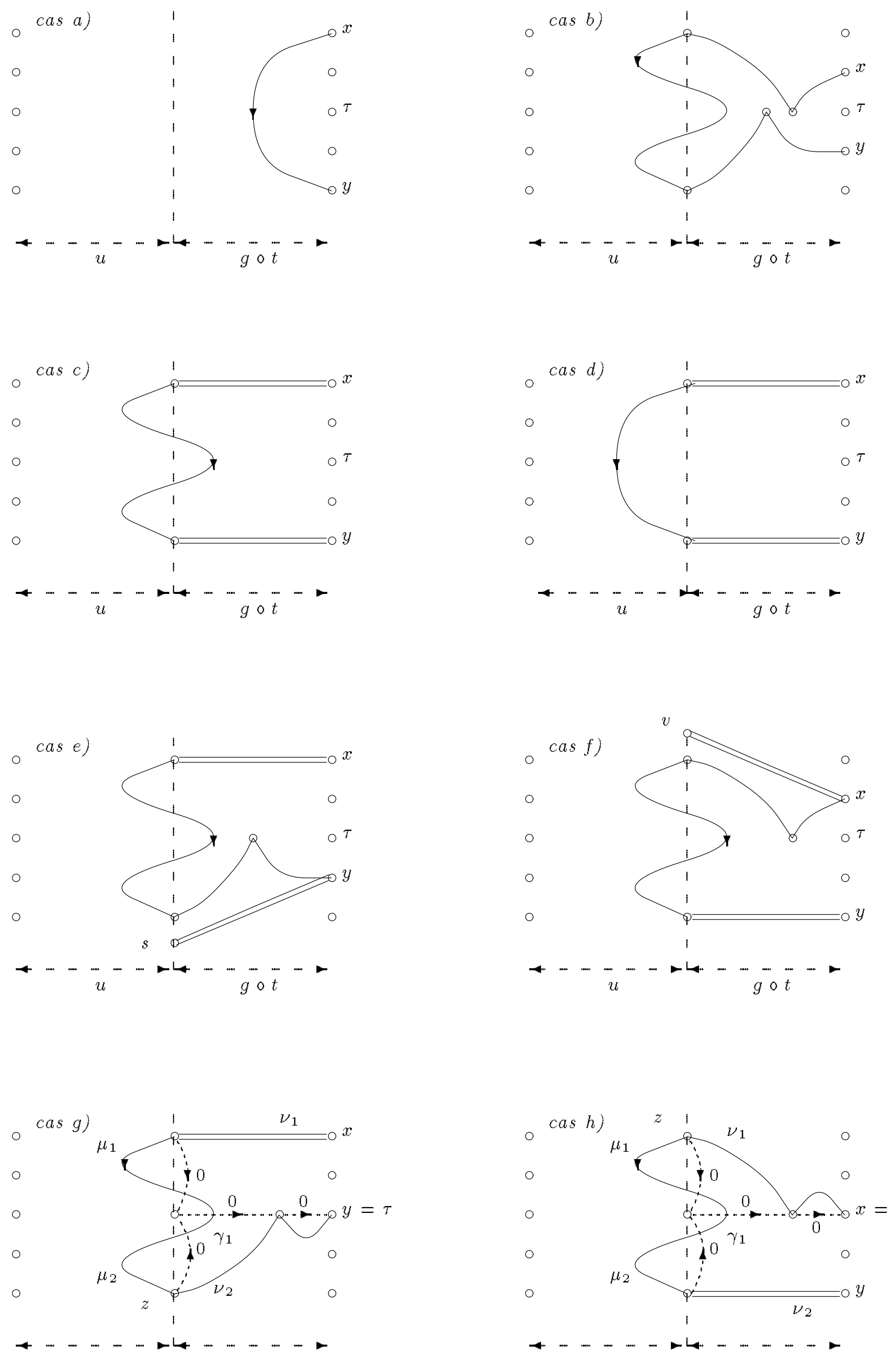


FIG. 6.12: *Les différentes décompositions de* $(x'z')_{u \cdot g}$

Si $poids(\mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p) > K$, on note z le sommet destination de ν_1. On a $poids(\nu_1) \geq 0$ à cause du cycle formé par ν_1 et du chemin de poids nul entre z et $\tau^{|u \circ g \circ t|}$. c'est donc que $poids(\gamma) > K$. (P) est alors trivialement vérifié.

Reste le cas où $poids(\mu_1 \cdot \gamma_1 \cdot \mu_2 \cdots \gamma_{p-1} \cdot \mu_p) \in [-K, K]$. Comme précédemment, on utilise la propriété 37 pour pouvoir faire valoir que $u \sim_2 v$. La seconde partie de la propriété 37 permet alors de montrer $(x'y')_{v \circ g \circ t} \trianglelefteq (x'y')_{u \circ g \circ t}$, qui implique (P).

□

Propriété 43 (commutation 1) *Soient u, v équivalents pour $\sim_1$ et $\sim_2$. On a $u \circ v \sim_1 v \circ u$.*

Preuve : En effet, soit $x, y \in V_0$, $g \in \mathcal{F}$ et γ un chemin minimum de x à y dans $u \circ v \circ g$ satisfaisable.

Si γ est un chemin de u, alors on a $(xy)_{u \circ v \circ g} \lhd\!\rhd (xy)_u \lhd\!\rhd (xy)_v \trianglerighteq (xy)_{v \circ u \circ g}$.

Sinon, puisque $u \circ g \sim_1 v \circ g$, par la propriété 33 il existe $\gamma' \lhd\!\rhd \gamma$ dans $u \circ u \circ g$. Par la propriété 37, il existe $g' \in \mathcal{F}$ tel que $(xy)_{u \circ g'} \lhd\!\rhd (xy)_{u \circ u \circ g} \trianglelefteq \gamma'$, mais par définition de $\sim_1$ on a $(xy)_{v \circ g'} \lhd\!\rhd (xy)_{u \circ g'}$. Puisque $u \sim_2 v$, la seconde partie de la propriété 37 permet d'obtenir que $(xy)_{v \circ u \circ g} \trianglelefteq (xy)_{v \circ g'}$. Finalement on a $(xy)_{v \circ u \circ g} \trianglelefteq (xy)_{u \circ v \circ g}$, et par symétrie $u \circ v \sim_1 v \circ u$. □

Remarque 44 *La figure 6.13 où l'on a $poids((x'\tau')_{u \circ v}) = -1$ et $poids((x'\tau')_{v \circ u}) = 0$, est un exemple où $u \circ v \not\sim_2 v \circ u$, avec pourtant $u \sim_1 v, u \sim_2 v$.*

6.4 La congruence complète

Les relations $\sim_1$ et $\sim_2$, outre le résultat de commutation que l'on a montré dans le lemme 9, ont un rôle systématique dans chacune des preuves qui suit. Ce n'est pas le cas des autres relations que nous introduisons à présent. Chacune d'entres elles est dédiée à la préservation par commutation d'un des types de chemin (xy'), $(x'y)$, ou $(x'y')$. Elles seront toutes non triviales et donc présentées séparément en même temps que leurs propriétés.

Nous aurons toutefois besoin d'une relation triviale $\sim_0$ qui, si l'on fait le parallèle avec les exécutions temporisées, indique l'ensemble des horloges qui ont été remises à zéro. Dans le cas des Tgraphes cela s'exprime par :

Définition 43 *La relation d'équivalence $\sim_0$ sur les Tgraphes se définit par :*

$$u \sim_0 v \stackrel{def}{=} \forall x \in V, \neg Reset(x, u) \Leftrightarrow \neg Reset(x, v)$$

où, si $x^0 \in V_0, x^u \in V_{|u|}$ sont de projection x :

$$\neg Reset(x, u) \stackrel{def}{=} \left\{ \begin{array}{l} \exists \gamma, \gamma' \text{ respectivement de } x^0 \text{ à } x^u \text{ et de } x^u \text{ à } x^0 \\ \text{de poids } 0 \text{ et d'opérateur } \leq \\ \text{qui ne contiennent aucun sommet de } \bigcup \tau^k \end{array} \right.$$

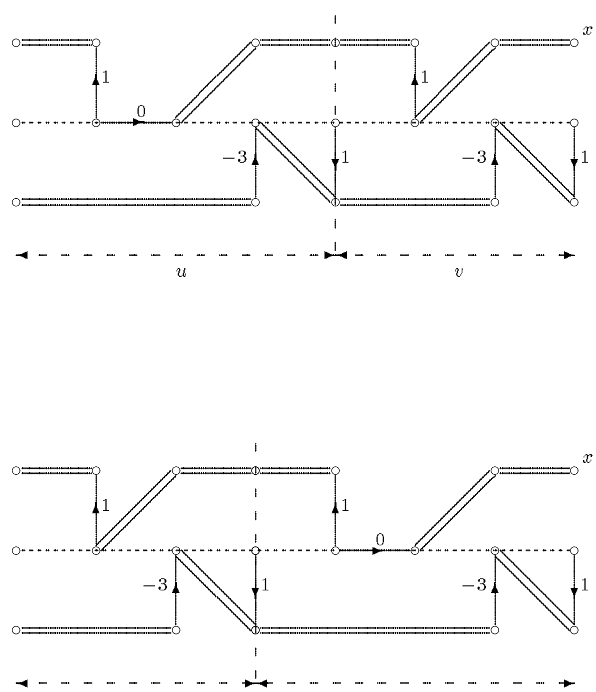


FIG. 6.13: $u \circ v \not\sim_2 v \circ u$

Les régularités sur les Tgraphes donnent trivialement :

Lemme 10 *La relation $\sim_0$ est une congruence droite d'indice $O(2^{|C|})$*

6.4.1 Préservation des chemins (xy')

Chaque Tgraphe g possède un chemin transverse particulier qui relie τ^0 à $\tau^{|g|}$. Les chemins de V_0 à $V_{|g|}$ qui s'en détachent contiennent une information que nous allons capturer grâce à $\sim_3$.

Définition 44 *La relation d'équivalence $\sim_3$ sur les Tgraphes se définit par $u \sim_3 v$ ssi $\forall x, y \in V, \forall g \in \mathcal{Q}(y)$, $u \circ g$ satisfaisable $\Leftrightarrow v \circ g$ satisfaisable et si c'est le cas, alors :*

$$\exists \gamma \text{ sans arête dans } u \text{ tel que } (xy')_u \cdot \gamma \subset (x\tau')_{u\circ g}$$

$$\Rightarrow \begin{cases} \exists \gamma' \lhd\!\rhd \gamma \text{ sans arête dans } v \text{ tel que } (xy')_v \cdot \gamma' \subset (x\tau')_{v\circ g} \\ ope((xy')_u) = ope((xy')_v) \\ poids((xy')_u) - poids((\tau\tau')_u) = poids((xy')_v) - poids((\tau\tau')_v) \end{cases}$$

Lemme 11 *La relation conjonction de $\sim_1$, $\sim_2$ et $\sim_3$ est une congruence droite d'indice fini. L'indice de $\sim_3$ est en $O(K^{|C|^2 * K})$.*

Propriété 45 *Soit u un Tgraphe, $x, y \in V$ et $g \in Q(y)$ tel qu'il existe γ sans arête de u avec $(x\tau')_u \cdot \gamma \subset (x\tau')_{u\circ g}$. Alors $poids((xz')_u) - poids((\tau\tau')_u) \in [-K, K]$.*

Preuve : La figure 6.14 permet de suivre cette démonstration.

Puisqu'il existe toujours un chemin de poids nul entre le sommet de V_0 de projection x et τ^0, et un autre entre $\tau^{|u|}$ et $\tau^{|u\circ g|}$, par minimalité de $(xy')_u \cdot \gamma$ on a $poids((xy')_u) + poids(\gamma) \leq 0 + poids((\tau\tau')_u) + 0$. Puisque $poids(\gamma)$ est égal à une somme de poids d'arêtes de g, il est dans $[-K, 0]$, on a donc $poids((xy')_u) - poids((\tau\tau')_u) \leq K$.

Si l'un des chemins de $(xy')_u$ ne passe par aucun $\bigcup \tau^k$ c'est que $poids((xy')_u) = 0$, et puisque $poids((\tau\tau')_u) \leq 0$ on a $poids((xy')_u) - poids((\tau\tau')_u) \geq 0$.

Sinon, on est dans le cadre de la figure 6.14. Soit β le poids du premier arc d'un représentant de $(xy')_u$ qui passe par un élément τ^i de $\bigcup \tau^k$ et η la suite de ce chemin. Il existe un chemin μ_0 dans u de τ^0 à τ^i de poids nul, et un chemin μ_1 du sommet de $V_{|u|}$ de projection y à $\tau^{|u|}$. On a $(\tau\tau')_u \trianglelefteq \mu_0 \cdot \eta \cdot \mu_1$, et donc $poids((\tau\tau')_u) \leq poids((xy')_u) - \beta$ qui assure que $poids((xy')_u) - poids((\tau\tau')_u) \geq -K$.

$\square$

Cette propriété permet alors de calculer l'indice de $\sim_3$: puisque les poids sont aussi des entiers, et que $|Q(y)|$ est en $O(K * |C|)$ on a :

Corollaire 46 *L'indice de la relation $\sim_3$ est $O(K^{|C|^2 * K})$.*

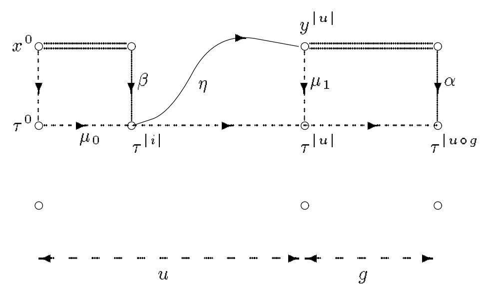


FIG. 6.14: $\sim_3$ *est d'indice fini*

La propriété suivante montre que si l'on considère un chemin transverse dans un graphe $u \circ w$ qui traverse u avec un certain chemin, alors le même découpage a aussi cours dans les graphes $v \circ w$ pourvu que u et v soient équivalents.

Propriété 47 *Soient $u \sim_2 v$ et $u \sim_3 v$, et w tels que $u \circ w$ soit satisfaisable. Pour tout $x, y_1, y_2 \in V$, s'il existe δ et η dont la première arête n'est pas dans u ni dans v, tels que l'on ait $(x\tau')_{u \circ w} = (xy_1')_u \cdot \delta$ et $(x\tau')_{v \circ w} = (xy_2')_v \cdot \eta$, alors :*

$$\begin{cases} ope((xy_1')_u) = ope((xy_1')_v) \\ poids((xy_2')_u)) - poids((xy_1')_u) = poids((xy_2')_v)) - poids((xy_1')_v) \\ \exists \eta' \lhd\!\rhd \eta \text{ tel que } (xy_1')_u \cdot \delta \lhd\!\rhd (xy_2')_u \cdot \eta' \end{cases}$$

Preuve : Voir la figure 6.15.

Soit δ_1 le plus court préfixe de δ qui passe par $\bigcup \tau^k$. En considérant l'élément de $\mathcal{Q}(y_1)$ dont l'arc significatif est comparable à δ_1, on rentre dans le cadre des conditions de préservation par $\sim_3$. Alors on a :

$$\begin{cases} ope((xy_1')_u) = ope((xy_1')_v) \\ poids((xy_1')_u)) - poids((\tau\tau')_u) = poids((xy_1')_v)) - poids((\tau\tau')_v) \end{cases}$$

Et symétriquement :

$$\begin{cases} ope((xy_2')_v) = ope((xy_2')_u) \\ poids((xy_2')_v)) - poids((\tau\tau')_v) = poids((xy_2')_u)) - poids((\tau\tau')_u) \end{cases}$$

On en déduit que :

$$poids((xy_2')_u)) - poids((xy_1')_u) = poids((xy_2')_v)) - poids((xy_1')_v)$$

Puisque $u \sim_2 v$ on peut appliquer la propriété 34 pour exhiber un chemin $\eta' \lhd\!\rhd \eta$ dans $u \circ w$, du sommet de $V_{|u|}$ de projection y_2, à $\tau^{|u \circ w|}$, et un chemin $\delta' \lhd\!\rhd \delta$ dans $v \circ w$, du sommet de $V_{|v|}$ de projection y_1, à $\tau^{|v \circ w|}$.

Par minimalité dans $v \circ w$, on a $(xy_2')_v \cdot \eta \unlhd (xy_1')_v \cdot \delta'$, en utilisant alors les invariants précédents des poids et des opérateurs, on a aussi $(xy_2')_u \cdot \eta' \unlhd (xy_1')_u \cdot \delta$. Mais $(xy_1')_u \cdot \delta$ est un minimum, c'est donc que $(xy_2')_u \cdot \eta' \lhd\!\rhd (xy_1')_u \cdot \delta$
□

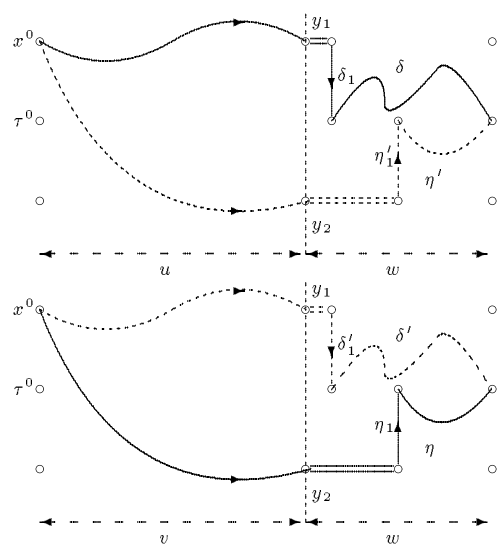


FIG. 6.15: *Propriété de* $\sim_3$

On en déduit que :

Corollaire 48 *La relation composée de* $\sim_1$, $\sim_2$, *et* $\sim_3$ *est une congruence droite.*

Preuve : (voir figure 6.16)

Soient u et v équivalents pour $\sim_1, \sim_2, \sim_3$, w un Tgraphe, $t \in V$ et $g \in \mathcal{Q}(t)$ tel que $u \circ w \circ g$ est satisfaisable, et qu'il existe un chemin γ tel que $(xt')_{u \circ w} \cdot \gamma \in (x\tau')_{u \circ w \circ g}$. Un représentant particulier de $(xt')_{u \circ w} \cdot \gamma$ peut se décomposer en $(xy')_u \cdot \delta \cdot \gamma$ pour un certain $y \in V$. Par la propriété 47 il existe $\mu \lhd\!\rhd \delta \cdot \gamma$ tel que $(xy')_v \cdot \mu \in (x\tau')_{v \circ w \circ g}$. Or puisque $u \sim_2 v$ la propriété 34 permet de trouver δ' et γ' comparables à δ et γ, avec δ' entre les sommets de $V_{|v|}$ et de $V_{|v \circ w|}$ de projection y et t, et γ', sans arête de $v \circ w$, entre le sommet de $V_{|v \circ w|}$ de projection t et $\tau^{|v \circ w \circ g|}$. Alors $\delta' \cdot \gamma' \lhd\!\rhd \delta \cdot \gamma \lhd\!\rhd \mu$, ce qui assure que $(xt')_{v \circ w} \cdot \gamma' \in (x\tau')_{v \circ w \circ g}$ avec γ' sans arête de $v \circ w$, c'est à dire que la condition de préservation pour $\sim_3$ est aussi respectée dans $v \circ w$.

La propriété 47 assure aussi que $(xy')_u$ et $(xy')_v$ ont même opérateur. Puisque $\delta \lhd\!\rhd \delta'$, les ensembles $(xt')_{u \circ w}$ et $(xt')_{v \circ w}$ ont aussi les mêmes opérateurs.

Mais aussi :

$$poids((xy')_u) - poids((\tau\tau')_u) = poids((xy')_v) - poids((\tau\tau')_v)$$

Considérons à présent un chemin dans $(\tau\tau')_{u \circ w \circ g}$. Il existe $z \in V$ et η tels que ce chemin se décompose en $(\tau z')_u \cdot \eta$. Par la propriété 47, il existe $\eta' \lhd\!\rhd \eta$ dans $v \circ w \circ g$ tel que $(\tau z')_v \cdot \eta' \in (\tau\tau')_{v \circ w \circ g}$. Alors on a :

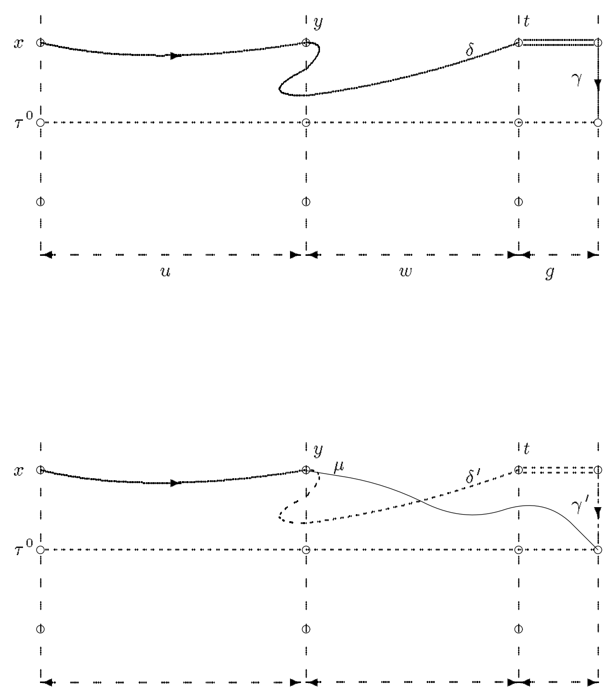


FIG. 6.16: $\sim_3$ *est une congruence droite*

$$poids((\tau z')_u) - poids((\tau\tau')_u) = poids((\tau z')_v) - poids((\tau\tau')_v)$$

et puisque $\eta \lhd\!\rhd \eta'$ et $\delta \lhd\!\rhd \delta'$, on a :

$$poids((xt')_{u\circ w}) - poids((\tau\tau')_{u\circ w}) = poids((xy')_{v\circ w}) - poids((\tau\tau')_{v\circ w})$$

Ce qui achève la preuve de la propriété. □

6.4.2 Préservation des cheminss $(x'y)$

De la même façon, on cherche à comparer deux chemins transverses d'un Tgraphe g, entre $V_{|g|}$ et V_0. Mais, dans ce sens, il n'existe pas de chemin de référence similaire à $(\tau\tau')_g$. La définition de $\sim_4$ est donc un peu plus technique car elle préserve les différences entre tous les chemins transverses de $V_{|g|}$ à V_0 (partie (2) de la définition) de poids, dès lors qu'ils vérifient une condition de mise en concurrence (partie (2) de la définition).

Définition 45 *La relation d'équivalence $\sim_4$ sur les Tgraphes se définit par $u \sim_4 v$ ssi les deux conditions suivantes sont réalisées :*

$$(1)\left\{\begin{array}{l} \forall x,y,z \in V, \forall g \in \mathcal{F}(x,z), \mathcal{U}(x,y,z,g,u) \Leftrightarrow \mathcal{U}(x,y,z,g,v) \text{ où } : \\ \mathcal{U}(x,y,z,g,u) \overset{def}{=} \left\{\begin{array}{l} u\circ g \text{ satisfaisable} \\ poids((x'\tau')_u) \geq -K \\ (xy')_u \lhd\!\rhd (xy')_{u\circ g} \end{array}\right. \end{array}\right.$$

$$(2)\left\{\begin{array}{l} \forall x,y,z,s \in V, \exists g_1 \in \mathcal{F}(x,z), g_2 \in \mathcal{F}(z,x), \mathcal{U}(x,y,z,g_1,u) \wedge \mathcal{U}(z,s,x,g_2,u) \\ \Rightarrow \left\{\begin{array}{l} ope((x'y)_u) = ope((x'y)_v) \\ ope((z's)_u) = ope((z's)_v) \\ poids((x'y)_u) - poids((z's)_u) = poids((x'y)_v) - poids((z's)_v) \end{array}\right. \end{array}\right.$$

La partie (1) teste si les chemins $(x'y)_g$ sont éventuellement utiles pour composer un chemin transverse, au regard d'une exécution future. La partie (2) assure que si deux chemins $(x'y)_g$ et $(z't)_g$ peuvent être tous deux utiles pour un trajet transverse, alors leurs opérateurs et leur différence de poids sont préservés. (Voir figure 6.17)

Propriété 49 *Soient $u \sim_4 v$ satisfaisables, et $x,y,z,s \in V, g_1 \in \mathcal{F}(x,z), g_2 \in \mathcal{F}(z,x)$ tels que $\mathcal{U}(x,y,z,g_1,u)$ et $\mathcal{U}(z,s,x,g_2,u)$. On a alors*

$$poids((x'y)_u) - poids((z's)_u) \in [-2K, 2K]$$

Preuve :

On montre d'abord que $\forall a,b \in V$ on a $poids((a'b)_u) \geq -K$. En effet, si un représentant γ de $(a'b)_u$ ne passe par aucun sommet de $\bigcup \tau^k$, on a $poids(\gamma) = 0$. Sinon, soit γ_1 le plus petit préfixe de γ qui contienne un sommet τ^i de $\bigcup \tau^k$, et γ_2 le suffixe correspondant.(Le premier dessin de la figure 6.18). Il existe

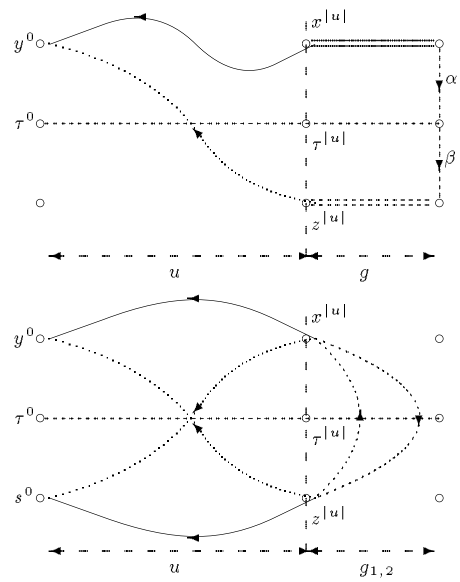


Fig. 6.17: *Définition de* $\sim_4$

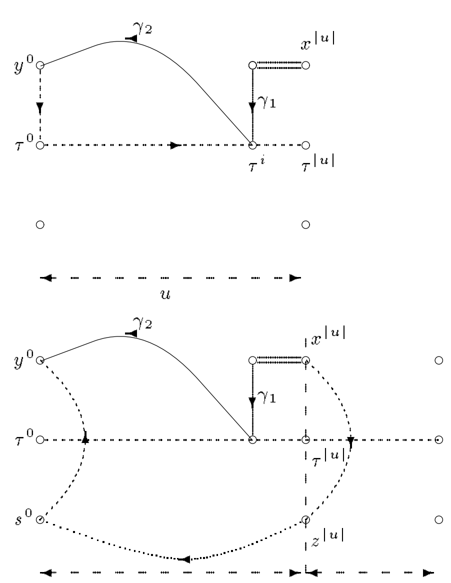


Fig. 6.18: $\sim_4$ *est d'indice fini*

un chemin de poids 0, du sommet de V_0 de projection a, à τ^i, qui forme un cycle avec γ_2. Puisque u est satisfaisable ce cycle est de poids positif, et donc $poids(\gamma_2) \geq 0$. D'autre part le choix de γ_1 impose $poids(\gamma_1) \in [-K, 0]$, ce qui assure $poids((a'b)_u) \geq -K$.

On montre à présent que $poids((x'y)_u) - poids((z's)_u \leq 2K$, la propriété s'en déduira par symétrie des rôles joués par x, y et z, s.

S'il existe un représentant de $(x'y)_u$ qui ne contienne aucun $\bigcup \tau^k$, alors son poids est nul. Mais puisque $poids((z's)_u) \geq -K$, on a bien $poids((x'y)_u) - poids((z's)_u \leq K \leq 2K$.

Sinon, nous sommes dans le cas du second dessin de la figure 6.18. C'est à dire qu'un représentant γ de $(x'y)_u$ se décompose en $\gamma_1 \cdot \gamma_2$, de sorte que la source de γ_2 soit un τ^i. Mais alors $(sy)_u$ est non vide, parce qu'il existe au moins un chemin du sommet de V_0 de projection s à τ^i, et qu'alors γ_2 complète le chemin pour rejoindre le sommet de V_0 de projection y. D'un autre côté on a $(x'y)_u \trianglelefteq (x'y)_{u \circ g_1}$. Soit μ un chemin de $u \circ g_1$ sans arête de u qui relie les deux sommets de $V_{|u \circ g_1|}$ de projection x et z. On a $(x'y)_u \trianglelefteq \mu \cdot (z's)_u \cdot (s_y)_u$. On a $poids(\mu) \in [-K, K]$, et par la propriété 38 on a aussi $poids((sy)_u) \in [-K, K]$. Ce qui assure que $poids((x'y)_u) - poids((z's)_u \leq 2K$. □

Corollaire 50 $\sim_4$ *est d'indice* $O(K^{|C|^4 K^4})$

On cherche à montrer à présent une propriété similaire à celle de la propriété 47 pour $\sim_3$, c'est-à-dire à imposer un découpage d'un chemin transverse indépendant du représentant $u \sim_4 v$.

Propriété 51 *Soient* u, v, w *des Tgraphes tels que* $u \sim_2 v$ *et* $u \sim_4 v$.
$\forall x, y, z \in V_0, g \in \mathcal{F}(x, z)$ *tels que* $\mathcal{U}(x, y, z, g, u \circ w)$ *et* $(x'y)_{u \circ w}$ *non vide, il existe* $x_1 \in V$ *tel que :*

$$\begin{cases} \exists \delta \text{ avec } \delta \cdot (x_1'y)_u \in (xy')_{u \circ w}, \\ \exists \delta' \Leftrightarrow \delta \text{ tel que } \delta' \cdot (x_1'y)_v \in (x'y)_{v \circ w} \\ (x'y)_{v \circ w} \trianglelefteq (x'y)_{v \circ w \circ g} \end{cases}$$

Preuve : Elle est illustrée par la figure 6.19.

Prenons deux représentants de $(x'y)_{u \circ w}$ et de $(x'y)_{v \circ w \circ g}$, respectivement $\delta \cdot (x_1'y)_u$ et $\eta \cdot (x_2'y)_v$, lorsque $(x'y)_{u \circ w} \Leftrightarrow (x'y)_{u \circ w \circ g}$, et avec $x_1 \neq x_2$.

a) Puisque $u \sim_2 v$ et $(x'\tau')_{v \circ w} \geq -K$, on peut appliquer la propriété 34. Il existe donc $\eta' \Leftrightarrow \eta$ dans $u \circ w \circ g$ tel que $\eta' \cdot (x_2'y)_u$ ait les mêmes extrémités que $(x'y)_{u \circ w \circ g}$. On a donc : $\delta \cdot (x_1'y)_u \trianglelefteq \eta' \cdot (x_2'y)_u$

b) De même, il existe $\delta' \Leftrightarrow \delta$ dans $v \circ w$ tel que $\delta' \cdot (x_1'y)_v$ ait les mêmes extrémités que $(x'y)_{v \circ w}$. Puisqu'il existe un chemin de poids nul entre les sommets de $V_{v \circ w \circ g}$ et de $V_{v \circ w}$ de projection x, on a : $\eta \cdot (x_2'y)_v \trianglelefteq \mu' \cdot (x_1'y)_v$.

c) Si δ ne passe pas par $\bigcup \tau^k$, alors il existe un chemin en sens inverse de δ, δ^{-1}, de poids nul qui forme avec η' un chemin $\delta^{-1} \cdot \eta'$ dont la première et dernière arêtes ne sont pas dans u. D'après la remarque 38, on peut

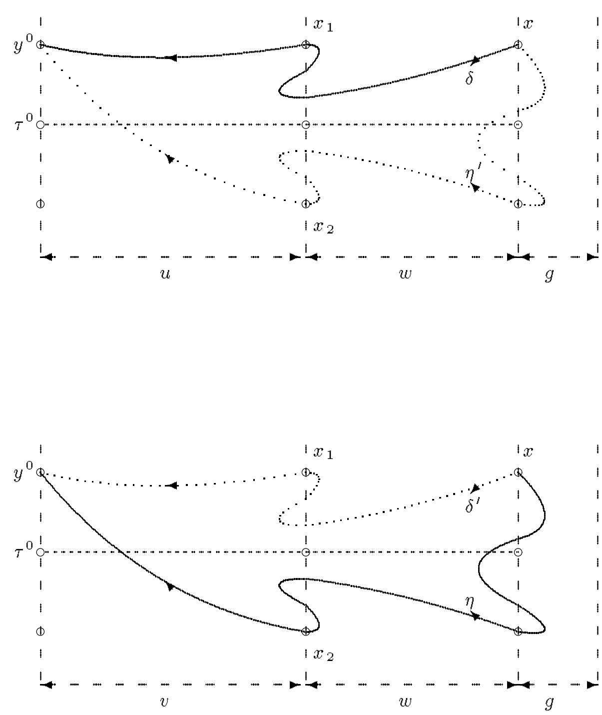


FIG. 6.19: *Découpage identique pour* $\sim_4$

lui associer un élément g_1 de $\mathcal{F}(x, x_2)$ tel que $\mathcal{U}(x, y, x_2, g_1, u)$, et puisque que $u \sim_4 v$ on a alors $\mathcal{U}(x, y, x_2, g_1, v)$. Si l'on considère alors δ', on a $\delta' \cdot (x_1'y)_v \in (x'y)_{v \circ w \circ g} \lhd\!\rhd \eta \cdot (x_2'y)_v$. C'est donc que l'on peut prendre x_1 pour une décomposition commune.

d) Si c'est η qui ne passe pas par $\bigcup \tau^k$, le même raisonnement permet de prendre $x_2 = x$ au lieu de x_1.

e) Si δ et η passent par $\bigcup \tau^k$ on montre que puisque $u \sim_4 v$, on a conservé les opérateurs des chemins $(x_1'y)_u$, $(x_2'y)_u$ et leur différence de poids. Ainsi, avec les chemins $\delta' \lhd\!\rhd \delta$ et $\eta' \lhd\!\rhd \eta$ construits en a) et b) on peut montrer que le choix de x_1 ou de x_2 est indifférent.

δ (resp. η) passe par $\bigcup \tau^k$ alors la propriété 14 assure que $poids((x_1'\tau')_u) \geq -K$ (resp. $poids((x_2'\tau')_v) \geq -K$). Puisque $u \sim_2 v$ les conditions $poids((x_1'\tau')_v) \geq -K$ et $poids((x_2'\tau')_u) \geq -K$ sont vérifiées.

Puisque δ et η passent par $\bigcup \tau^k$, alors $(x_1x_2)_{w \circ g}$ et $(x_2x_1)_{w \circ g}$ existent. On considère alors les éléments g_1 de $\mathcal{F}(x_1, x_2)$ et g_2 de $\mathcal{F}(x_2, x_1)$ associés, au sens de la remarque 38, aux chemins $(x_1x_2)_{w \circ g}$ et $(x_2x_1)_{w \circ g}$. On a bien $\mathcal{U}(x_1, y, x_2, g_1, u)$ et $\mathcal{U}(x_2, y, x_1, g_2, u)$.

□La propriété 36, le lemme 9 et la propriété 51 permettent de montrer respectivement la préservation des trois sous cas de la définition de $\mathcal{U}$ par échange de u par v s'ils sont équivalents pour $\sim_2$ et $\sim_4$:

Corollaire 52 *Si $u \sim_2 v$ et $u \sim_4 v$, alors* $\forall x, y, z \in V, g \in \mathcal{F}(x, z), \mathcal{U}(x, y, z, g, u \circ w) \Leftrightarrow \mathcal{U}(x, y, z, g, v \circ w)$

Lemme 12 *La relation composée de $\sim_4$ et de $\sim_2$ est une congruence droite. L'indice de $\sim_4$ est $O(K^{K^4|C|^4})$.*

Preuve : L'ordre de grandeur de l'indice de $\sim_4$ a été démontré dans le corollaire 50. Il reste à montrer que l'on a une congruence droite. Le corollaire 52 permet de nous concentrer sur la seconde partie de la définition de $\sim_4$. On peut suivre la preuve sur la figure 6.20.

Soient u, v, w tels que $u \sim_2 v$, $u \sim_4 v$ et x, y, z, s, g_1, g_2 tels que $\mathcal{U}(x, y, z, g_1, u \circ w)$ et $\mathcal{U}(z, s, x, g_2, u \circ w)$. La propriété 51 indique qu'il existe x_1, δ, δ' et z_1, η, η' tels que $\delta \lhd\!\rhd \delta'$, $\eta \lhd\!\rhd \eta'$, et $\delta \cdot (x_1'y)_u \subset (x'y)_{u \circ w}$, $\delta' \cdot (x_1'y)_v \subset (x'y)_{v \circ w}$, et $\delta \cdot (z_1's)_u \subset (z's)_{u \circ w}$, $\delta' \cdot (z_1's)_v \subset (z's)_{v \circ w}$.

Grâce à g_1 et que η' passe ou non par $\bigcup \tau^k$, il existe toujours un chemin dans $(x_1z_1)_{w \circ g_1}$ (resp. dans $(z_1x_1)_{w \circ g_2}$ en considérant g_2 et si δ passe par $\bigcup \tau^k$ ou non). On pose g_1' (resp. g_2') l'élément de $\mathcal{F}(x_1, z_1)$ (resp $\mathcal{F}(z_1, x_1)$) d'anticipation au sens de la propriété 37 pour $w \circ g_1$ (resp. $w \circ g_2$). On a $\mathcal{U}(x_1, y, z_1, g_1', u)$ (resp. $\mathcal{U}(x_1, y, z_1, g_1', u)$).

On a donc $ope((x_1'y)_u) = ope((x_1'y)_v), ope((z_1's)_u) = ope((z_1's)_v)$ et $poids((x_1'y)_u) - poids((z_1's)_u) = poids((x_1'y)_v) - poids((z_1's)_v)$. Et puisque $\delta' \lhd\!\rhd \delta$ et $\eta' \lhd\!\rhd \eta$, on a bien : $ope((x'y)_{u \circ w}) = ope((x'y)_{v \circ w}), ope((z's)_{u \circ w}) =$

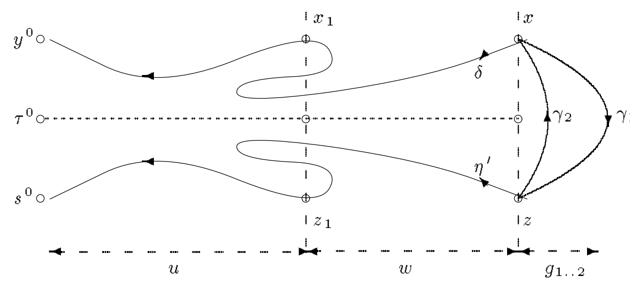


FIG. 6.20: $\sim_4$ *est une congruence droite*

$ope((z's)_{v\circ w})$ et $poids((x'y)_{u\circ w}) - poids((z's)_{u\circ w}) = poids((x'y)_{v\circ w}) - poids((z's)_{v\circ w})$.

Ce qui prouve que $u \circ w \sim_4 v \circ w$. $\square$

6.4.3 Préservation des chemins $(x'y')$

La relation $\sim_2$ permet déjà une forme de préservation de ces contraintes. Elle est toutefois insuffisante puisque nous avons vu lors de la remarque 44 que l'on n'a pas nécessairement $u \circ v \sim_2 v \circ u$. Intuitivement cela est dû au fait que $\sim_2$ ne prend pas en compte des contraintes induites par les passés possibles. Ainsi lorsque l'on commute u et v dans $u \circ v \circ w$, l'influence de u doit s'adapter à celle de v qui évolue dans un contexte avec passé et futur. C'est la raison de l'introduction d'une équivalence supplémentaire $\sim_5$

Sa définition est illustrée par la figure 6.21

Définition 46 *La relation d'équivalence $\sim_5$ sur les Tgraphes se définit par $u \sim_5 v$ ssi u et v sont insatisfaisables, ou lorsque les deux conditions suivantes sont réalisées :*

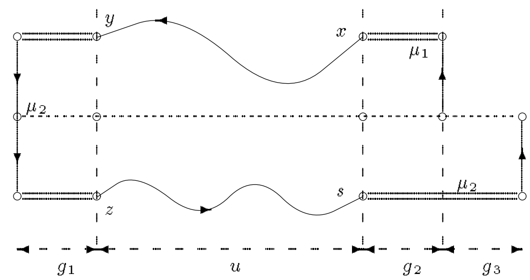


FIG. 6.21: *définition de* $\sim_5$

$$(1)\left\{\begin{array}{l}\forall x,y,z,s\in V,\forall g_1\in\mathcal{P}(y,z),\forall(g_2,g_3)\in\mathcal{Q}(s)\times\mathcal{R}(x)\cup\mathcal{R}(x)\times\mathcal{Q}(s),\\ \mathcal{M}(x,y,z,s,g_1,g_2,g_3,u)\Leftrightarrow\mathcal{M}(x,y,z,s,g_1,g_2,g_3,v)\\ \quad \textit{où on définit } \mathcal{M}(x,y,z,s,g_1,g_2,g_3,u) \textit{enposant :}\\ \textit{Si } g_2\in\mathcal{R},\ \textit{on pose } i=|g_1\circ u\circ g_2| \textit{ et } j=|g_1\circ u\circ g_2\circ g_3|.\\ \textit{Si } g_3\in\mathcal{R},\ \textit{on pose } i=|g_1\circ u\circ g_2\circ g_3| \textit{ et } j=|g_1\circ u\circ g_2|.\\ \textit{on note } x^{g_1\circ u} \textit{ et } s^{g_1\circ u} \textit{ les sommets de } V_{|g_1\circ u|} \textit{ de projection respective } x \textit{ et } s\\ \mu_1 \textit{ est le plus court chemin de } \tau_i \textit{ à } x^{|g_1\circ u|}\\ \mu_2\in(yz)_{g_1}\\ \mu_3 \textit{ est un chemin minimal sans arête de } g_1\circ u \textit{ entre } s^{|g_1\circ u|} \textit{ et } \tau^j\\ \textit{alors :}\\ \mathcal{M}(x,y,z,s,g_1,g_2,g_3,u)\stackrel{def}{=}\mu_1\cdot(x'y)_{g_1\circ u}\cdot\mu_2\cdot(zs')_{g_1\circ u}\cdot\mu_3\subset(\tau^i\tau_j)_{g_1\circ u\circ g_2\circ g_3}\end{array}\right.$$

$$(2)\left\{\begin{array}{l}\mathcal{M}(x,y,z,s,g_1,g_2,g_3,u)\Rightarrow\\ \quad\left[\begin{array}{l} poids((x'y)_u)+poids((zs')_u)=poids((x'y)_v)+poids((zs')_v)\\ min\_ope(ope((x'y)_u),ope((zs')_u))=min\_ope(ope((x'y)_v),ope((zs')_v))\end{array}\right.\end{array}\right.$$

Si la définition est technique les preuves suivantes s'en trouvent simplifiées :

Propriété 53 *$\mathcal{M}(x,y,z,s,g_1,g_2,g_3,u) \Rightarrow poids((x'y)_u) + poids((zs')_u) \in [-2K,2K]$, et donc $\sim_5$ est d'indice $O(K^{|C|^4K^4})$*

Preuve :

On peut suivre cette démonstration sur la figure 6.21.

Puisque μ_1 et μ_3 passent par $\bigcup\tau^k$, les propriétés 14 et 15 assurent que $poids((x'y)_{g_1\circ u}\cdot\mu_2\cdot(zs')_{g_1\circ u})\in[-K,K]$. Or puisque $poids(\mu_2)\in[-K,K]$, la propriété est bien réalisée.

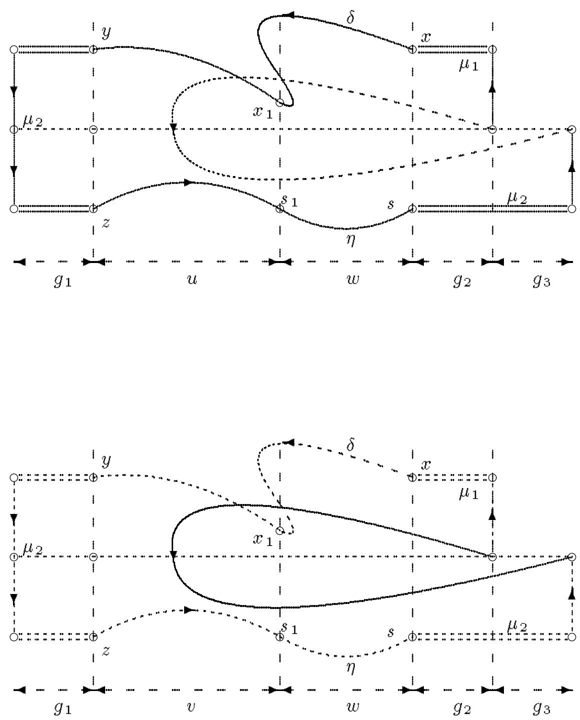


FIG. 6.22: $\sim_5$ *est une congruence droite*

□

Lemme 13 *La composée des relations* $\sim_2$ *et* $\sim_5$ *est une congruence droite.*

Preuve :

Soient u, v, w tels que $u \sim_2 v$ et $u \sim_4 v$. Soient $x, y, z, s \in V, g_1 \in \mathcal{P}(y,z), (g_2, g_3) \in \mathcal{Q}(s) \times \mathcal{R}(x) \cup \mathcal{R}(x) \times \mathcal{Q}(s)$ tels que $\mathcal{M}(x, y, z, s, g_1, g_2, g_3, u \circ w)$. (Voir figure 6.22) On reprend les notations de la définition de $\mathcal{M}$.

Soient $x_1, s_1 \in V$ et γ, δ tels qu'un représentant de $\mu_1 \cdot (x'y)_{g_1 \circ u \circ w} \cdot \mu_2 \cdot (zs')_{g_1 \circ u \circ w} \cdot \mu_3$ s'écrive $\delta \cdot (x_1'y)_{g_1 \circ u} \cdot \mu_2 \cdot (zs_1')_{g_1 \circ u} \cdot \eta$. Soit $g_2' \in \mathcal{R}(x_1), g_3' \in \mathcal{S}(s_1)$ dont les arcs significatifs respectifs sont comparables au dernier arc de δ et au premier de η qui passent par $\bigcup \tau^k$. Par minimalité du chemin choisi on a donc $\mathcal{M}(x_1, y, z, s_1, g_1, g_2', g_3', u)$.

On pose $(i', j') = (i - |u| + |v|, j - |u| + |v|)$. La propriété 34 assure qu'il existe $\delta' \lhd\!\rhd \delta$ et $\eta' \lhd\!\rhd \eta$ dans $g_1 \circ v \circ g_2 \circ g_3$ respectivement de source $\tau^{i'}$ et de destination $\tau_{j'}$.

Ces deux points permettent d'exhiber un chemin, lorsque l'on remplace u par v, qui assure que $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ w \circ g_2 \circ g_3} \trianglelefteq \delta' \cdot (x'_1 y)_{g_1 \circ v} \cdot \mu_2 \cdot (zs'_1)_{g_1 \circ v} \cdot \eta' \lhd\!\rhd (\tau^i \tau^j)_{g_1 \circ u \circ w \circ g_2 \circ g_3}$

Par symétrie, si l'on a $\mathcal{M}(x, y, z, s, g_1, g_2, g_3, v \circ w)$, on aura l'inégalité inverse, et plus particulièrement :

$$\begin{cases} poids((x'y)_{v \circ w}) + poids((zs')_{v \circ w}) = poids((x'y)_{u \circ w}) + poids((zs')_{u \circ w}) \\ min\_ope(ope((x'y)_{v \circ w}), ope((zs')_{v \circ w})) = min\_ope(ope((x'y)_{u \circ w}), ope((zs')_{u \circ w})) \end{cases}$$

Si ce n'est pas le cas, il y a deux cas de figure :

a) aucun arc significatif de g_2 n'intervient dans un chemin de $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ g_2 \circ g_3}$. Alors on a $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ g_2 \circ g_3} \lhd\!\rhd (\tau^{i'-|g_1|}\tau^{j'-|g_1|})_{v \circ g_2 \circ g_3}$. Mais alors on utilise que $u \sim_2 v$. Ainsi $(\tau^{i-|g_1|}\tau^{j-|g_1|})_{u \circ w \circ g_2 \circ g_3} \lhd\!\rhd (\tau^{i'}\tau^{j'})_{g_1 \circ v \circ g_2 \circ g_3}$. On a alors $(\tau^i \tau^j)_{g_1 \circ u \circ w \circ g_2 \circ g_3} \trianglelefteq (\tau^{i'}\tau^{j'})_{g_1 \circ v \circ g_2 \circ g_3}$ or $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ w \circ g_2 \circ g_3} \trianglelefteq \delta' \cdot (x'_1 y)_{g_1 \circ v} \cdot \mu_2 \cdot (zs'_1)_{g_1 \circ v} \cdot \eta' \lhd\!\rhd (\tau^i \tau^j)_{g_1 \circ u \circ w \circ g_2 \circ g_3}$. C'est donc que l'on avait aussi $\mathcal{M}(x, y, z, s, g_1, g_2, g_3, v \circ w)$.

b) si l'inégalité est stricte et que seul l'un des deux arcs significatifs de g_1, noté e, intervient dans un chemin de $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ g_2 \circ g_3}$. Alors on considère y_1, z_1 les projections des extrémités de e, et $g'_1 \in \mathcal{P}(y_1, z_1)$ dont le seul arc significatif est comparable à e. On a alors $\mathcal{M}(x, y_1, z_1, s, g'_1, g_2, g_3, v \circ w)$. On peut alors exhiber comme auparavant un chemin dans $g_1 \circ u \circ w \circ g_2 \circ g_3$ de τ^i à τ^j comparable à $(\tau^{i'}\tau^{j'})_{g_1 \circ v \circ w \circ g_2 \circ g_3}$ qui donne l'inégalité inverse puisque g'_1 est un sous graphe de g_1, ce qui montre que l'on a aussi $\mathcal{M}(x, y_1, z_1, s, g'_1, g_2, g_3, u \circ w)$.

□

En conclusion, si l'on regroupe ensemble les lemmes, 9, 10, 11, 12 et 13, on obtient :

Théorème 5 *La relation d'équivalence $\sim$ conjonction de $\sim_{0..5}$ est une congruence droite d'indice $O(K^{|C|^4 K^4})$.*

6.5 Commutation

Le but de la définition de $\sim$ est d'assurer la propriété de commutation du corollaire 31 du théorème 3. Il nous reste à vérifier que cette congruence vérifie bien les conditions requises. Cela se fait en quatre étapes, en montrant que les chemins $(xy), (xy'), (x'y)$ et $(x'y')$ sont comparables dans $u \circ v \circ w$ et $v \circ u \circ w$ dès lors que $u \sim v \sim w$.

Le lemme 9 assure déjà que $(xy)_{u \circ v \circ w} \lhd\!\rhd (xy)_{v \circ u \circ w}$.

Lemme 14 *Soient $u \sim v \sim w$, $\forall x, y \in V$ on a :*

$$(xy')_{u \circ v \circ w} \lhd\!\rhd (xy')_{v \circ u \circ w}$$

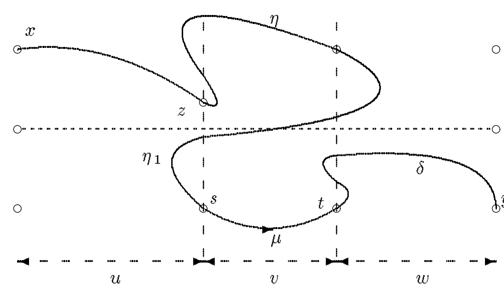


FIG. 6.23: $(xy')_{u\circ v\circ w}$ $\lhd\!\rhd$ $(xy')_{v\circ u\circ w}$

Preuve : Soit $\gamma \in (xy')_{u\circ v\circ w}$. Il existe une décomposition unique de γ en $(xz')_u \cdot \eta \cdot \eta_1 \cdot \mu \cdot \delta$ où la première et dernière arête de η ne sont pas dans u, η_1 est dans u, μ est le translaté de $|u|$ d'un chemin d'un ensemble $(st')_v$, et δ est un chemin sans arête de u. (Voir figure 6.23)

Si δ ne passe par aucun sommet de $\bigcup \tau^k$, alors on a $\neg Reset(y, w)$ et puisque $u \sim_0 v \sim_0 w$ on a $(xy')_{u\circ v\circ w}$ $\lhd\!\rhd$ $(xy)_{u\circ v\circ w}$. Mais on a déjà vu que $(xy)_{u\circ v\circ w}$ $\lhd\!\rhd$ $(xy)_{v\circ u\circ w}$. Alors, encore parce que $u \sim_0 v \sim_0 w$, on a $(xy)_{v\circ u\circ w}$ $\lhd\!\rhd$ $(xy')_{v\circ u\circ w}$ et donc $(xy')_{u\circ v\circ w}$ $\lhd\!\rhd$ $(xy')_{v\circ u\circ w}$.

On suppose à présent que δ contient un sommet de $\bigcup \tau^k$.

Puisque δ est sans arc de u, on peut faire valoir que $u \sim_2 v$ dans la propriété 34 pour exhiber δ' $\lhd\!\rhd$ δ dans $v \circ u \circ w$.

En considérant les premiers passages dans $\bigcup \tau^k$ de $\eta \cdot \eta_1 \cdot \mu \cdot \delta$ et de δ, on peut vérifier que u et v remplissent les conditions de préservation pour $\sim_3$. En particulier les chemins $(xz')_v$ et $(st')_u$ existent. Soit μ' le translaté de $|v|$ d'un représentant de $(st')_u$.

Puisque $\mu \cdot \delta$ contient des sommets de $\bigcup \tau^k$, et que η est non vide si η_1 est non vide, les propriétés 14 et 15 assurent que $poids(\eta_1) \in [-K, K]$. Soit η_1' son image dans v.

On cherche à présent un chemin η' $\lhd\!\rhd$ η dans $v \circ u \circ w$. Pour cela on utilise que $v \circ w \sim_1 u \circ w$ pour appliquer la propriété 33 sur η dans le graphe $u \circ u \circ w$, puis parce que $u \sim_2 v$ on applique la propriété 34 dans $v \circ (u \circ w)$. On obtient alors un chemin η' $\lhd\!\rhd$ η dans $v \circ u \circ w$.

Le chemin $(xz')_v \cdot \eta' \cdot \eta_1' \cdot \mu' \cdot \delta'$ est un chemin dans $v \circ u \circ w$ de mêmes extrémités que $(xz')_u \cdot \eta \cdot \eta_1 \cdot \mu \cdot \delta$. Montrons à présent qu'ils sont comparables.

On a déjà δ $\lhd\!\rhd$ δ', η $\lhd\!\rhd$ η', η_1 $\lhd\!\rhd$ η_1', et par $\sim_3$, $ope((xz')_u) = ope((xz')_v)$ et $ope((st')_u) = ope((st')_v)$.

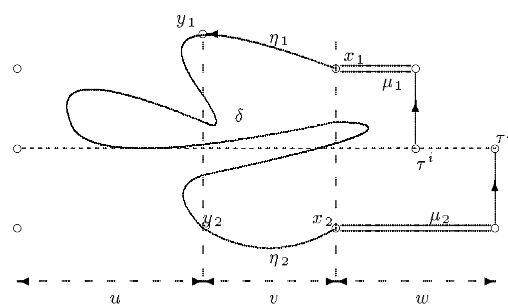


FIG. 6.24: *Utilisation de* $\sim_5$

Mais aussi :

$$\frac{\begin{array}{r} poids((xz')_v) - poids((\tau\tau')_v) = poids((xz')_u) - poids((\tau\tau')_u) \\ poids(\mu') - poids((\tau\tau')_u) = poids(\mu) - poids((\tau\tau')_v) \end{array}}{poids((xz')_v) + poids(\mu') = poids((xz')_u) + poids(\mu)}$$

Les deux chemins sont donc comparables.

Alors $(xy')_{v\circ u\circ w} \trianglelefteq (xy')_{u\circ v\circ w}$, et par symétrie $(xy')_{v\circ u\circ w} \lhd\!\rhd (xy')_{u\circ v\circ w}$

$\square$

Le résultat suivant et son corollaire seront utiles pour les préservations des contraintes $(x'y)$ et $(x'y')$ par commutation.

Propriété 54 *Soient u, v tels que $u \sim_1 v, u \sim_2 v$ et $u \sim_5 v$. Si γ est un chemin minimum de τ^i à τ^j dans $u \circ v \circ w$ pour $i, j \geq |u \circ v|$, qui se décompose en $\mu_1 \cdot \eta_1 \cdot \delta \cdot \eta_2 \cdot \mu_2$ (voir figure 6.24), avec μ_0 et μ_1 qui ne contiennent qu'un passage dans $\bigcup \tau^k$, η_1, η_2 sont les translatés de $|u|$ de chemins de la forme $(x_1'y_1)_v$ et respectivement $(y_2x_2')_v$, alors il existe $\gamma' \lhd\!\rhd \gamma$ dans $v\circ u\circ w$ de mêmes extrémités.*

Preuve : Puisque $\mu_1 \cdot \eta_1$ et $\eta_2 \cdot \mu_2$ passent par $\bigcup \tau^k$, il existe un élément g_1 de $\mathcal{P}(y_1, y_2)$ qui puisse remplacer u au sens de la propriété 39. Soit $g_2 \in \mathcal{R}(x_1)$ tel que $(\tau' x_1')_{g_2} \lhd\!\rhd \mu_1$, et $g_3 \in \mathcal{Q}(x_2)$ tel que $(x_2'\tau')_{g_2} \lhd\!\rhd \mu_2$. On a $\mathcal{M}(x_1, y_1, y_2, x_2, g_1, g_2, g_3, v) \vee \mathcal{M}(x_1, y_1, y_2, x_2, g_1, g_3, g_2, v)$. Puisque $u \sim_5 v$ c'est que $(x_1'y_1)_u$ et $(y_2x_2')_u$ existent.

Soient $y_1^v, y_2^v \in V_{|v|}$ de projection y_1 et y_2. Puisque $\mu_1 \cdot \eta_1$ et $\eta_2 \cdot \mu_2$ passent par $\bigcup \tau^k$, la propriété 34 s'applique. En appliquant la propriété 33 sur le chemin résultant, on montre qu'il existe $\delta' \lhd\!\rhd \delta$ dans $v \circ u \circ w$ entre y_1^v et y_2^v.

Si l'on pose η_1', η_2' comme étant les translatés de $|v|$ de représentants de $(x_1'y_1)_u$ et de $(y_2x_2')_u$, le chemin $\gamma' = \mu_1 \cdot \eta_1' \cdot \delta' \cdot \eta_2' \cdot \mu_2$ est un chemin de mêmes extrémités que γ dans $v \circ u \circ w$. On peut montrer qu'ils sont comparables.

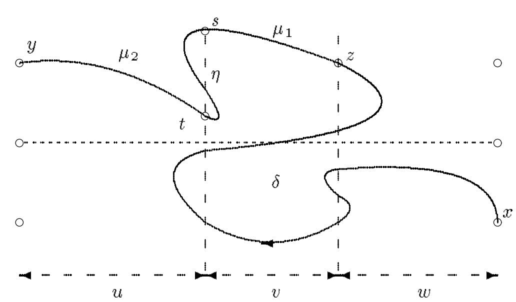


FIG. 6.25: $(x'y)_{u\circ v\circ w} \triangleleft\!\triangleright (x'y)_{v\circ u\circ w}$

On a déjà $\delta' \triangleleft\!\triangleright \delta$. Et puisque $u \sim_5 v$:

$$\begin{cases} min\_ope((x'_1y_1)_u, (y_2x'_2)_u) = min\_ope((x'_1y_1)_v, (y_2x'_2)_v) \\ poids((x'_1y_1)_u) + poids((y_2x'_2)u) = poids((x'_1y_1)_v) + poids((y_2x'_2)v) \end{cases}$$

On a bien $\gamma' \triangleleft\!\triangleright \gamma$. $\square$

Corollaire 55 *Soient u, v équivalents pour $\sim_1, \sim_2, \sim_5$. Pour tout $i, j \geq |u \circ v|$ on a $(\tau^i\tau^j)_{u\circ v\circ w} \triangleleft\!\triangleright (\tau^i\tau^j)_{v\circ u\circ w}$*

Preuve : En effet, soit $\gamma \in (\tau^i\tau^j)_{u\circ v\circ w}$. Il se décompose en $\gamma = \mu_0\Pi_k\delta_k \cdot \mu_k$ où les μ_k sont sans arête de $u \circ v$, et les δ_k sont dans $u \circ v$.

Les chemins μ_k sont aussi des chemins de $v \circ u \circ w$.

Puisqu'il existe un passage par $\bigcup \tau^l$ dans μ_k et μ_{k-1}, les propriétés 15 et 14 assurent que $poids(\delta_k) \in [-K, K]$. Ainsi, si δ_k est sans arête de u, on peut faire valoir que $u \sim_2 v$ pour exhiber un chemin comparable de mêmes extrémités dans $v \circ u \circ w$. Tandis que si δ_k contient des arêtes de u, on a le même résultat grâce à la propriété 54.

Ainsi on a $(\tau^i\tau^j)_{v\circ u\circ w} \trianglelefteq (\tau^i\tau^j)_{u\circ v\circ w}$, et par symétrie $(\tau^i\tau^j)_{v\circ u\circ w} \triangleleft\!\triangleright (\tau^i\tau^j)_{u\circ v\circ w}$ $\square$

Lemme 15 *Soient $u \sim v \sim w$, $\forall x, y \in V$ on a :*

$$(x'y)_{u\circ v\circ w} \triangleleft\!\triangleright (x'y)_{v\circ u\circ w}$$

Preuve :

Soit γ un représentant de $(x'y)_{u\circ v\circ w}$. Il se décompose (voir figure 6.25) en $\gamma = \delta \cdot \mu_1 \cdot \eta \cdot \mu_2$, où la dernière arête de δ n'est pas dans $u \circ v$, μ_1 est le translaté de $|u|$ d'un représentant d'un $(z's)_v$, η est dans $u \circ v$, et μ_2 est un représentant d'un $(t'y)_u$.

S'il n'y a pas de préfixe de δ, sans arête dans $u \circ v$, qui passe dans $\bigcup \tau^k$, c'est que $\neg Reset(x, w)$. Et puisque $u \sim_0 v \sim_0 w$, on a $(x'y)_{u \circ v \circ w} \lhd\rhd (xy)_{u \circ v \circ w} \lhd\rhd (xy)_{v \circ y \circ w} \lhd\rhd (x'y)_{v \circ u \circ w}$, et le résultat est acquis.

Sinon, on est dans le cas du corollaire 55. Il existe donc $\delta' \lhd\rhd \delta$ dans $v \circ u \circ w$ de mêmes extrémités.

Soient $s^v, t^v \in V_{|v|}$ de projection s et t. Puisque δ passe par $\bigcup \tau^k$, on peut appliquer les propriétés 33 et 34, pour exhiber $\eta' \lhd\rhd \eta$ dans $v \circ u \circ w$, d'extrémités s^v et t^v.

La destination de la dernière arête de δ qui passe par $\bigcup \tau^k$ est de projection z. Puisque w est un Tgraphe, c'est que $(tz)_w$ existe. Et puisque $v \sim w$, $(tz)_v$ existe aussi.

Soit $g_1 \in \mathcal{F}(t, z)$ correspondant à $(tz)_v$ au sens de la remarque 38. Par minimalité de γ on a $\mathcal{U}(t, y, z, g_1, u)$.

De même, si $\eta \neq \epsilon$ ou si μ_1 passe dans $\bigcup \tau^k$, $(zt)_v$ existe et donc $(zt)_w$. Sinon, on a $s = z = t$. Il existe donc $g_2 \in \mathcal{F}(z, t)$, (éventuellement $g_2 = g_\epsilon$) tel que $\mathcal{U}(z, s, t, g_2, v)$.

Les chemins $(t'y)_v$ et $(z's)_u$ existent et ont mêmes opérateurs que $(t'y)_u$ et $(z's)_v$. De plus puisque $poids((z's)_v) - poids((t'y)_v) = poids((z's)_u) - poids((t'y)_u)$, on a $poids((z's)_v) + poids((t'y)_u) = poids((z's)_u) + poids((t'y)_v)$.

Si $\mu_2' \in (t'y)_v$, et μ_1' est le translaté de $|v|$ d'un représentant de $(z's)_u$, on a alors $\delta' \cdot \mu_1' \cdot \eta' \cdot \mu_2' \lhd\rhd \gamma$, et donc $(x'y)_{v \circ u \circ w} \lhd (x'y)_{u \circ v \circ w}$. Alors par symétrie $(x'y)_{v \circ u \circ w} \lhd\rhd (x'y)_{u \circ v \circ w}$.

□

Lemme 16 *Soient $u \sim v \sim w, \forall x, y \in V$ on a :*

$$(x'y')_{u \circ v \circ w} \lhd\rhd (x'y')_{v \circ u \circ w}$$

Preuve : Soit γ un représentant de $(x'y')_{u \circ v \circ w}$.

Si γ est sans arête de $u \circ v$ alors γ est aussi un chemin de $v \circ u \circ w$, et $(x'y')_{v \circ u \circ w} \unlhd (x'y')_{u \circ v \circ w}$.

Sinon, (voir figure 6.26), γ se décompose en $\mu_1 \cdot \delta \cdot \mu_2$, tels que la première et dernière arêtes de δ sont dans $u \circ v$, tandis que μ_1 et μ_2 sont sans arête de $u \circ v$.

Si μ_1 ne passe par aucun sommet de $\bigcup \tau^k$, c'est que $\neg Reset(x, w)$ et puisque $u \sim_0 v \sim_0 w$, on a $(x'y')_{u \circ v \circ w} \lhd\rhd (xy')_{u \circ v \circ w}$ et par le lemme 14 $(xy')_{u \circ v \circ w} \lhd\rhd (xy')_{v \circ u \circ w}$ et encore parce que $u \sim_0 v \sim_0 w$ on a $(xy')_{v \circ u \circ w} \lhd\rhd (x'y')_{v \circ u \circ w}$.

Si μ_2 ne passe par aucun sommet de $\bigcup \tau^k$, c'est que $\neg Reset(y, w)$. On fait alors le même raisonnement en utilisant le lemme 15 pour montrer encore que $(x'y')_{u \circ v \circ w} \lhd\rhd (x'y')_{v \circ u \circ w}$.

Si μ_1 et μ_2 passent chacun par des sommets de $\bigcup \tau^k$, alors on est dans le cadre du corollaire 55. Il existe donc $\delta' \lhd\rhd \delta$ dans $v \circ u \circ w$ qui a les mêmes extrémités. On a alors $(x'y')_{v \circ u \circ w} \unlhd (x'y')_{u \circ v \circ w}$.

Par symétrie, on a donc $(x'y')_{u \circ v \circ w} \lhd\rhd (x'y')_{v \circ u \circ w}$

□

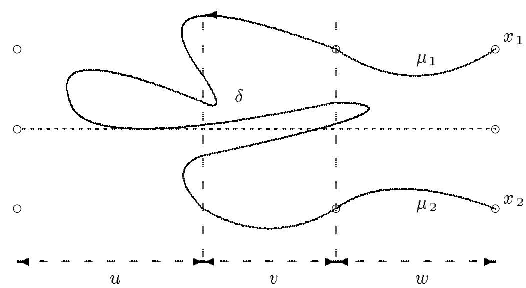


FIG. 6.26: $(x'y')_{u\circ v\circ w} \lhd\!\rhd (x'y')_{v\circ u\circ w}$

On peut à présent conclure cette section en regroupant les résultats des lemmes 9, 14, 15 et 16. La relation $\sim$ vérifie bien les conditions du théorème 3 puisque l'on a :

Théorème 6 *Soient $u \sim v \sim w$, on a $\Phi(u \circ v \circ w) \models\!\!| \Phi(v \circ u \circ w)$.*

Alors le corollaire 31 s'applique et on a :

Théorème 7 *Pour tout automate à compteurs codant un automate temporisé, il existe un automate plat effectivement constructible qui l'émule.*

Puisqu'alors le théorème 2 s'applique, et que l'on peut déduire la relation binaire d'accessibilité d'un automate temporisé à partir de la relation binaire d'accessibilité de l'automate à compteurs qui le code (Cf section 2.3.2) en préservant la longueur des exécutions, c'est que :

Théorème 8 *La relation binaire d'accessibilité pour les automates temporisés avec la longueur des exécutions comme paramètre est expressible dans $\mathcal{T}h$.*

Le corollaire 22 du chapitre 4 s'applique également pour les automates temporisés. On a donc également en conséquence :

Corollaire 56 *Soit w une séquence de transition d'un automate temporisé. Le problème de savoir si l'itération infinie ou non bornée de w existe est exprimable dans $\mathcal{T}h_{\mathbb{R}}$.*

Et puisque les automates plats émulent les automates temporisés on a aussi :

Corollaire 57 *Le problème de savoir si il existe une exécution de longueur infini entre deux états d'un automate temporisé est décidable.*

Chapitre 7

Logique pour la vérification

Les logiques dans lesquelles on exprime les propriétés des modèles pour le model-checking sont généralement des logiques temporelles (voir [Eme90]). Le model checking est alors décidable pour les automates d'états finis et ne l'est pas toujours pour des extensions quantitatives de la logique. Par exemple il l'est pour TCTL, et ne l'est pas pour TLTL [AH94]). Pourtant cet aspect est capital lorsque les modèles contiennent des variables. D'autre part les contraintes quantitatives exprimées par ces logiques sont souvent faibles, et peuvent se résumer a de simples comparaisons avec des valeurs fixés.

Dans ce chapitre nous introduisons la logique temporelle CTL^* et sa variante temporisé $PTCTL$ pour mieux comprendre le type de propriétés qu'elles peuvent exprimer. Puisque les variables de nos modèles sont a valeurs numérique, nous avons pris le parti de ne pas étendre les logiques temporelles existante mais d'exprimer les propriétés dans le cadre plus naturel d'une extension la théorie additive des réels que nous présentons. Nous donnons enfin des éléments de comparaison avec les logiques temporelles sans montrer formellement d'inclusion ou d'équivalence, mais nous conjecturons que les formules de $PTCTL$ peuvent se transcrire dans notre logique. Nous concluons ce chapitre par quelques exemples significatif des possibilités de notre model checking.

7.1 La logique temporelle CTL^*

La logique temporelle CTL^* permet d'exprimer des propriétés sur les exécutions d'un système en terme d'ordonnancement. Ses propriétés sont typiquement de la forme "le prochain état du système vérifie une propriété P", "il est toujours vrai que le système fini par vérifier une certaine propriété", "le système, ou une exécution de ce système, vérifie un nombre arbitraire de fois une propriété".

Cette logique est une généralisation des deux logiques temporelles CTL et $PLTL$ sur lesquelles les premiers résultat de model checking ont été obtenu.

7.1.1 Définition

Sa syntaxe est donnée par la grammaire suivante :

$$
\begin{array}{ll}
\phi : = S \mid P & \\
S : = & q_0 \mid q_1 \mid q_2 \ldots \\
 & \mid \neg S \mid S \vee S \mid S \wedge S \mid S \Rightarrow S \\
 & \mid E\ P \mid A\ P \\
P : = & S \\
 & \mid \neg P \mid P \vee P \mid P \wedge P \mid P \Rightarrow P \\
 & \mid \bigcirc P \mid G\ P \mid F\ P \mid P\ \mathcal{U}\ P
\end{array}
$$

Elle est donc munie des connecteurs booléens usuels, d'un ensemble $\{q_0, q_1, q_2 \ldots\}$ de variables propositionnelles, et de modalité, $\bigcirc, G, F, E, A$ et $\mathcal{U}$, utilisées pour décrire les comportements futurs du système.

Ces formules sont de deux sortes, celles engendrées par le non terminal S, dites formules d'états, et celles engendrées par le non terminal P, dites formules de chemins.

Les modèles de cette logiques sont des paires formées d'un automate et d'un état pour les formules d'états, et d'un automate et d'une exécution infini pour les formules de chemin.

Nous définissons a présent la sémantique de cette logique, par induction sur la formule. Ici on dénote par s les configurations et par σ les exécutions. En ce qui concerne les connecteurs logique, on présente que les cas de $\neg$ et de $\vee$:

$\mathcal{A}, s \models q$	ssi q est l'état de la configuration s
$\mathcal{A}, s \models \neg\phi$	ssi $\mathcal{A}, s \not\models \phi$
$\mathcal{A}, s \models \phi \vee \psi$	ssi $\mathcal{A}, s \models \phi$ ou $\mathcal{A}, s \models \psi$
$\mathcal{A}, s \models E\ \phi$	ssi il existe σ issue de s telle que $\mathcal{A}, \sigma \models \phi$
$\mathcal{A}, s \models A\ \phi$	ssi pour toute exécution σ issue de s : $\mathcal{A}, \sigma \models \phi$
$\mathcal{A}, \sigma \models \phi$ formule d'état	ssi $\mathcal{A}, s \models \phi$, où s est la configuration initiale de σ
$\mathcal{A}, \sigma \models \neg\phi$	ssi $\mathcal{A}, \sigma \not\models \phi$
$\mathcal{A}, \sigma \models \phi \vee \psi$	ssi $\mathcal{A}, \sigma \models \phi$ ou $\mathcal{A}, \sigma \models \psi$
$\mathcal{A}, \sigma \models G\ \phi$	ssi pour suffixe σ' de $\sigma, \mathcal{A}, \sigma' \models \phi$
$\mathcal{A}, \sigma \models F\ \phi$	ssi il existe un suffixe σ' de σ tel que $\mathcal{A}, \sigma' \models \phi$
$\mathcal{A}, \sigma \models \phi\ \mathcal{U}\ \psi$	ssi $\exists \alpha, \beta, \sigma = \alpha \cdot \beta$ et $\mathcal{A}, \beta \models \psi$
	et $\forall \alpha_1 \cdot \alpha_2 = \alpha, \mathcal{A}, \alpha_2 \cdot \beta \models \phi$
$\mathcal{A}, (s_1, s_2, \ldots) \models \bigcirc\ \phi$	ssi $\mathcal{A}, (s_2, \ldots) \models \phi$

Lorsque l'automate considéré est clairement identifié, on préferera ecrire $s \models \phi$ plutot que $\mathcal{A}, s \models \phi$.

On remarque qu íci on a identifié les nom des etats et ceux des variables propositionnelle. La présentation standart voudrait que chaque état soit associé à l'ensemble des variables propositionnelles vraies dans cet état et que $\mathcal{A}, s \models q$ ssi la variable q est vraie dans s. Notre choix ne diminue en rien l'expressivite des formules. On parlera en effet des etats du systeme en les nommant directement plutot que d'y faire reference implicitement : chaque variable propositionnelle renvoyant à l'ensemble des etats etiquetes par cette variable. En effet, considérons l'exemple $\bigcirc P$ où P est une variable propositionnelle, et pas un état. On pose Q_P l'ensemble des etats qui verifient la proposition P, il se represente par une disjonction entre ses états. La propriété équivalente est $\bigcirc Q_P$. C'est à dire que l'on transforme "l'état suivant vérifie P" en "l'état suivant est l'un de ceux qui verifie P".

Ce choix nous permettra de comparer plus facilement les logiques temporelles et celle que nous employons.

7.1.2 Expressivité

Nous regardons à présent brievement quelques exemples de formule de CTL^* et leur signification :

$\sigma \models G\ (p \vee q)$	l'execution σ ne parcours que les etats p et q
$s \models E\ F\ p$	de l'etat s il existe une execution qui mene en p
$s \models A\ G\ (p \vee q)$	de l'etat s toutes les executions ne passent que par les états p ou q
$\sigma \models F\ p$	l'execution σ fini par passer par l'etat p
$s \models (E\ F\ p) \wedge (E\ F\ q)$	de l'état s il existe une execution qui mene en p et une qui mene en q
$s \models E\ (F\ p \wedge F\ q)$	de s il existe une execution qui mene en p et en q

Mais aussi, l'un des intérêts de la logique temporelle est de pouvoir exprimer des propriétés qui sont vérifiées périodiquement. Ainsi la formule $AGF\ p$ est vérifiée en un état s si tous les chemins issus de cet état passent une infinité de fois par l'état p, la formule $AG(p \Rightarrow E\ q)$ exprime que chaque fois qu'une exécution passe par p elle repasse par q par la suite.

La logique que nous venons de présenter, et a fortiori ses fragments, ne permettent pas d'exprimer de propriétés quantitatives. Elle peuvent assurer qu'une propriété sera vérifie, ou périodiquement vérifiée, mais elle ne peuvent exprimer le fait qu'une requête sera toujours satisfaite dans un délai donné, ou encore moins calculer un ensemble de conditions sur ces délais pour qu'une requête soit satisfaite.

7.1.3 Extensions quantitatives

Il existe des extensions de ces logiques pour s'attaquer à ce problème. Elles explorent deux directions. Une première consiste à considérer que certaines actions, ou certain passages dans des états, correspondent a une action quantifiable, par exemple un incrément du temps lors du passage dans un état "tick". Il n'est pas nécessaire alors d'étendre les modèles pour exprimer des propriétés temporelles, et on enrichit la logique seulement. Cette approche conduit par exemple à la définition de RTCTL [EMSS90].

Lorsque l'on s'intéresse a des propriétés "temporelle", il est aussi naturel d'introduire des variables qui évoluent continuement au fil du temps. On choisit alors d'enrichir le modèle et de contraindre ses exécutions selon les valeurs de ces variables. C'est ce qui a été fait avec les automates temporises [AD94] ou les automates hybrides [ACH+95]. Dans ce cas on étant à la fois le modèle et la logique.

La logique couramment utilisée pour le modèle checking des automates temporises est TCTL. Nous présentons ici rapidement une de ses descendante, la logique PTCTL [Wan95]. Il s'agit d'une logique qui exprime des formules d'états, sa syntaxe est donnée par la grammaire :

$$\begin{array}{rl} \phi : = & false \mid true \mid q_0 \mid q_1 \mid \ldots \\ & \mid x \sim c \\ & \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \Rightarrow \phi \\ & \mid E\ \phi\ \mathcal{U}_{\sim\theta}\ \phi \mid A\ \phi\ \mathcal{U}_{\sim\theta}\ \phi \end{array}$$

où x est une des horloges du système, $\sim$ est un opérateur dans $\{<, >, \leq, \geq, =\}$, c est un entier dans $\mathbb{N}$, et θ est soit un entier dans $\mathbb{N}$ soit une variable libre de la formule appelée paramètre.

Cette logique permet d'utiliser comme atomes des contraintes sur les valeurs courantes des horloges ($x \sim c$), et des contraintes sur les délais nécessaires à la satisfaction des formules, c'est là le rôle des indices de la modalité $\mathcal{U}$. La sémantique informelle de $E\ \phi\ \mathcal{U}_{\sim\theta}\ \psi$ est : *"à partir de la configuration courante, il existe une exécution qui satisfait ϕ et qui plus tard satisfera ψ, dans un intervalle de temps $\sim \theta$"*.

Lorsque θ est un entier fixé par la formule on cherche a vérifier une contrainte de délai fixé, comme *" l'ascenseur arrive dans au plus 3 secondes"*. Lorsque la valeur θ n'est pas fixée, et c'est la l'originalité de PTCTL, l'ensembles de valeurs est un paramètre du modèle. Pour l'exemple on peut parler de *"l'ensemble des délais que peut mettre l'ascenseur pour arriver"*. Le model checking de $PTCTL$ est décidable, et l'ensemble des valeurs des paramètres est exprimable sous la forme d'une expression linéaire.

7.2 Notre logique

La logique avec laquelle nous construisons nos formules est une extension de $\mathcal{T}h$ qui intègre des modalités interprétées à l'aide des relations binaires d'accessibilité.

7.2.1 Définition

La syntaxe de cette logique est donnée par la grammaire :

$$
\begin{array}{lll}
\phi : = & & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \Rightarrow \phi \\
 & \mid \exists x, \phi \mid \forall x, \phi & \\
 & \mid Form & \\
Form : = & & Terme \leq Terme \mid Terme < Terme \\
 & \mid Entier(Terme) & \\
 & \mid (x_1, \ldots, x_p)\ \Box^{Q}_{qq'}(n)\ (y_1, \ldots, y_p) & \\
 & \mid (x_1, \ldots, x_p)\ \boxed{\omega}^{Q}_{q} & \\
Terme : = & & Terme + Terme \\
 & \mid \mathbb{R} \mid \mathcal{V} &
\end{array}
$$

Les variables sont prisent dans l'ensemble $\mathcal{V}$ et sont interprétés dans $\mathbb{R}$. La sémantique de $+, \leq, <$ est celle utilisée habituellement sur les réels. Le prédicat *Entier* caractérise tout les réels qui sont aussi des entiers. Informellement la modalité $\Box^{Q}_{qq'}(n)$ a pour objet d'assurer l'existence d'une exécution de n transitions entre les états q et q' du modèle, avec $x_1 \ldots x_p$ pour valeurs initiales des variables, et $y_1 \ldots y_p$ pour valeurs finales. Cette exécution ne doit traverser que des états précisés dans Q. Plus formellement la sémantique de cette modalité est reliée à la relation binaire d'accessibilité. Si les modèles sont des automates on note A_Q le sous automate de A restreint aux états de Q. On définit $A, v_1, \ldots v_p, n, v'_1, \ldots, v'_p \models (x_1, \ldots, x_p)\Box^{Q}_{qq'}(n)(y_1, \ldots, y_p)$ ssi l'automate A a p compteurs, et $(v_1, \ldots, v_p) \xrightarrow[A_Q, q, q']{n} (v'_1, \ldots, v'_p)$. La sémantique pour $\boxed{\omega}^{Q}_{q}$ est donnée par $(v_1, \ldots, v_p) \models (x_1, \ldots, x_p)\ \boxed{\omega}^{Q}_{q}$ ssi de la configuration $(q, v_1, \ldots, v_p)$ il existe une exécution infinie qui ne passe par aucun états de Q.

Lorsque les modèles sont des automates à compteurs plats, ou des automates temporisés, le théorème 1 et le corollaire 22, et 8, 57 assurent que le model-checking est décidable pour ces formules.

Nous donnons à présent un petit aperçu de la puissance de cette logique, en traduisant des formules de base de CTL^* dans notre logique.

7.2.2 Expressivité

Il est clair que selon la dichotomie formules d'états, formules de chemins, nos formules n'expriment que des formules d'états.

- Soit Q un ensemble d'états, et Q_A l'ensemble des états de l'automate A. La formule $A, (q, X) \models EF\ Q$ signifiant *"il existe un chemin issu de q qui fini par passer par des états de Q"* peut se reformuler en *"il existe un chemin issu de q qui passe par des états de Q"*. Ainsi elle est équivalente à $\exists X', \exists n, \bigvee_{q' \in Q} X \Box^{Q_A}_{qq'}(n) X'$.

- Pour une formule comme $E(F\ Q \wedge F\ Q')$ on décomposera le chemin en considérant soit un premier passage dans Q prolongé par un passage dans Q', soit inversement, un premier passage dans Q' prolongé par un passage dans Q.

Dans ces deux formules l'hypothèse sur les exécution infinies faites dans CTL^* n'est pas exprimée. Ces formules sont donc encore incorrectes mais on peut choisir, à la suite de ces formules, d 'imposer des conditions sur la configuration (q', X') pour que le chemin existant soit prolongeable en un chemin infini ou éventuellement de façon non bornée seulement. Pour exprimer que seulement cette dernière condition on ajouterait $\forall n', \exists X'', \bigvee_{q'' \in Q_A} X' \Box^{Q_A}_{q'q''}(n') X''$, et pour un prolongement infini, $\boxed{\omega}^{Q_A}_{q}$.

• Lorsque l'on considère des formules traduisant des invariants, comme $AG(\ q \Rightarrow F\ q')$, le caractère infini des exécutions est primordial. Cette formule est équivalente a $\neg\ EF(q \wedge G\ \ \neg q')$. Et on peut retranscrire $(q_0, X) \models AG(\ q \Rightarrow F\ q')$ par $(q_0, X) \models \neg(\exists Y, \exists n, X \Box^{Q_A}_{q_0 q}(n) Y \wedge Y \boxed{\omega}^{Q_A \setminus q'}_{q})$. Notre logique permet donc d'exprimer des formules de vivacité.

• Une autre formule simple a transcrire est $(q_0, V) \models EG\ Q$ équivalente à $V \models X \boxed{\omega}^{Q}_{q_0}$.

Notre logique permet donc d'exprimer des propriétés non triviales de la logique temporelle. De plus il est clair qu'en comparaison des logiques qualitatives existantes, notre logique a l'avantage de pouvoir exprimer une grande variété de contraintes sur ses variables libres qui sont par exemple des valeurs intermédiaires des compteurs ou des horloges. Pour ne citer qu'une différence avec TCTL, et PTCTL, on ne peut pas y exprimer qu'il existe une exécution bouclant sur la même configuration apres un délai fixé. En effet, ces logiques expriment des relations entre des régions, qui sont des approximations des configurations, et seules des propriétés entre régions peuvent y être vérifiées.

Notre logique peut en outre exprimer et permettre la vérification de propriétés particulièrement fines sur les valeurs des variables à l'issue d'une exécution. Par exemple une propriété comme *la valeur finale de l'horloge peut être égale au double de celle qu'elle aurait été si la valeur initiale avait été la somme de cette valeur et de la valeur initiale courante.*

Si les états de départ et d'arrivée sont q et q', et les variables initiales et finales respectivement x et y, la formule précédente se traduit par : $\exists n, x \Box^{Q_A}_{qq'}(n) y \wedge \exists y', \exists n', x + y \Box^{Q_A}_{qq'} y' \wedge y = 2y'$.

Elle peut de la même façon se référer au passé ou au futur.

Il est a remarquer qu'elle permet aussi d'exprimer le nombre de transitions des exécutions par des variables de première classe, permettant par exemple d'exprimer la propriété *"il existe une exécution de longueur double si je pars de q plutôt que de q' et qui soit de même durée"*. Ce qui n'est pas le cas d'aucune logique à notre connaissance, elle permet donc en un sens de combiner plusieurs notion de temps.

On peut lui ajouter l'avantage non négligeable de pouvoir caractériser les ensemble de valeurs pour que de telles exécutions soient possibles puisque le résultat du model checking est une formule de $\mathcal{T}_h$ sur les variables restées libres.

Bien sur on peut se demander si ce type de propriétés ont un sens sur les automates temporisés. Pour s'en convaincre, il faut imaginer que ce procédé peut

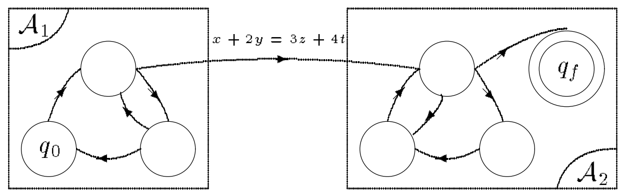


FIG. 7.1: *Un systeme plat par bloc*

être utilisé sur des systèmes plus complexe que les automates temporisés, et dans lesquels on peut isoler des composants modélisables par des automates temporisés et interagissant avec un environnement plus complexe. Prenons l'exemple de systèmes qui se décomposent de blocs d'automates temporisés reliés entre eux par des transitions effectuant une transformation du premier ordre sur les valeurs des variables. Si ce système par bloc est plat (voir figure 7.1) alors il est est analysable par une technique de model-checking.

Il semble donc que nos résultats sur l'expressibilité de la relation binaire permette de définir une nouvelle logique pour la vérification d'intersection non triviale avec les logiques temporelles, et une comparaison plus poussée reste indispensable. En effet, la syntaxe présentée ici pour notre logique est encore maladroite et n'utilise pas pleinement les possibilités offertes par l'expression de $lrstepnq, q', A$. Il nous semble plus précisément qu'une simple présentation plus habile des modalités $\Box$ se prêterait à un codage de toutes les formules de $TCTL$. Si cette conjecture s'avère exact, cela montrerait le caractère particulièrement expressif de cette logique.

Bien entendu la complexité de ce model checking ne serait en rien intéressante, mais un tel résultat pourrait toutefois permettre de proposer de nouvelles extensions de $TCTL$, dont le model checking est décidable, et dont la complexité exacte pourrait être plus raisonnable. Par exemple dans notre logique le nombre de transition est un paramètre de première classe. On pourrait donc naturellement tenter d'étendre $TCTL$ avec des modalités sur les nombres de transitions. Cette proposition serait un point de vue certainement peu complexe combinant les logiques $TCTL$ et $RTCTL$.

Chapitre 8

Conclusion

8.1 Résumé

Nous avons présenté ici quelques nouveaux résultats de décidabilité [CJ98, CJ99] qui portent sur l'étude de systèmes modélisés par des automates à variables numériques.

La possibilité d'exprimer dans une théorie décidable le point fixe d'une itération, et plus précisément la relation entre valeurs d'entrées, de sorties et le nombre d'itérations permet d'effectuer une vérification du modèle. Nous avons présenté deux calculs de cette relation : pour le modèle des automates à compteurs plats, et pour celui des automates temporisés.

L'hypothèse de platitude des automates, qui doivent avoir au plus une boucle par état, peut sembler très éloignée des besoins des systèmes réels et ne pas avoir d'exemples pratique d'applications. Nous montrons, à la suite de [FO97], qu'au contraire il existe des modèles très utilisés dont l'étude se ramène à celle d'automates plats. C'est le cas en particulier des automates temporisés, ce qui donne un nouveau point de vue, quelques peu surprenant, sur ces automates.

En outre ce résultat permet de décider de nouvelles propriété de systèmes temporisés, qui ne sont pas modélisable par des automates temporisés seuls. Ainsi des propriété d'accessibilité pour des systèmes d'automates temporisés reliés entre eux par des transition gardées par des formules de premier ordre restent décidables.

8.2 Travaux futurs

On peut bien entendu encore chercher à étendre ou à préciser ces résultats en étudiant plusieurs directions :

• La première consiste, à la base, à chercher à étendre le fragment des opérations autorisées pour les automates plats. Nous avons vu que le résultat initial sur les contraintes "gap-order" [Rev90] a été étendu d'au moins deux façons, soit en introduisant des contraintes de périodicité, soit en autorisant une addition faible, mais dans l'optique de ne pas faire d'hypothèses sur la structure du modèle. De même on pourrais essayer de rapprocher notre point de vue, où les transformations sont non déterministes, de celui de [Boi98], par exemple en imaginant des transformations de la forme $MX \leq B \Rightarrow \exists T, NT \leq C \wedge X' = AX + T$. L'étude de ce fragment semble toutefois particulièrement difficile car il n'est pas une simple conséquence du résultat sur les transformations linéaires de [Boi98] ou du notre. En effet, même si l'on se restreint à une matrice A qui soit une projection le point fixe n'est pas nécessairement exprimable dans la théorie additive.

Il reste également la possibilité de chercher à combiner des variables non numériques et numériques. L'étude des $CQDD$ [BH97] qui combinent déjà dans leur représentations symboliques des automates de mots et des contraintes sur les occurrences des transitions pourrait constituer un bon point de départ.

• Une seconde direction serait de regarder des modèles existant, en cherchant à les réduire en automates plats. On sait déjà par exemple qu'en plus des automates temporisés une sous classe de réseaux de Pétri [FO97] est déjà équivalente à des automates plats.

Symétriquement, on peut constater que le fragment des automates plats dans lequel s'expriment les automates temporisés n'est qu'un sous fragment de $\mathcal{F}_c$, et que l'on utilise donc pas toute l'expressivité des automates à compteurs plats. On pourrait donc chercher des règles de commutation sur d'autres fragments de $\mathcal{F}_c$ qui rendent l'hypothèse de platitude inutile. Cette approche pourrait conduire à la définition de nouveaux modèles, par exemple étendant les automates temporisés, qui soient encore émulés par des automates plats.

• Nous sommes également curieux des conséquences exacte de l'expression de la relation binaire d'accessibilité et du prédicat infiniment itérable en terme de model checking. Un travail plus immédiat serait donc de préciser ses interactions exactes avec les logiques temporelles classiques.

Bibliographie

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AH94] R. Alur and T. Henzinger. A really temporal logic. *ACM*, 41:182–204, 1994.

[BB96] Patrice Godefroid Bernard Boigelot. Symbolic verification of communication protocols with infinite state spaces using qdds. In *Proc. Computer Aided Verification*, Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, 1996.

[BB99] L. Fribourg B. Bérard. Reachability analysis of (timed) petri nets using real arithmetic. In *Proc. 10th Int. Conf. Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 178–193. Springer-Verlag, 1999.

[BBC+96a] B. Barras, S. Boutin, C. Cornes, J. Courant, J.-C. Filliâtre, H. Herbelin, G. Huet, P. Manoury, C. Muñoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saïbi, and B. Werner. *The Coq Proof Assistant Reference Manual Version 6.1.* INRIA-Rocquencourt-CNRS-ENS Lyon, December 1996.

[BBC+96b] N. Bjørner, A. Browne, E. Chang, M. Colón, A. Kapur, Z. Manna, H. B. Sipma, and T. E. Uribe. STeP: deductive-algorithmic verification of reactive and real-time systems. In *Proc. 8th Int. Conf. Computer Aided Verification (CAV'96), New Brunswick, NJ, USA, July-Aug. 1996*, volume 1102 of *Lecture Notes in Computer Science*, pages 415–418. Springer-Verlag, 1996.

[BGP97] T. Bultan, R. Gerber, , and W. Pugh. Symbolic model checking of infinite state systems using presburger arithmetic. In O. Grumberg, editor, *Proc. Computer Aided Verification*, volume 1254 of *LNCS*, Haifa, Israel, 1997. Springer-Verlag.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-chanels systems with nonregular sets of configurations. In *Procedings of the 24th ICALP*, LNCS 1256, July 1997.

[Boi98] Bernard Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD thesis, Universite de Liege, 1997-1998.

[Bry92] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagram. *ACM Computing survey*, 1992.

[BW94a] Bernard Boigelot and Pierre Wolper. Symbolic verification with periodic sets. In *Computer Aided Verification, Proc. 6th Int. Conference*, LNCS, Stanford, June 1994. Springer-Verlag.

[BW94b] Bernard Boigelot and Pierre Wolper. Symbolic verification with periodic sets. In *Computer Aided Verification, Proc. 6th Int. Conference*, LNCS, Stanford, June 1994. Springer-Verlag.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent system using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244-263 April 1986.

[CH78] P. Cousot and N. Halbwachs. Automatic dicovery of linear restraints among variables of a program. In *Proceedings of the Fifth Annual Symposium on Principles of Programming Languages (POPL'78)*, pages 84–96. ACM Press, 1978.

[CJ98] Hubert Comon and Yan Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In A. Hu and M. Vardi, editors, *Proc. Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer-Verlag, 1998.

[CJ99] Hubert Comon and Yan Jurski. Timed automata and the theory of real numbers. In *Proc. Conf. on Concurrency theory*, Eindhoven, 1999. To appear. The full version can be found at http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rr-lsv-1999-6.rr.ps.

[CLR90] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to algorithms*. MIT Press, 1990.

[DCR94] D.Toman, J. Chomicki, and D.S. Rogers. Datalog with integer periodicity constraints. In *Int. Symp. on Logic Programming*, 1994.

[DP99] Giorgio Delzanno and Andreas Podelski. Model checking in clp. In Rance Cleaveland, editor, *Proceedings of TACAS'99, the Second International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer-Verlag, March 1999.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[EMSS90] E.A Emerson, A.K Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *Procedings of the 2nd Int. conf. Computer-Aided Verification (CAV'90)*, volume 531 of *LNCS*, pages 136–145. Springer-Verlag, June 1990.

[FMR68] P.C. Fisher, A.R. Meyer, and A.L. Rosenberg. Counter machines and counter languages. *Math. Systems Theory, 2, pp. 265-283*, 1968.

[FO97] Laurent Fribourg and Hans Olsen. A decompositional approach for computing least fixed-point of datalog programs with z-counters. *J. Constraints*, 1997.

[Fri98] Laurent Fribourg. A closed-form evaluation for extended timed automata. Technical report, LSV-98-2, Lab. Specification and Verification, ENS de Cachan, Cachan, France, March 1998.

[HHW97] T. A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. HyTech: A model-checker for hybrid systems. In *Proc. 9th Int. Conf. Computer Aided Verification (CAV'97), Haifa, Israel, June 1997*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–463. Springer-Verlag, 1997.

[KPRS94] Waine Kelly, William Pugh, Evan Rosser, and Tatiana Shpeisman. Cs-tr-3457, transitive closure of infinite graphs and its applications. Technical report, Univ. of Maryland CS technical reports, 1994.

[Min61] M.L. Minsky. Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines. *Annals of Mathematics*, 74, 1961.

[NI97] M. Vardi N. Immerman. Model checking and transitive closure logic. In *Proc. Computer Aided Verification*, Lecture Notes in Computer Science, pages 291–302. Springer-Verlag, 1997.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. Pvs: A prototype verification system. In *11th International Conference on Automated Deduction (CADE)*, volume Volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, 1992.

[PA98] Bengt Jonsson Parosh Abdulla, Ahmed Bouajjani. On-the-fly analysis of systems with unbounded, lossy fifo channels. In A. Hu and M. Vardi, editors, *Proc. Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 1998.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Foundation of Computer Science*, pages 46–57, Oct-Nov 1977.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurent systems in cesar. In *Proc. Int. Symp. on Programming*, volume 137 LNCS, pages 337–351. Springer-Verlag, april 1982.

[Rev90] P. Revesz. A closed form for datalog querise with integer order. In *Proc.3rd International Conference on Database Theory*, pages 187–201, 1990.

[Rev98] Peter Z. Revesz. Safe datalog queries with linear constraints. In Michael J. Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming - CP98, 4th International Conference*, volume 1520 of *Lecture Notes in Computer Science*, pages 355–369. Springer-Verlag, October 1998.

[Ric53] H.G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the AMS, 89, 25-59*, 1953.

[Sch99] Phillipe Schnoebelen. *Ouvrage collectif, Verification de Logiciel, techniques et outils du model-checking*. Vuibert, 1999.

[SFRC99] G. Sutre, A. Finkel, O. Roux, and F. Cassez. Effective recognizability and model checking of reactive fiffo automata. In *Proc. 7th Int. Conf. Algebraic Methodology and Software Technology (AMAST'98)*, volume volume 1548 of *Lecture Notes in Computer Science*, pages pages 106–123. Springer-Verlag, Jan 1999.

[Wan95] F. Wang. Timing behaviour analysis for real time systems. In *Tenth Annual IEEE Symposium on Logic in Computer Science*, San Diego, CA, June 1995. IEEE Comp. Soc. Press.

[Yov97] S. Yovine. Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1/2):123–133, 1997.