THÈSE

présentée par

François Laroussinie

pour obtenir le titre de Docteur
de l'Institut National Polytechnique de Grenoble

(Arrêté ministériel du 23 novembre 1988)

Spécialité : Informatique

Logique temporelle avec passé pour la spécification et la vérification des systèmes réactifs

Thèse soutenue le 15 novembre 1994 devant le jury :

Ph. Jorrand	président
P. Enjalbert C. Jard	rapporteurs
P. Schnoebelen J. Sifakis	examinateurs

Thèse préparée au sein du Lab. d'Informatique Fondamentale et d'Intelligence Artificielle.

A Nanouche,

Remerciements

Je tiens tout d'abord à remercier les membres du jury:

- Philippe Jorrand qui a accepté de présider le jury.
- Patrice Enjalbert et Claude Jard qui ont accepté d'être rapporteurs. Je les remercie pour tous les contacts que nous avons eus et pour tous leurs conseils concernant un manuscrit rédigé trop rapidement !
- Joseph Sifakis qui a accepté de participer au jury.

Je tiens à remercier tout particulièrement Philippe Schnoebelen. Je le remercie d'avoir encadré cette thèse, de m'avoir "appris la recherche", de m'avoir guidé avec tant de disponibilité. Je remercie donc mon chef ! Mais je remercie aussi un ami avec qui j'aime discuter, à qui j'ai si souvent "piqué" le Monde ou le Canard, qui m'a prêté tant de livres, qui m'a appris le gambit Benko... Merci pour toutes ces années, et pour toutes celles qui vont suivre car ce n'est qu'un début (et notre ascenseur, on le publie où ?)...

Je remercie les membres et les anciens de l'équipe Parallélisme du LIFIA: Cyril, Juan, Wilfried, Cherif, Ferroudja, Zineb... Et bien-sûr particulièrement Sophie Pinchinat à qui je dois beaucoup...

Je remercie Philippe Jorrand de m'avoir accueilli au sein du LIFIA où j'ai passé d'excellentes années: j'ai beaucoup apprécié l'ambiance qui règne dans ce laboratoire et j'en garderai un très bon souvenir.

Je tiens à remercier Ricardo Caferra pour ses conseils, nombreux et utiles, depuis notre rencontre à l'Ensimag. Je remercie aussi Thierry Fraichard, Denis Lugiez et Cécile Tron pour ces années passées en leur compagnie.

Je remercie aussi l'équipe "Theory of Computation" de l'Université d'Ålborg où j'ai fait les dernières mises au point du manuscrit. Je remercie particulièrement Luca Aceto, Anna Ingólfsdóttir, Kåre Jelling Kristoffersen, Kim Guldstrand Larsen et... little Roberto pour leur accueil chaleureux.

Je salue aussi mes nombeux amis. Je ne peux en citer ici que quelques uns. Les grenoblois: Denis, Lydie, Mazdak, Jean-Jacques... Et les parisiens: René-Marc, Christine, Catherine, Patrick, Eric, Pascale, Bruno, Michèle et Christian P. Enfin, je remercie ma famille, et tout particulièrement ma mère qui m'a tant aidé depuis... toujours ! Je remercie Marie de m'avoir supporté (à tous les sens du terme !) toutes ces années... Et bien sûr Paco...

Sommaire

Chapitre 1

Introduction

1.1 Approche formelle pour le développement de programmes

Tandis que les systèmes informatiques se sont très largement développés, leur taille s'est accrue, leur structure et leur fonctionnement sont de plus en plus complexes. Maîtriser cette complexité est une nécessité quand il s'agit de systèmes dont les défaillances peuvent mettre en jeu l'environnement ou des vies humaines comme c'est le cas pour le contrôle automatique des grandes installations technologiques, l'aide au pilotage d'avions ou l'assistance médicale [LS92].

La conception de systèmes complexes et la garantie de leur fiabilité incitent au développement d'outils formels pour s'assurer qu'un programme donné se comporte bien comme on le souhaite. Pour ce faire, on représente les programmes (et leur comportement) par des objets mathématiques, il devient alors possible d'énoncer formellement ce qu'est la "correction" d'un programme et de démontrer en un sens mathématique qu'un programme donné est "correct".

La spécification d'un système informatique S est alors la description formelle du comportement souhaité pour S. Il est intéressant que cette description se fasse à un certain niveau d'abstraction, sans faire référence aux détails d'implémentation en permettant ainsi plusieurs choix d'implémentation possibles.

1.2 Les systèmes réactifs

Dans cette thèse, nous nous intéressons à une classe particulière de systèmes informatiques, *les systèmes réactifs* [Pnu86b] qui regroupent ceux dont l'objectif est plutôt le maintien d'une interaction avec un environnement que le calcul d'un résultat à partir de données en entrée. C'est le cas, par exemple, des systèmes de contrôle en automatique, des systèmes d'exploitation ou des protocoles de communication. En fait, par le terme "système réactif", on désigne à la fois des systèmes particuliers et un certain point de vue qu'on a sur eux à un moment donné de leur conception.

Pour énoncer la correction de ces systèmes, les formalismes comme la logique de Hoare dont les énoncés se présentent sous la forme d'une précondition et d'une postcondition ne peuvent être utilisés [1]: en effet, d'une part ces systèmes peuvent ne pas terminer (tout en restant corrects), et d'autre part il est nécessaire d'exprimer des propriétés portant sur le déroulement même de leur exécution. Par exemple, on peut souhaiter énoncer des propriétés sur l'ordonnancement des actions effectuées par un programme lors de son exécution.

1.3 La logique temporelle

Utilisée depuis longtemps par les linguistes [MT27], la logique temporelle n'a été utilisée comme langage de spécification en informatique que depuis une vingtaine d'années [Pnu77]. Elle permet d'exprimer des propriétés le long des exécutions grâce à des assertions telles que "P sera vraie un jour" ou "P sera vraie tant que Q est faux". La logique temporelle s'est révélée être un bon formalisme pour la spécification des systèmes réactifs [AEF89, Eme90, MP92].

En pratique, on associe à chaque système réactif, décrit par exemple dans un langage de programmation, un modèle sémantique qui représentera son comportement et c'est sur ce modèle que l'on établira une spécification formelle.

Les modèles que nous considérons dans cette étude sont des graphes étiquetés (structures de Kripke) où chaque nœud du graphe représente un état du système et est étiqueté par un ensemble de propositions atomiques (correspondant, par exemple, aux valeurs des différentes variables du système) et où chaque arête du graphe représente un pas d'évolution du système. Un chemin dans ce graphe correspond alors à une exécution du système étudié.

La logique temporelle propositionnelle permet de spécifier à un assez bon niveau de détail le comportement des systèmes, de raisonner sur les spécifications (la sa-

[1] De plus, même si on ne tient pas compte des limitations en terme d'expressivité, la logique de Hoare s'adapte très difficilement aux programmes parallèles.

tisfaisabilité est presque toujours décidable), de valider automatiquement les programmes à nombre fini d'états ou de synthétiser des schémas de programmes. Nous allons rappeler rapidement chacun de ces aspects qui expliquent pourquoi la logique temporelle est particulièrement bien adaptée à la spécification des systèmes réactifs.

1.4 Preuves, vérification, synthèse de programmes

Une fois choisie une logique temporelle comme formalisme de spécification, il est utile de disposer d'un système de preuve permettant de prouver formellement qu'un système réactif donné satisfait une spécification donnée. Un tel système de preuve contient des axiomes et des règles d'inférence permettant de raisonner sur les formules de la logique et les données manipulées par le programme et la spécification, ainsi que des règles de déduction spécifiques au système réactif étudié [MP83]. Une fois défini un tel système de preuve, on essaie de prouver que le système réactif satisfait sa spécification en utilisant les différents axiomes et règles d'inférence. Un inconvénient de cette approche est que la construction de telles preuves est souvent ardue et fastidieuse, requiert une grande compétence et leur automatisation est donc difficile.

Dans certains cas, le problème de la vérification, le *model-checking*, (étant donnés un système S et une spécification P, vérifier si S satisfait P) est décidable et il est alors possible de définir des méthodes de vérification automatique [CE81, QS82]. C'est notamment le cas des systèmes à nombre fini d'états. Depuis la définition des premiers algorithmes de model-checking, des progrès considérables ont été réalisés: l'utilisation des méthodes symboliques et l'emploi de structures de données comme les BDD (*binary decision diagram)* [Bry86] ont permis de valider des circuits VLSI de plus de 10^{100} états [BCM+90, BCMD90, CGL92].

Pour certaines logiques, il existe des procédures de décision pour la satisfaisabilité, c'est-à-dire des algorithmes permettant de décider si, étant donnée une formule f, il existe un modèle vérifiant f. Ce type d'outils est très utile pour raisonner formellement sur les spécifications. Cela permet, par exemple, de vérifier automatiquement si une spécification en entraîne une autre.

D'autre part, la logique temporelle est aussi utilisée dans les travaux sur la synthèse de programmes [EC82]: il s'agit alors de construire un schéma de programme à partir d'une spécification. Pour cela, on utilise des procédures qui, étant donnée une formule satisfaisable f, permettent de construire un modèle vérifiant f. On peut ainsi associer un modèle à une spécification: le modèle ainsi obtenu est alors correct par construction. Un inconvénient de cette approche réside dans la complexité exponentielle des procédures utilisées pour la construction des modèles pour de nombreuses logiques considérées dans la littérature (comme CTL ou PTL) [SC85].

1.5 Notre étude

Si la logique temporelle est un bon formalisme de spécification, son expressivité a malgré tout des limites tant théoriques que pratiques: certaines propriétés ne sont pas exprimables au moyen des logiques temporelles présentées dans la littérature, et d'autres le sont mais difficilement. Pour pallier ces limitations, on pourrait essayer d'étendre le formalisme, à l'aide par exemple de nouveaux opérateurs, pour augmenter l'expressivité. Néanmoins, nous avons vu qu'une série d'outils étaient définis à partir du langage de spécification (axiomatisations, systèmes de preuves, model-checking,...) et sa modification implique de réadapter l'ensemble de ces outils. De plus, l'extension de l'expressivité du formalisme de spécification peut se faire au détriment de l'efficacité de ces outils. Par exemple, le problème de la vérification automatique admet pour certaines logiques des algorithmes très efficaces (comme CTL pour laquelle il existe un algorithme dont la complexité dépend linéairement de la taille du modèle et de la taille des formules) mais est PSPACE-complet pour d'autres logiques (comme CTL^* ou PTL [SC85]).

Dans notre étude, nous n'avons pas cherché à étendre l'expressivité théorique des logiques mais plutôt leur expressivité pratique en définissant des logiques munies d'opérateurs facilitant l'expression des propriétés, qui peuvent se traduire dans les logiques classiques pour lesquelles existent déjà des outils de model-checking performants.

Parmi les opérateurs de la logique temporelle, on distingue les opérateurs du futur permettant de faire référence aux instants futurs du système et ceux du passé permettant de faire référence aux instants antérieurs. Les opérateurs du passé facilitent l'expression de nombreuses propriétés [LPZ85, KVd83] en permettant des assertions comme "P a été vraie dans le passé", ou "P est vraie depuis que Q a été vraie". L'usage de ces opérateurs rend les spécifications plus naturelles et plus concises. Mais dans la littérature, les logiques pour lesquelles existent des algorithmes de model-checking efficaces, n'autorisent pas ce type d'opérateurs. Dans cette thèse nous avons cherché à concilier la facilité d'expression obtenue avec les opérateurs du passé et l'efficacité de la vérification. Pour cela, nous avons défini plusieurs traductions entre logiques avec passé et logique "pur-futur": ce type de traductions, facilement implémentables, permet d'une part de spécifier avec un langage enrichi et d'autre part, d'utiliser les outils de model-checking habituels (et efficaces) après traduction.

Le chapitre 2 présente la logique $PCTL^*$ qui correspond à la logique CTL^* étendue avec des opérateurs du passé: chaque logique considérée dans cette étude peut être vue comme un fragment de $PCTL^*$. Ce chapitre contient essentiellement des rappels sur la logique temporelle (opérateurs, équivalences,...).

Dans le chapitre 3, nous présentons tout d'abord un exemple de spécification d'un ascenseur avec des opérateurs du passé. Ensuite nous essayons de construire une spécification du même ascenceur avec la logique CTL (pur-futur). Cet exemple souligne nettement les facilités d'expression que permettent les opérateurs du passé.

Le chapitre 4 présente un ensemble de résultats concernant l'expressivité comparée des fragments de $PCTL^*$. En particulier, nous présentons certaines logiques avec passé et nous montrons comment les traduire dans les logiques CTL ou $ECTL^+$. D'autre part, nous montrons l'impossibilité de telles traductions pour d'autres logiques.

Le chapitre 5 porte sur l'étude d'un nouvel opérateur (Now) qui se révèle être très utile pour la spécification lorsqu'on dispose d'opérateurs du passé. Nous montrons aussi comment ce nouvel opérateur peut être ajouté aux logiques considérées au chapitre 4 sans compromettre les possibilités de traduction.

Le chapitre 6 contient des résultats concernant l'expressivité comparée d'un ensemble particulier de fragments de $PCTL^*$ (sans opérateur *Next* et *Previous*). Nous montrons alors comment les techniques élaborées aux chapitres 4 et 5 peuvent s'adapter à ces logiques.

Chapitre 2

La logique $PCTL^*$

2.1 Introduction

Au chapitre précédent, nous avons expliqué en quoi la spécification des systèmes réactifs nécessitait d'autres formalismes que ceux classiquement utilisés (comme la logique de Hoare) dans le domaine des systèmes simplement transformationnels.

Ainsi Pnueli a-t-il proposé l'utilisation de la logique temporelle comme langage de spécification pour les systèmes réactifs [Pnu77]. Depuis, de nombreuses études ont montré la pertinence de cette approche [Lam80, Pnu86a, vB89, Eme90, MP92].

Avant d'être utilisée en informatique, la logique temporelle fut notamment utilisée par les linguistes [MT27]. Mais en fait, la logique temporelle, et plus généralement la logique modale, ont été étudiées dès la Grèce antique où se sont opposées différentes visions de la logique modale. Au Moyen Âge, elle fut aussi utilisée dans des argumentations théologiques. Nous renvoyons à [Sma64] pour une vue historique des problèmes concernant la formalisation des propriétés temporelles.

A l'origine, l'idée de la logique modale repose sur la volonté de représenter plusieurs formes de vérité, en particulier la *nécessité* ou la *possibilité*. La logique temporelle prolonge cette idée en faisant intervenir un contexte temporel: par exemple, ce qui est *nécessaire* sera *toujours* (notion temporelle) vrai.

En informatique, l'intérêt de la logique temporelle pour la spécification réside dans la possibilité d'exprimer des assertions telles que "P sera vraie (un jour)" ou "P sera vraie tant que Q est faux"... Ainsi peut-on énoncer des propriétés portant sur les exécutions du système (et pas seulement sur son état initial et sur son état final). En plus des spécifications assez naturelles, la logique temporelle permet de construire des algorithmes de vérification, dits de *model-checking* dont l'objectif est de décider si un système donné satisfait une spécification donnée. De plus, on peut, pour de nombreuses logiques temporelles, disposer d'axiomatisations et d'algorithmes permettant de vérifier si une spécification formelle est satisfaisable.

De nombreuses logiques temporelles ont été proposées dans la littérature. Elles diffèrent tant par la structure du temps considérée (linéaire ou arborescente, discrète ou continue,...) que par les modalités temporelles utilisées. Emerson [Eme90] donne un aperçu de la diversité des logiques temporelles considérées dans la littérature informatique. Dans cette étude, nous considérons des logiques temporelles propositionnelles interprétées sur des structures de Kripke (graphes étiquetés).

La diversité des logiques temporelles ainsi que l'existence d'autres formalismes (automates [Tho90], logiques dynamiques [KT90]) a conduit à définir des critères de comparaison entre tous ces formalismes. Ainsi les liens entre certaines logiques temporelles et la logique du premier ordre [Kam68, GPSS80, Sch90a] ou la logique monadique du second ordre [HT87] ont-ils été étudiés. D'autre part, plusieurs critères de comparaison entre logique temporelle ont été définis: on peut s'intéresser au *pouvoir de distinction* d'une logique (i.e. la capacité d'une logique temporelle à distinguer deux modèles) ou à son *pouvoir expressif*, ou *expressivité* (i.e. sa capacité à exprimer des propriétés). Outre ces deux critères, on peut aussi comparer les logiques temporelles selon la facilité (la concision etc ...) avec laquelle elles permettent d'énoncer des propriétés.

Dans cette étude, nous allons nous intéresser au problème d'expressivité comparée entre logiques: *étant donnée une formule f de la logique L_1, existe-t-il une formule de la logique L_2 équivalente à f ?*. L'enjeu de ce type de question est d'établir une hiérarchie entre les différentes logiques selon leur pouvoir expressif et d'aider ainsi à déterminer quelle est la logique la plus intéressante pour spécifier un système réactif.

Ce chapitre contient surtout des rappels qui servent à fixer les définitions et notations pour le reste de la thèse. D'une part, nous allons définir formellement les différentes logiques que nous allons étudier dans les chapitres suivants. Nous allons tout d'abord définir la syntaxe et la sémantique de la logique $PCTL^*$ qui correspond à la logique CTL^* augmentée des opérateurs du passé. Toutes les logiques que nous considérerons par la suite seront décrites comme des fragments de $PCTL^*$ et se situeront donc dans un cadre unifié (pour la syntaxe, la sémantique, ...). Nous rappelerons plusieurs résultats concernant les opérateurs temporels de $PCTL^*$. Ensuite nous présenterons plusieurs logiques temporelles souvent considérées dans la littérature (CTL^*, CTL, PTL, ...) et enfin nous rappelerons les résultats de comparaison d'expressivité pour ces logiques, qui permettent de donner une hiérarchie des logiques en fonction de leur pouvoir expressif.

Nous n'aborderons pas dans cette étude les problèmes d'axiomatisation des logiques temporelles. Pour le *model-checking*, nous rappelerons les résultats concernant la complexité de ce problème pour les différentes logiques considérées mais nous ne présenterons pas les algorithmes associés.

2.2 La logique $PCTL^*$

Nous allons tout d'abord définir la syntaxe et la sémantique de la logique $PCTL^*$ qui correspond à la logique CTL^* [EH86] à laquelle on a ajouté des opérateurs du passé (notre définition diffère de celle utilisée dans [HT87], cela fera l'objet de plusieurs remarques dans ce chapitre ainsi que dans le chapitre 5).

2.2.1 Syntaxe de $PCTL^*$

On suppose fixé un ensemble fini de propositions atomiques que l'on note $Prop$.

Définition 2.2.1 *(Syntaxe de PCTL*)*

Les formules de $PCTL^*$ sont données par la grammaire suivante:

$$PCTL^* \ni f,g ::= a \mid f \wedge g \mid \neg f \mid \mathsf{E} f \mid f \,\mathsf{U}\, g \mid \mathsf{X} f \mid f \,\mathsf{S}\, g \mid \mathsf{X}^{-1} f$$

où $a \in Prop$.

U est l'opérateur "*Until*" et X l'opérateur "*Next*" ou "*Next-Time*". S est l'opérateur "*Since*" parfois noté U^{-1}, et X^{-1} l'opérateur "*Previous*". Ces quatres opérateurs sont des opérateurs temporels linéaires (i.e. exprimant des propriétés le long d'une exécution). On distingue les opérateurs du futur (U et X) et ceux du passé (S et X^{-1}). E est un quantificateur (sur les futurs possibles).

Nous définissons les abréviations standards à partir des opérateurs précédents: $\top$, $\bot$, $f \vee g$, $f \Rightarrow g$, ... Ainsi que:

$$\begin{array}{lll} \mathsf{F} f \overset{\text{def}}{\equiv} \top \,\mathsf{U}\, f & \mathsf{F}^{-1} f \overset{\text{def}}{\equiv} \top \,\mathsf{S}\, f & \overset{\infty}{\mathsf{F}} f \overset{\text{def}}{\equiv} \mathsf{G}\mathsf{F} f \\ \mathsf{G} f \overset{\text{def}}{\equiv} \neg \mathsf{F} \neg f & \mathsf{G}^{-1} f \overset{\text{def}}{\equiv} \neg \mathsf{F}^{-1} \neg f & \overset{\infty}{\mathsf{G}} f \overset{\text{def}}{\equiv} \mathsf{F}\mathsf{G} f \end{array}$$

On définit aussi le dual de l'opérateur E : $\mathsf{A} f \overset{\text{def}}{\equiv} \neg \mathsf{E} \neg f$.

X, F, G, X^{-1}, F^{-1} et G^{-1} correspondent respectivement aux notations $\bigcirc$, $\Diamond$, $\Box$, $\bullet$, $\blacklozenge$ et $\blacksquare$ souvent utilisés dans la littérature. Nous allons maintenant définir la sémantique de ces opérateurs, d'abord formellement, nous en donnerons ensuite une idée intuitive.

2.2.2 Sémantique de $PCTL^*$

Suivant l'approche formulée par Kripke [Kri63], on interprète les formules de $PCTL^*$ dans des graphes où chaque nœud est étiqueté par l'ensemble des propositions atomiques qu'il vérifie.

Définition 2.2.2 *(Structure de Kripke)*

Une structure de Kripke *S est un triplet $S = \langle Q_S, R_S, l_S\rangle$ où Q_S est un ensemble d'états, $R_S \subseteq Q_S \times Q_S$ est une relation d'accessibilité totale, et $l_S : Q_S \rightarrow 2^{Prop}$ est une fonction d'étiquetage.*

La restriction à des relations d'accessibilité totales (i.e. telles que tout état a au moins un successeur) permet certaines simplifications techniques, mais ne change rien d'essentiel. De plus, cette restriction est minime lorsque l'on souhaite modéliser le comportement de systèmes réactifs [Lam80, Eme90]: si un état n'a pas de successeur, on peut supposer que le système reste définitivement (i.e. toujours) dans cet état (celui-ci serait alors son propre successeur).

Etant donnée une structure de Kripke S, une *exécution* de S est une séquence infinie d'états $q_0.q_1\ldots$ telle que $q_i\ R_S\ q_{i+1}$ pour tout $i \geq 0$. Dans la suite, nous utiliserons les lettres π, σ ...pour désigner des exécutions.

On note $\Pi_S(q)$ l'ensemble de toutes les exécutions de S issues de l'état q, et $\Pi(S)$ l'ensemble de toutes les exécutions de S. Etant donnée une exécution $\pi = q_0.\ q_1 \ldots$, on désigne par $\pi(i)$ l'état q_i, $\pi_{|i}$ représente le préfixe $q_0.\ q_1 \ldots q_i$, et π^i représente le suffixe $q_i.\ q_{i+1}.\ q_{i+2}\ldots$ On peut remarquer que π^i est une exécution et non $\pi_{|i}$.

De manière générale, les formules de $PCTL^*$ expriment des propriétés sur *un état le long d'une exécution d'une structure* S. On notera $\pi, n \models_S f$ pour signifier que l'état $\pi(n)$ le long de π satisfait la formule f. Intuitivement, $\pi(n)$ représente l'état courant, le préfixe $\pi_{|n}$ le passé, et π^n un futur possible à partir de $\pi(n)$.

Dans cette étude, le passé a trois caractéristiques: il est fini, déterministe et cumulatif (l'histoire d'un système augmente au fil du temps en incorporant les états traversés). Ces trois caractéristiques correspondent bien à l'idée intuitive que l'on a du passé d'un système réactif. D'autres choix sont possibles : par exemple, dans [Sti92] le passé n'est pas déterministe, on peut alors quantifier sur toutes les exécutions permettant d'atteindre un état donné. Enfin on peut noter que le fait de considérer un passé fini et déterministe et un futur non-déterministe correspond à une vision ockhamiste du temps [Pri67, ZC93].

Etant donnée une structure de Kripke S, nous allons définir par induction sur f la relation $\models_S$:

Définition 2.2.3 *(Sémantique de $PCTL^*$)*

$\pi, n \models_S a$	*ssi*	$a \in l_S(\pi(n))$,
$\pi, n \models_S f \wedge g$	*ssi*	$\pi, n \models_S f$ *et* $\pi, n \models_S g$,
$\pi, n \models_S \neg f$	*ssi*	$\pi, n \not\models_S f$,
$\pi, n \models_S \mathsf{E} f$	*ssi*	*il existe* $\pi' \in \Pi(S)$ *avec* $\pi'_{\vert n} = \pi_{\vert n}$ *t.q.* $\pi', n \models_S f$,
$\pi, n \models_S f \mathsf{U} g$	*ssi*	*il existe* $k \geq n$ *t.q.* $\pi, k \models_S g$ *et* $\pi, i \models_S f$ *pour tout* $n \leq i < k$,
$\pi, n \models_S \mathsf{X} f$	*ssi*	$\pi, n+1 \models_S f$,
$\pi, n \models_S f \mathsf{S} g$	*ssi*	*il existe* $0 \leq k \leq n$ *t.q.* $\pi, k \models_S g$ *et* $\pi, i \models_S f$ *pour tout* $k < i \leq n$,
$\pi, n \models_S \mathsf{X}^{-1} f$	*ssi*	$n > 0$ *et* $\pi, n-1 \models_S f$.

On omettra l'indice S dans $\models_S$ lorsqu'aucune confusion ne sera possible. Nous pouvons donner une idée intuitive de la sémantique des opérateurs temporels de $PCTL^*$:

- $\mathsf{E}\, f$ est vraie dans l'état courant $\pi(n)$ si il existe un futur possible de $\pi_{\vert n}$ pour lequel f est vraie. L'opérateur E permet ainsi de quantifier sur les futurs possibles à partir de l'état courant.

- $\mathsf{A}\, f$ est vraie dans l'état courant $\pi(n)$ si, toutes les exécutions π' à partir de cet état et ayant le même passé vérifient f en n.

- π, n satisfait $f\ \mathsf{U}\ g$ lorsque f est vraie le long de π à partir de l'état courant jusqu'à ce qu'on atteigne un état vérifiant g.

- π, n satisfait $\mathsf{X}\, f$ si f sera vraie à l'instant suivant le long de π.

- π, n vérifie $f\ \mathsf{S}\ g$ si g a été vraie dans le passé et si, depuis (pour les positions intermédiaires), f est vraie.

- π, n satisfait $\mathsf{X}^{-1} f$ si f a été vraie à l'instant précédent (nécessitant $n > 0$).

- π, n satisfait $\mathsf{F}\, f$ (resp. $\mathsf{F}^{-1} f$) si il existe un instant dans le futur (resp. dans le passé) où f est vraie.

- π, n satisfait $\mathsf{G}\, f$ (resp. $\mathsf{G}^{-1} f$) si f est vraie le long de π pour tous les instants futurs (resp. passés) de $\pi(n)$.

- π, n satisfait $\overset{\infty}{\mathsf{F}}\ f$ si f est vraie pour une infinité d'instants futurs sur π.

- π, n satisfait $\overset{\infty}{\mathsf{G}}\ f$ si il existe une position $k \geq n$ sur π à partir de laquelle f sera toujours vraie sur π.

Remarque La sémantique de notre opérateur E diffère de celle utilisée par Hafer et Thomas [HT87] pour leur version de $PCTL^*$. Leur définition est la suivante: $\pi, n \models_S \mathsf{E}\, f$ ssi il existe $\pi' \in \Pi(\pi(n))$ tq $\pi', 0 \models_S f$. L'opérateur E ainsi défini permet donc de quantifier sur le futur mais "oublie" le passé de l'état courant (celui-ci est alors considéré comme l'état initial de la nouvelle exécution choisie). Nous reviendrons sur cette définition au cours du chapitre 5 qui concerne les logiques avec l'opérateur Now. $\square$

A partir de la définition de $\models_S$, nous pouvons définir plusieurs notions importantes:

$\pi \models_S f$	$\overset{\text{def}}{\Leftrightarrow}$	$\pi, 0 \models_S f$	*"l'exécution π satisfait f"*
$q \models_S f$	$\overset{\text{def}}{\Leftrightarrow}$	$\pi \models_S f$ pour tout $\pi \in \Pi(q)$	*"l'état q satisfait f"*
$S \models f$	$\overset{\text{def}}{\Leftrightarrow}$	$\pi \models_S f$ pour tout $\pi \in \Pi(S)$	*"la structure S satisfait f"*
$\models_g f$	$\overset{\text{def}}{\Leftrightarrow}$	$\pi, n \models_S f$ pour tout (π, n) et pour tout S	*"f est (globalement) valide"*
$\models_i f$	$\overset{\text{def}}{\Leftrightarrow}$	$\pi, 0 \models_S f$ pour tout π et tout S	*"f est initialement valide"*

On peut noter que $\models_g f$ implique $\models_i f$ mais la réciproque est fausse. En fait, le lien entre $\models_g$ et $\models_i$ est établi par la proposition suivante:

Proposition 2.2.4 *$\models_g f$ ssi $\models_i \mathsf{G} f$.*

On peut remarquer que la définition de $S \models f$ dépend de la valeur de vérité de f pour l'état initial de chaque exécution de S et non de la valeur de vérité de f pour chaque (π, n) de S. En fait, lorsqu'on construit une spécification $Spec$ pour un système donné, on dispose, en général, d'un état initial p_0 et on dira que le système satisfait sa spécification lorsque toutes les exécutions **issues de** p_0 vérifient $Spec$. En effet, il est très pratique, pour spécifier, de ne considérer que ces exécutions [MP89]: par exemple, lorsque l'on veut énoncer que "un problème est toujours précédé par une cause", on souhaite utiliser la formule $\mathsf{AG}\ (\mathit{probleme} \Rightarrow \mathsf{F}^{-1}\mathit{cause})$ [1], mais ceci n'est possible que si l'on considère des exécutions qui partent de l'état initial du système [2]. Dans ce cadre, on pourra donc considérer que deux spécifications sont équivalentes si leurs valeurs de vérité en $(\pi, 0)$ sont identiques pour toute exécution π de toute structure S.

Ces considérations nous amènent à définir deux équivalences entre formules. La première, *l'équivalence globale*, correspond à la notion classique d'équivalence: deux formules sont équivalentes si pour tout modèle π, n leur valeur de vérité est identique. La seconde est *l'équivalence initiale*: deux formules sont équivalentes initialement si pour toute exécution π, leur valeur de vérité en $\pi, 0$ est identique:

[1] qui énonce qu'à tout moment (AG), si la proposition *probleme* est vraie alors il y a eu (précédemment) un état vérifiant *cause*.

[2] Ceci n'est pas dû à la présence d'opérateurs du passé dans la formule: on a le même problème avec la formule $\neg\mathsf{E}\,(\neg \mathit{cause})\ \mathsf{U}\ (\neg \mathit{cause} \wedge \mathit{probleme})$ qui est, nous le verrons ultérieurement, équivalente d'un certain point de vue.

Définition 2.2.5 *1. On dit que deux formules f et g sont équivalentes (globalement) si et seulement si pour tout π, n de toute structure S, on a:*
$\pi, n \models f \quad \Leftrightarrow \quad \pi, n \models g$. *On note:* $f \equiv g$.

2. On dit que deux formules f et g sont équivalentes initialement si et seulement si pour tout π de toute structure S, on a:
$\pi, 0 \models f \quad \Leftrightarrow \quad \pi, 0 \models g$. *On note:* $f \equiv_i g$.

L'équivalence initiale est plus faible que l'équivalence globale: il est clair que $f \equiv g \quad \Rightarrow \quad f \equiv_i g$. La réciproque n'est pas vraie puisqu'on a clairement $\mathsf{X}^{-1}\top \equiv_i \bot$ et $\mathsf{X}^{-1}\top \not\equiv \bot$. Mais les remarques faites précédemment montrent qu'étant données deux spécifications f et f' équivalentes initialement, alors pour tout système S, on a clairement $S \models f$ ssi $S \models f'$. C'est donc cette équivalence que nous retiendrons pour comparer les spécifications.

Concernant les opérateurs du passé et l'équivalence initiale, nous avons la proposition suivante:

Proposition 2.2.6 *Pour toutes formules f et g de $PCTL^*$, on a:*

$$\mathsf{X}^{-1} f \;\equiv_i\; \bot \qquad f\,\mathsf{S}\,g \;\equiv_i\; g \qquad \mathsf{F}^{-1} f \;\equiv_i\; f \tag{2.1}$$

Bien entendu, pour deux formules ne contenant pas d'opérateur du passé, l'équivalence globale et l'équivalence initiale coïncident. De plus, nous avons:

Proposition 2.2.7

- $f \equiv g$ *ssi* $\models_g f \Leftrightarrow g$
- $f \equiv_i g$ *ssi* $\models_i f \Leftrightarrow g$

L'équivalence globale est une congruence pour tous les opérateurs de $PCTL^*$: étant donnée une sous-formule g de f, la formule f' obtenue après substitution de g par une formule g' équivalente, est équivalente à f.

Mais l'équivalence initiale n'est une congruence que pour les opérateurs booléens, le quantificateur E et les opérateurs du passé: étant donnée une sous-formule g de f, alors si g n'apparaît pas dans la portée d'opérateurs du futur (U ou X), la formule f' obtenue après substitution de g par une formule g' équivalente initialement, est équivalente initialement à f.

2.3 Propriétés de base

Dans cette section, nous allons rappeler une série de propriétés classiques (voir [Eme90, MP92]) concernant les opérateurs de $PCTL^*$.

2.3.1 Les opérateurs X et X^{-1}

Nous rappelons les règles de distributivité de ces opérateurs par rapport aux combinateurs booléens:

$$\begin{array}{lll@{\qquad}lll}
\mathsf{X}\,(f \wedge g) & \equiv & \mathsf{X}\,f \wedge \mathsf{X}\,g & \mathsf{X}^{-1}\,(f \wedge g) & \equiv & \mathsf{X}^{-1}\,f \wedge \mathsf{X}^{-1}\,g \\
\mathsf{X}\,(f \vee g) & \equiv & \mathsf{X}\,f \vee \mathsf{X}\,g & \mathsf{X}^{-1}\,(f \vee g) & \equiv & \mathsf{X}^{-1}\,f \vee \mathsf{X}^{-1}\,g \\
\mathsf{X}\,(\neg f) & \equiv & \neg\mathsf{X}\,f & \mathsf{X}^{-1}\,(\neg f) & \equiv & \neg\mathsf{X}^{-1}\,f \wedge \mathsf{X}^{-1}\,\top \\
 & & & \neg\mathsf{X}^{-1}\,f & \equiv & \mathsf{X}^{-1}\,\neg f \vee \neg\mathsf{X}^{-1}\,\top
\end{array}$$

La différence quant à l'effet de la négation sur le *Next* et le *Previous*, est due au fait que l'on considère des exécutions infinies (à droite [3] mais pas à gauche). Ainsi chaque état $\pi(n)$ a-t-il un successeur sur π, mais il n'a pas toujours de prédécesseur. Lorsqu'un état ne vérifie pas $\mathsf{X}^{-1}\,f$, il y a deux cas possibles: soit l'état précédent ne vérifiait pas f, soit il n'y avait pas d'état précédent.

On en déduit les deux équivalences suivantes:

$$\mathsf{X}\,\mathsf{X}^{-1}\,f \equiv f \qquad\qquad \mathsf{X}^{-1}\,\mathsf{X}\,f \equiv \mathsf{X}^{-1}\,\top \wedge f$$

2.3.2 Les opérateurs F , F^{-1}, G et G^{-1}

Nous avons les propriétés suivantes qui énoncent la distributivité du $\vee$ (resp. $\wedge$) pour les opérateurs F et F^{-1} (resp. G et G^{-1}), et dualité de F et G (resp. F^{-1} et G^{-1}).

$$\begin{array}{lll@{\qquad}lll}
\mathsf{F}\,(f \vee g) & \equiv & \mathsf{F}\,f \vee \mathsf{F}\,g & \mathsf{F}^{-1}(f \vee g) & \equiv & \mathsf{F}^{-1}f \vee \mathsf{F}^{-1}g \\
\mathsf{G}\,(f \wedge g) & \equiv & \mathsf{G}\,f \wedge \mathsf{G}\,g & \mathsf{G}^{-1}(f \wedge g) & \equiv & \mathsf{G}^{-1}f \wedge \mathsf{G}^{-1}g \\
\mathsf{F}\,(\neg f) & \equiv & \neg\mathsf{G}\,f & \mathsf{F}^{-1}(\neg f) & \equiv & \neg\mathsf{G}^{-1}f \\
\mathsf{G}\,(\neg f) & \equiv & \neg\mathsf{F}\,f & \mathsf{G}^{-1}(\neg f) & \equiv & \neg\mathsf{F}^{-1}f
\end{array}$$

Ces résultats sont naturels: pouvoir atteindre un état vérifiant f ou vérifiant g, est bien équivalent à pouvoir atteindre un état vérifiant f **ou** pouvoir atteindre un état vérifiant g. Mais pouvoir atteindre un état vérifiant f et g implique (et n'est pas équivalent à) pouvoir atteindre un état vérifiant f **et** pouvoir atteindre un état vérifiant g...

De plus, nous avons les propriétés d'idempotence suivantes:

$$\begin{array}{lll@{\qquad}lll}
\mathsf{F}\,\mathsf{F}\,f & \equiv & \mathsf{F}\,f & \mathsf{F}^{-1}\mathsf{F}^{-1}f & \equiv & \mathsf{F}^{-1}f \\
\mathsf{G}\,\mathsf{G}\,f & \equiv & \mathsf{G}\,f & \mathsf{G}^{-1}\mathsf{G}^{-1}f & \equiv & \mathsf{G}^{-1}f
\end{array}$$

[3] Ceci est toujours possible car R_S est totale.

2.3.3 les opérateurs U et S

Les règles de distributivité pour la disjonction et la conjonction sont les suivantes:

$$\begin{array}{rclrcl} f\,\mathsf{U}\,(g \vee h) & \equiv & f\,\mathsf{U}\,g \vee f\,\mathsf{U}\,h & f\,\mathsf{S}\,(g \vee h) & \equiv & f\,\mathsf{S}\,g \vee f\,\mathsf{S}\,h \\ (f \wedge g)\,\mathsf{U}\,h & \equiv & f\,\mathsf{U}\,h \wedge g\,\mathsf{U}\,h & (f \wedge g)\,\mathsf{S}\,h & \equiv & f\,\mathsf{S}\,h \wedge g\,\mathsf{S}\,h \end{array}$$

Concernant la négation, nous avons les deux équivalences suivantes:

$$\begin{array}{rcl} \neg(f\,\mathsf{U}\,g) & \equiv & \Big((\neg g)\,\mathsf{U}\,(\neg f \wedge \neg g)\Big) \vee \mathsf{G}\,\neg g \\ \neg(f\,\mathsf{S}\,g) & \equiv & \Big((\neg g)\,\mathsf{S}\,(\neg f \wedge \neg g)\Big) \vee \mathsf{G}^{-1}\neg g \end{array} \tag{2.2}$$

Les équivalences (2.2) s'expliquent par le fait qu'un état vérifie $f\,\mathsf{U}\,g$ si (1) g est vraie dans le futur, et si (2) f est vraie tant g n'est pas vérifiée. La négation d'une telle propriété est donc bien: (1) g n'est jamais vraie, ou (2) avant d'arriver à un état vérifiant g, on passe par un état ne vérifiant ni f ni g. On fait un raisonnement identique pour l'opérateur S.

Nous avons aussi les propositions suivantes qui permettent de donner une caractérisation en terme de point fixe des opérateurs U et S:

$$\begin{array}{rcl} f\,\mathsf{U}\,g & \equiv & g \vee \Big(f \wedge \mathsf{X}\,(f\,\mathsf{U}\,g)\Big) \\ f\,\mathsf{S}\,g & \equiv & g \vee \Big(f \wedge \mathsf{X}^{-1}\,(f\,\mathsf{S}\,g)\Big) \end{array}$$

De plus, certaines logiques imposent que tout opérateur U ou X soit immédiatement précédé par un quantificateur E ou A (c'est le cas de CTL), on peut alors étendre ces propositions à l'opérateur $\mathsf{E}\,\_\,\mathsf{U}$:

$$\mathsf{E}\,a\,\mathsf{U}\,b \equiv b \vee \Big(a \wedge \mathsf{EX}\,(\mathsf{E}\,a\,\mathsf{U}\,b)\Big) \tag{2.3}$$

Ces caractérisations sont notamment utilisées dans les algorithmes de vérification automatique où, pour calculer l'ensemble des états qui vérifient par exemple $\mathsf{E}\,a\,\mathsf{U}\,b$, on cherche le plus petit point fixe de l'équation obtenue à partir de (2.3).

2.3.4 Les opérateurs $\overset{\infty}{\mathsf{F}}$ et $\overset{\infty}{\mathsf{G}}$

Nous avons les équivalences suivantes concernant la distributivité, l'idempotence et la dualité pour les opérateurs $\overset{\infty}{\mathsf{F}}$ et $\overset{\infty}{\mathsf{G}}$:

$$\begin{array}{rclrclrcl} \overset{\infty}{\mathsf{F}}\,(f \vee g) & \equiv & \overset{\infty}{\mathsf{F}}\,f \vee \overset{\infty}{\mathsf{F}}\,g & \neg\,\overset{\infty}{\mathsf{F}}\,f & \equiv & \overset{\infty}{\mathsf{G}}\,\neg f & \overset{\infty}{\mathsf{F}}\overset{\infty}{\mathsf{F}}\,f & \equiv & \overset{\infty}{\mathsf{F}}\,f \\ \overset{\infty}{\mathsf{G}}\,(f \wedge g) & \equiv & \overset{\infty}{\mathsf{G}}\,f \wedge \overset{\infty}{\mathsf{G}}\,g & \neg\,\overset{\infty}{\mathsf{G}}\,f & \equiv & \overset{\infty}{\mathsf{F}}\,\neg f & \overset{\infty}{\mathsf{G}}\overset{\infty}{\mathsf{G}}\,f & \equiv & \overset{\infty}{\mathsf{G}}\,f \end{array}$$

De plus, il est clair que $\overset{\infty}{\mathsf{F}}\ (f \wedge g)$ implique (mais n'est pas équivalent à) $\overset{\infty}{\mathsf{F}}\ f \wedge \overset{\infty}{\mathsf{F}}\ g$. D'autre part, nous avons aussi les propositions suivantes:

$$\begin{array}{l} \mathsf{G}\ \overset{\infty}{\mathsf{F}}\ f\ \equiv \overset{\infty}{\mathsf{F}}\ \mathsf{F}\, f\ \equiv \overset{\infty}{\mathsf{G}}\overset{\infty}{\mathsf{F}}\ f\ \equiv\ \mathsf{F}\ \overset{\infty}{\mathsf{F}}\ f\ \equiv \overset{\infty}{\mathsf{F}}\ f \\ \mathsf{F}\ \overset{\infty}{\mathsf{G}}\ f\ \equiv \overset{\infty}{\mathsf{G}}\ \mathsf{G}\, f\ \equiv \overset{\infty}{\mathsf{F}}\overset{\infty}{\mathsf{G}}\ f\ \equiv\ \mathsf{G}\ \overset{\infty}{\mathsf{G}}\ f\ \equiv \overset{\infty}{\mathsf{G}}\ f \end{array}$$

2.3.5 Les quantificateurs E et A

Voici maintenant les propriétés des quantificateurs E et A par rapport aux opérateurs booléens:

$$\begin{array}{lcl@{\qquad}lcl} \mathsf{E}\,(f \vee g) & \equiv & \mathsf{E}\, f \vee \mathsf{E}\, g & \mathsf{A}\,(f \wedge g) & \equiv & \mathsf{A}\, f \wedge \mathsf{A}\, g \\ \neg\mathsf{E}\, f & \equiv & \mathsf{A}\, \neg f & \neg\mathsf{A}\, f & \equiv & \mathsf{E}\, \neg f \end{array}$$

De plus, nous avons: $\mathsf{E}\,\mathsf{E}\, f \equiv \mathsf{E}\, f$ et $\mathsf{A}\,\mathsf{A}\, f \equiv \mathsf{A}\, f$.

2.4 De nouveaux opérateurs

Il existe une grande variété d'opérateurs temporels dans la littérature. Parmi ceux-ci on distingue ceux qui peuvent être définis à partir des opérateurs de $PCTL^*$ et ceux qui permettent l'expression de propriétés inexprimables avec $PCTL^*$. Dans cette section, nous allons principalement nous intéresser à des formes particulières de *Since* et de *Until* qui sont définissables avec $PCTL^*$.

2.4.1 *Until* et *Since* stricts

Nous pouvons considérer des versions strictes des opérateurs *Until* et *Since* avec la sémantique suivante:

$$\begin{array}{lll} \pi, n \models f\ \mathsf{U}_s\ g & \text{ssi} & \text{il existe } k > n \text{ t.q. } \pi, k \models g \text{ et } \pi, i \models f \text{ pour tout } n < i < k, \\ \pi, n \models f\ \mathsf{S}_s\ g & \text{ssi} & \text{il existe } 0 \leq k < n \text{ t.q. } \pi, k \models g \text{ et } \pi, i \models f \text{ pour tout } k < i < n, \end{array}$$

Ces opérateurs U_s et S_s sont, par exemple, utilisés par Gabbay dans [Gab87]. En fait, on peut définir U_s (resp. S_s) avec U et X (resp. S et X^{-1}) et réciproquement:

$$\begin{array}{lcl@{\qquad}lcl} f\,\mathsf{U}_s\, g & \equiv & \mathsf{X}\,(f\,\mathsf{U}\, g) & f\,\mathsf{U}\, g & \equiv & g \vee \Big(f \wedge (f\,\mathsf{U}_s\, g)\big) \\ & & & \mathsf{X}\, f & \equiv & \bot\,\mathsf{U}_s\, f \end{array}$$

$$\begin{array}{lcl@{\qquad}lcl} f\,\mathsf{S}_s\, g & \equiv & \mathsf{X}^{-1}\,(f\,\mathsf{S}\, g) & f\,\mathsf{S}\, g & \equiv & g \vee \Big(f \wedge (f\,\mathsf{S}_s\, g)\big) \\ & & & \mathsf{X}^{-1}\, f & \equiv & \bot\,\mathsf{S}_s\, f \end{array}$$

On peut aussi définir $\mathsf{F}_s\, f \stackrel{\text{def}}{\equiv} \top\, \mathsf{U}_s\, f$ et $\mathsf{F}_s^{-1} f \stackrel{\text{def}}{\equiv} \top\, \mathsf{S}_s\, f$. Dans la suite de l'étude, nous nous intéresserons beaucoup à l'opérateur F_s^{-1}. En effet, il permet d'exprimer facilement l'opérateur F^{-1} ($\mathsf{F}^{-1} f \equiv f \vee \mathsf{F}_s^{-1} f$) et nous avons aussi les propositions suivantes:

$$\neg \mathsf{X}^{-1}\, \top \equiv \neg \mathsf{F}_s^{-1} \top \qquad\qquad \mathsf{X}^{-k}\, \mathsf{F}^{-1} f \equiv \mathsf{F}_s^{-k} f$$

où X^{-k} (resp. F_s^{-k}) dénote l'emboîtement de k opérateurs X^{-1} (resp. F_s^{-1}). Au chapitre 4, nous comparerons les intérêts respectifs de F^{-1} et F_s^{-1} pour la spécification.

2.4.2 "*Unless*" ou "*Waiting for*"

Nous pouvons définir l'opérateur W ("*Unless*", "*Waiting for*" ou "*Weak Until*"). Informellement une exécution vérifie $f \mathsf{W}\, g$ si f est vraie le long de cette exécution tant que g n'est pas vraie:

$$\pi, n \models f\, \mathsf{W}\ g \quad \text{ssi} \quad \text{soit } \pi, n \models f\, \mathsf{U}\, g, \text{ soit } \pi, n \models \mathsf{G}\, f$$

Clairement on a: $f\, \mathsf{U}\, g \equiv f \mathsf{W}\, g \wedge \mathsf{F}\, g$ et $\mathsf{G}\, f \equiv f \mathsf{W} \perp$. Nous avons aussi les deux équivalences suivantes:

$$\begin{array}{rcl} \neg(f\, \mathsf{U}\, g) & \equiv & (\neg g) \mathsf{W}\, (\neg f \wedge \neg g) \\ \neg(f \mathsf{W}\, g) & \equiv & (\neg g)\, \mathsf{U}\, (\neg f \wedge \neg g) \end{array}$$

L'opérateur W permet notamment d'exprimer la "propriété de devancement" [4] définie dans [MP81]: on note $a\, \mathsf{P}\, b$ pour signifier qu'une occurrence de a précède la première occurrence de b si celle-ci existe. En fait on a:

$$a\, \mathsf{P}\, b \equiv (\neg b) \mathsf{W}\, (\neg b \wedge a)$$

Ce type de formulation est très utile pour la spécification, nous y reviendrons au chapitre 3.

2.4.3 "*Back-to*"

De la même manière, il existe l'opérateur B ("*Back-to*") qui est l'équivalent pour le *Since*:

$$\pi, n \models f\, \mathsf{B}\ g \quad \text{ssi} \quad \text{soit } \pi, n \models f\, \mathsf{S}\, g, \text{ soit } \pi, n \models \mathsf{G}^{-1} f$$

[4] *precedence property*

On a : $f \,\mathsf{S}\, g \equiv f \mathsf{B}\, g \;\wedge\; \mathsf{F}^{-1} g$ et $\mathsf{G}^{-1} f \equiv f \mathsf{B}\, \bot$. De la même manière que pour W et U, on a :

$$
\begin{array}{rcl}
\neg(f \,\mathsf{S}\, g) & \equiv & (\neg g) \mathsf{B}\, (\neg f \;\wedge\; \neg g) \\
\neg(f \mathsf{B}\, g) & \equiv & (\neg g) \,\mathsf{S}\, (\neg f \;\wedge\; \neg g)
\end{array}
$$

2.4.4 Et encore...

Cette liste déjà longue est loin d'être exhaustive. On peut trouver d'autres opérateurs dans [Kam68, BdG92] définissables à partir de U, S, X et X^{-1}.

Beaucoup d'autres opérateurs ne peuvent pas être définis à l'aide des opérateurs de $PCTL^*$. Il en est ainsi de l'opérateur *Chop* [HKP82, RP86] qui permet d'exprimer qu'une exécution est la concaténation de deux séquences d'états vérifiant chacune une certaine propriété (on définit alors la sémantique des combinateurs sur des exécutions éventuellement finies).

D'autre part, Wolper [Wol83] a aussi proposé une logique étendue où l'on peut définir ses propres opérateurs sous forme de grammaire régulière. Il est alors possible, par exemple, de définir un opérateur G_2 tel que $\mathsf{G}_2 a$ soit vraie ssi a est vraie pour tous les états pairs (depuis l'état initial) alors que cette propriété est inexprimable en $PCTL^*$.

2.5 Formule d'état, du futur ou d'histoire

La logique $PCTL^*$ est une logique riche et contient des formules très différentes. Ainsi regroupe-t-elle à la fois les différents fragments du temps arborescent et ceux du temps linéaire souvent considérés dans la littérature. Dans les logiques du temps arborescent, les opérateurs U et X sont toujours placés dans la portée d'un quantificateur E ou A. Dans les logiques du temps linéaire, les formules ne contiennent pas de quantificateur E ou A : une telle formule s'interprète sur une exécution et chaque état ne dispose alors que d'un seul futur possible.

Cette diversité de la logique $PCTL^*$ rend utile la définition de plusieurs familles ou groupes de formules ayant des caractéristiques communes. Nous avons vu que les formules de $PCTL^*$ expriment des propriétés sur un état $\pi(n)$ le long d'une exécution π dans une structure S. Nous allons partir de cette remarque pour différencier plusieurs familles de formules de $PCTL^*$. Nous distinguerons les formules qui expriment des propriétés sur le préfixe $\pi_{|n}$, celles qui expriment des propriétés sur le suffixe π^n et celles exprimant des propriétés sur l'état courant $\pi(n)$. On va donc définir trois familles de formules à l'aide de critères sémantiques:

Définition 2.5.1 • *Une formule f de $PCTL^*$ est* une formule du futur *si la valeur de vérité de f en (π, n) ne dépend que du futur sélectionné (le suffixe π^n), i.e. ssi :*

$$\pi^n = \pi'^m \ \ implique \ \ \pi, n \models f \ \Leftrightarrow \ \pi', m \models f$$

- *Une formule f de $PCTL^*$ est* une formule d'histoire *si la valeur de vérité de f en (π, n) ne dépend que de l'histoire $\pi_{|n}$, i.e. ssi :*

$$\pi_{|n} = \pi'_{|n} \ \ implique \ \ \pi, n \models f \ \Leftrightarrow \ \pi', n \models f$$

- *Une formule f de $PCTL^*$ est* une formule d'état *si la valeur de vérité de f en (π, n) ne dépend que de l'état courant, i.e. ssi :*

$$\pi(n) = \pi'(m) \ \ implique \ \ \pi, n \models f \ \Leftrightarrow \ \pi', m \models f$$

Nous allons aussi définir trois familles de formules sur la base de critères syntaxiques:

Définition 2.5.2
- *Une formule de $PCTL^*$ est linéaire si elle ne contient aucun quantificateur E ou A.*
- *Une formule de $PCTL^*$ est pur-futur si elle ne contient aucun opérateur du passé (S, X^{-1}).*
- *Une formule de $PCTL^*$ est pur-passé si elle ne contient aucun opérateur du futur (U, X).*

Clairement, une formule linéaire exprimera des propriétés sur l'exécution π et sa valeur de vérité ne dépendra pas des éventuelles exécutions possibles à partir des états de π (puisqu'aucun quantificateur E ou A ne pourra y faire référence). D'autre part, toute formule pur-futur est une formule du futur, et toute formule pur-passé est une formule d'histoire. La réciproque est fausse puisque la formule $\mathsf{EX}\,\mathsf{X}^{-1}\,\mathsf{X}^{-1}\,a$ est une formule d'histoire (équivalente à $\mathsf{X}^{-1}\,a$) mais n'est pas pur-passé.

Les formules des logiques du temps arborescent (comme CTL) classiquement utilisées dans la littérature sont des formules d'état. Si on ajoute à ces logiques des opérateurs du passé (comme $CTL + \mathsf{S}$), alors les formules ainsi construites deviennent des formules d'histoire. On peut aussi noter que si f est une formule d'histoire, alors $\mathsf{E}\, f \equiv f$.

Pour définir les différents fragments de $PCTL^*$ nous allons utiliser la notation considérée dans [EH86]:

- Nous noterons $B(C_1, \ldots, C_n)$ l'ensemble des formules de $PCTL^*$ où les seuls opérateurs temporels autorisés sont les C_i et où chaque opérateur C_i est dans la portée immédiate d'un quantificateur E ou A.

  Par exemple: $\mathsf{EX}\,(a \wedge \mathsf{AF}\, b) \in B(\mathsf{X}, \mathsf{F})$, mais $\mathsf{EX}\,(a \wedge \mathsf{F}\, b) \notin B(\mathsf{X}, \mathsf{F})$.

- Nous noterons $B(C_1, \ldots, C_n, \neg, \wedge)$ l'ensemble des formules de $PCTL^*$ où les seuls opérateurs temporels sont les C_i, chacun étant dans la portée de quantificateurs E ou A, et où les seuls opérateurs autorisés entre les C_i et les quantificateurs sont $\neg$ et $\wedge$.

  Par exemple: $\mathsf{E}\,(\mathsf{F}\,a \wedge \neg \mathsf{F}\,b) \in B(\mathsf{F}, \neg, \wedge)$, mais $\mathsf{E}\,(\mathsf{F}\,a \wedge \mathsf{F}\,\neg \mathsf{F}\,b) \notin B(\mathsf{F}, \neg, \wedge)$.

- Enfin, on dénote par $L(C_1, \ldots, C_n)$ l'ensemble des formules de $PCTL^*$ construites avec les combinateurs booléens et les opérateurs temporels C_i (les quantificateurs E et A ne sont pas autorisés).

  Par exemple: $(\mathsf{F}\,a \wedge (b\,\mathsf{U}\,c)) \in L(\mathsf{U})$, mais $\mathsf{EF}\,a \notin L(\mathsf{U})$.

La notation $B(C, \ldots)$ nous permet de représenter aisément les différentes logiques du temps arborescent tandis que $L(C, \ldots)$ désigne les logiques de temps linéaire.

2.6 Les fragments classiques de $PCTL^*$

Nous allons maintenant rappeler les principales logiques que l'on rencontre dans la littérature et qui peuvent se définir comme des fragments de $PCTL^*$.

2.6.1 La logique CTL^*

La logique CTL^* a été proposée dans [EH86]. Formellement les formules de CTL^* sont les formules pur-futur de $PCTL^*$ où chaque opérateur de temps linéaire (U et X) est dans la portée (pas forcément immédiate) d'au moins un quantificateur E ou A. CTL^* est donc une logique d'état (elle se situe dans le cadre du temps arborescent).

Alors qu'un débat opposait les partisans du temps linéaire à ceux du temps arborescent, l'introduction de CTL^* a permis de donner un cadre commun à ces deux approches. En effet, la syntaxe de CTL^* autorise l'emboîtement des opérateurs U et X (sans nécessité d'intercaler des quantificateurs) et permet ainsi une expressivité très riche: la comparaison avec les logiques du temps linéaire devenait ainsi possible après avoir étendu la notion $\pi \models f_L$ (utilisée pour les formules du temps linéaire) à $S \models f_L$ (ssi $\forall \pi \in S, \pi \models f_L$) utilisée pour les formules de CTL^*.

Le problème du Model-checking pour CTL^* (*étant donnés un modèle S et une formule f de CTL^*, déterminer pour chaque état q de S si $q \models_S f$*) est PSPACE-complet [CES83]. Ceci constitue une sérieuse limite à l'emploi de CTL^* pour la vérification automatique. On verra que le problème équivalent pour le seul fragment linéaire de CTL^* est lui aussi PSPACE-complet.

Exemple 2.6.1 $\mathsf{E}\Big(\big(a\,\mathsf{U}\,b \vee c\,\mathsf{U}\,d\big)\,\mathsf{U}\,e\Big) \in CTL^*$

Cette formule est vraie si il existe une exécution π sur laquelle on peut atteindre un état q où e est vraie, tout en vérifiant précédemment (entre l'état courant et q) $a \mathsf{U} b$ ou $c \mathsf{U} d$ le long le π.

2.6.2 La logique $PCTL^*$ de [HT87]

La logique $PCTL^*$ définie par [HT87] diffère de celle que nous utilisons ici par la sémantique donnée au quantificateur E. Dans cette étude, nous désignerons cette logique par $PCTL^*_{HT}$. Nous verrons notamment que le théorème énonçant l'équivalence initiale entre $PCTL^*_{HT}$ et CTL^* s'adapte facilement à $PCTL^*$.

2.6.3 Les logiques de temps linéaire: LTL, PTL, $PPTL$

La logique LTL correspond au fragment linéaire $L(\mathsf{F}, \mathsf{X})$. Une variante de cette logique, $L(\mathsf{F})$, a été proposée dans [Lam80].

On définit le problème du model-checking pour le temps linéaire de la manière suivante: *étant données une structure S et une formule f_L, déterminer pour chaque état q de S si il existe une exécution π de S issue de q telle $\pi \models f_L$.* Le problème du Model-checking pour $L(\mathsf{F})$ est NP-complet et il est PSPACE-complet pour $L(\mathsf{F}, \mathsf{X})$ [SC85].

Exemple 2.6.2 $\mathsf{F}\Big(a \wedge \mathsf{X} b\Big) \in LTL$

La formule précédente est vraie pour une exécution π si il existe dans le futur un état de π où a est vraie et dont le successeur vérifie b.

La logique PTL correspond au fragment linéaire $L(\mathsf{U}, \mathsf{X})$ qui a été largement étudié (de même que son équivalent $L(\mathsf{U}_s))$ dans la littérature [Pnu81, GPSS80, MW84, Gab87, MP92]. On peut voir PTL comme la logique de référence pour le temps linéaire. Elle est aussi expressive que le langage du premier ordre muni de l'égalité, du prédicat binaire $<$ et des prédicats unaires a, b... interprété sur des modèles linéaires [GPSS80, Gab92, GHR92].

Le problème du model-checking pour PTL est PSPACE-complet [SC85]. Là encore, cette complexité limite l'utilisation de PTL pour la vérification automatique, néanmoins Lichtenstein et Pnueli [LP85] ont argué de l'existence d'un algorithme de vérification pour PTL dont la complexité est en $O(\mid S \mid .2^{|f|})$ pour défendre la faisabilité de la vérification pour PTL puisque la taille des spécifications est généralement faible (à la différence de la taille du modèle).

Exemple 2.6.3 $\mathsf{X} a \wedge \Big((b \vee \mathsf{X} a) \, \mathsf{U} c\Big) \in PTL$

La logique $PPTL$ correspond au fragment $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ [Kam68, GPSS80, LPZ85, Pnu86b]. Kamp [Kam68] a montré qu'elle était aussi expressive que le langage du premier ordre évoqué précédemment.

Exemple 2.6.4 $\mathsf{X}a \wedge \Big((b \vee \mathsf{X}^{-1} a)\, \mathsf{U}\, (d\, \mathsf{S}\, c)\Big) \in PPTL$

2.6.4 Les logique UB^-, UB, UB^+

La logique UB (*"United system of Branching time"*) [BPM83] correspond au fragment $B(\mathsf{F}, \mathsf{X})$ de $PCTL^*$. Chaque opérateur F ou X doit être immédiatement précédé par un quantificateur E ou A. On peut voir UB comme l'ensemble des formules construites avec les opérateurs suivants: $\wedge$, $\neg$, EX, EF, et AF. On autorise aussi AX (resp. AG, EG) le dual de EX (resp. EF, AF). La logique UB correspond à la logique $B(\mathsf{F})$ présentée dans [Lam80] étendue avec les opérateurs EX et AX.

Exemple 2.6.5 $\mathsf{EF}\Big(a \wedge \neg(\mathsf{AX}\, b \vee \mathsf{EG}\, c)\Big) \in UB$

La logique UB^- [MP79] contient les formules de UB construites avec les seuls opérateurs EF et EX (et leurs duaux AG et AX).

La logique UB^+ correspond au fragment $B(\mathsf{F}, \mathsf{X}, \neg, \wedge)$ de $PCTL^*$. Dans UB^+, chaque opérateur linéaire F ou X est dans la portée d'un quantificateur E ou A et les seuls opérateurs autorisés entre l'opérateur linéaire et le quantificateur sont les opérateurs booléens.

Exemple 2.6.6 $\mathsf{E}\Big(\mathsf{G}\, b \wedge \mathsf{F}\, a\Big) \begin{cases} \notin UB \\ \in UB^+ \end{cases}$

La formule précédente est vraie si il existe un chemin (à partir de l'état courant) le long duquel b est toujours vraie, et où l'on peut atteindre un état vérifiant a.

2.6.5 Les logiques CTL, CTL^+

La logique CTL [CE81] (*Computation Tree Logic*) correspond au fragment $B(\mathsf{U}, \mathsf{X})$. Dans CTL, chaque opérateur U et X est immédiatement dans la portée d'un quantificateur E ou A. On peut voir CTL comme la logique où les seuls opérateurs autorisés (en plus des opérateurs booléens) sont EX, $\mathsf{E\_U}$ et $\mathsf{A\_U}$. En plus d'une expressivité relativement riche, CTL bénéficie d'algorithmes de model-checking très efficaces [CG87] dont la complexité dépend linéairement de la taille du modèle et de la formule.

Concernant l'expressivité des opérateurs de CTL, nous pouvons remarquer que l'on peut définir $\mathsf{A\_U}$ avec $\mathsf{E\_U}$ et AF :

$$\mathsf{A}\, f\ \mathsf{U}\ g \equiv \mathsf{AF}\, g \ \wedge\ \neg\mathsf{E}\,(\neg g)\ \mathsf{U}\ (\neg f \wedge \neg g)$$

Ce résultat classique permet de simplifier les preuves par induction concernant CTL en permettant de ne considérer que les opérateurs EX, $\mathsf{E\_U}$ et AF. On peut noter que l'inverse (exprimer $\mathsf{E\_U}$ avec $\mathsf{A\_U}$ et EF) n'est pas possible [Lar94].

Exemple 2.6.7 $\mathsf{E}\, a\ \mathsf{U}\ (b \wedge \mathsf{EX}\, c) \begin{cases} \notin UB^+ \\ \in CTL \end{cases}$

La formule précédente est vraie si il existe une exécution le long de laquelle a est vraie jusqu'à ce que (1) b soit vraie et (2) il existe un successeur où c est vraie.

La logique CTL^+ correspond au fragment $B(\mathsf{X}, \mathsf{U}, \neg, \wedge)$. Son intérêt est de permettre d'exprimer aisément qu'une exécution vérifie plusieurs propriétés:

Exemple 2.6.8 $\mathsf{E}\Big(a\, \mathsf{U}\, b \wedge a'\, \mathsf{U}\, b'\Big) \begin{cases} \notin CTL \\ \in CTL^+ \end{cases}$

La formule précédente est vraie si il existe une exécution π vérifiant $a\ \mathsf{U}\, b$ et $a'\ \mathsf{U}\, b'$.

2.6.6 Les logiques $ECTL$, $ECTL^+$

Même si CTL est plus expressive que UB, elle ne permet pas d'exprimer plusieurs propriétés d'équité importantes. En effet, si il est possible d'énoncer qu'une propriété est vérifiée à une infinité d'instants le long de **toutes** les exécutions ($\mathsf{A}\, \overset{\infty}{\mathsf{F}}\ a \equiv \mathsf{AG\,AF}\, a$), il n'est pas possible d'énoncer en CTL qu'une proposition est vérifiée une infinité d'instants le long d'**une** exécution ($\mathsf{E}\, \overset{\infty}{\mathsf{F}}\ a$). La logique $ECTL$ (*"Extended CTL"*) [EH86] a donc été introduite pour pallier ce type de limitation.

La logique $ECTL$ correspond au fragment $B(\mathsf{X}, \mathsf{U}, \overset{\infty}{\mathsf{F}})$. Les formules de $ECTL$ sont construites de manière identique à celles de CTL mais avec deux opérateurs supplémentaires: $\mathsf{E}\, \overset{\infty}{\mathsf{F}}$ et $\mathsf{A}\, \overset{\infty}{\mathsf{F}}$.

Exemple 2.6.9 $\mathsf{E}\, \overset{\infty}{\mathsf{F}}\ (a \wedge \mathsf{EX}\, b) \begin{cases} \notin CTL \\ \in ECTL \end{cases}$

La formule précédente est vraie si il existe une exécution π le long de laquelle il y a une infinité d'instants vérifiant a et ayant un successeur (non obligatoirement sur π) vérifiant b.

Néanmoins $ECTL$ ne permet pas l'expression de la propriété d'équité $\mathsf{A}\, (\overset{\infty}{\mathsf{F}}\ a \Rightarrow \mathsf{F}\, b)$ [5] qui énonce que pour toutes les exécutions où a est vraie à une infinité

[5] correspond à la *strong eventual fairness* dans [Lam80]

d'instants, alors il existe un instant où b sera vraie. $ECTL$ ne permet pas non plus l'équité forte exprimée par la formule suivante $\mathsf{A}\,(\overset{\infty}{\mathsf{F}}\ a \Rightarrow \overset{\infty}{\mathsf{F}}\ b)$ qui énonce que pour toutes les exécutions où a est vraie à une infinité d'instants, alors b est vraie à une infinité d'instants. Ces deux formules appartiennent à la logique $ECTL^+$.

La logique $ECTL^+$ correspond au fragment $B(\mathsf{X}\,,\,\mathsf{U}\,,\overset{\infty}{\mathsf{F}}\,,\neg,\wedge)$. $ECTL^+$ est très proche de la logique $CTLF$ proposée dans [EC80].

Toujours dans l'objectif d'exprimer des propriétés d'équité, Emerson et Lei [EL87] ont proposé les logiques $FCTL$ et $GFCTL$. Une spécification en $FCTL$ est une paire (Φ_0, f) où Φ_0 est une formule d'équité construite avec l'opérateur $\overset{\infty}{\mathsf{F}}$, les combinateurs booléens et les propositions atomiques, et f est une formule CTL où chaque opérateur E et A ne quantifie que sur des exécutions vérifiant Φ_0. Cette idée d'isoler un ensemble d'exécutions et d'exprimer ensuite des propriétés sur cet ensemble, est intéressante et bien adaptée à la spécification des systèmes réactifs puisque l'on veut alors énoncer les propriétés que doivent vérifier les exécutions équitables du système étudié.

La logique $GFCTL$ étend $FCTL$ en associant à chaque quantificateur E et A une formule d'équité Φ particulière (construite avec l'opérateur $\overset{\infty}{\mathsf{F}}$, les combinateurs booléens et les formules de $GFCTL$). Clairement, les logiques $GFCTL$ et $ECTL^+$ ont la même expressivité.

Concernant le model-checking, il existe un algorithme [EL87] pour la logique $FCTL$ dont la complexité est en $O(|\,S\,|\,.\,|\,f\,|\,.\,|\,\Phi_0\,|)$ si la formule Φ_o a la forme suivante: $\bigvee_i \bigwedge_j (\overset{\infty}{\mathsf{F}}\ a_{ij} \vee \overset{\infty}{\mathsf{G}}\ b_{ij})$. Ce résultat est très intéressant car la restriction sur la structure de Φ_0 est faible puisqu'elle correspond bien au type de propriétés requises pour la spécification. Dans le cas général, on dispose d'un algorithme en $O(|\,S\,|\,.2^{|\Phi_0|}.\,|\,f\,|)$ et le problème du model-checking est NP-complet.

Pour la logique $GFCTL$, si l'on suppose que toutes les formules d'équité sont de la forme $\bigvee_i \bigwedge_j (\overset{\infty}{\mathsf{F}}\ a_{ij} \vee \overset{\infty}{\mathsf{G}}\ b_{ij})$, il existe un algorithme en $O(|\,S\,|,|\,f\,|^2)$. Ce résultat est aussi très intéressant et à comparer avec les algorithmes existants pour CTL^* ou PTL. Dans le cas général, on dispose d'un algorithme pour $GFCTL$ en $O(|\,S\,|\,.2^{|f|})$.

2.7 Comparaison de l'expressivité

Nous avons déjà mentionné que la comparaison des différentes logiques temporelles pouvait s'effectuer selon divers critères. Ainsi peut-on s'intéresser à leur *pouvoir de distinction*: on dit qu'une logique L distingue deux modèles M et M' donnés si il existe une formule f de L telle que $M \models f$ et $M' \not\models f$. Ce critère de comparaison est intéressant notamment en raison du lien qu'il établit avec les équivalences sémantiques entre modèles définies par ailleurs. Ce type d'approche permet ainsi d'obtenir des résultats d'adéquation ou de compatibilité entre les logiques temporelles et les équivalences sémantiques. On peut mentionner le fameux théorème d'adéquation de [HM80] entre la logique modale HML et la bisimulation forte dans le cadre des systèmes de transitions à branchement fini.

Cependant comparer les logiques selon leur pouvoir de distinction ne permet pas toujours une comparaison suffisamment fine. Ainsi la logique CTL^* et la logique $B(\mathsf{X})$ ont-elles le même pouvoir de distinction pour le cas des systèmes à branchement fini [BCG87, Sch90b, Pin92], mais il est clair que CTL^* a une expressivité beaucoup plus riche que $B(\mathsf{X})$. On souhaite donc disposer d'un critère plus précis pour comparer les logiques, on s'intéresse alors au pouvoir expressif (on parlera dans la suite d'*expressivité*) des logiques. La question posée est alors: *étant données deux logiques L_1 et L_2, les propriétés exprimables avec L_1 le sont-elles avec L_2 ?*

Définition 2.7.1 *Etant données deux logiques L_1 et L_2*

1. *L_1 n'est pas plus expressive que L_2 si pour toute formule f_1 de L_1, il existe une formule f_2 de L_2 telle que $f_1 \equiv f_2$. On note $L_1 \preceq_g L_2$.*

2. *L_1 n'est pas plus expressive initialement que L_2 si pour toute formule f_1 de L_1, il existe une formule f_2 de L_2 telle que $f_1 \equiv_i f_2$. On note $L_1 \preceq_i L_2$.*

Lorsque l'on a $L_1 \preceq_g L_2$ (resp. $L_1 \preceq_i L_2$), on dit aussi que L_1 se traduit (resp. se traduit initialement) dans L_2.

On définit $\prec_g$ et $\equiv$ (resp. $\prec_i$ et $\equiv_i$) l'ordre strict et l'équivalence associés à $\preceq_g$ (resp. $\preceq_i$):

Définition 2.7.2 *1. $L_1 \prec_g L_2$ ssi $L_1 \preceq_g L_2$ et $L_2 \not\preceq_g L_1$.*

2. *$L_1 \equiv L_2$ ssi $L_1 \preceq_g L_2$ et $L_2 \preceq_g L_1$.*

3. *$L_1 \prec_i L_2$ ssi $L_1 \preceq_i L_2$ et $L_2 \not\preceq_i L_1$.*

4. *$L_1 \equiv_i L_2$ ssi $L_1 \preceq_i L_2$ et $L_2 \preceq_i L_1$.*

On note $L_1 \bowtie L_2$, lorsque l'on a $L_1 \not\preceq_g L_2$ et $L_2 \not\preceq_g L_1$. De même, on note $L_1 \bowtie_i L_2$, lorsque l'on a $L_1 \not\preceq_i L_2$ et $L_2 \not\preceq_i L_1$. Lorsqu'on a $L_1 \bowtie L_2$ (resp. $L_1 \bowtie_i L_2$), on dit que les expressivités sont incomparables (resp. incomparables initialement).

Exemple 2.7.3 *A la section 2.4, nous avons vu que l'on pouvait d'une part exprimer les opérateurs U et X avec l'opérateur U_s (et réciproquement exprimer U_s avec U et X) et d'autre part exprimer les opérateurs S et X^{-1} avec S_s (et réciproquement exprimer S_s avec S et X^{-1}).*

Il en résulte d'une part: $PTL \preceq_g L(\mathsf{U}_s)$ *et* $L(\mathsf{U}_s) \preceq_g PTL$*. Et d'autre part, on a:* $PPTL \preceq_g L(\mathsf{U}_s, \mathsf{S}_s)$ *et* $L(\mathsf{U}_s, \mathsf{S}_s) \preceq_g PPTL$*. On en déduit donc:*

$$PTL \equiv L(\mathsf{U}_s) \qquad\qquad PPTL \equiv L(\mathsf{U}_s, \mathsf{S}_s)$$

Le lien entre $\equiv_i$ et $\equiv$ entraîne la proposition suivante:

Proposition 2.7.4 $L_1 \preceq_g L_2 \quad \Rightarrow \quad L_1 \preceq_i L_2$.

De plus, on a trivialement:

Proposition 2.7.5 $L_1 \subseteq L_2 \quad \Rightarrow \quad L_1 \preceq_g L_2$.

Pour que la comparaison de l'expressivité de deux logiques soit pertinente, il est nécessaire que leurs formules s'interprètent sur des modèles équivalents. Ainsi le problème se pose-t-il lorsque l'on compare les logiques pour le temps linéaire avec celles pour le temps arborescent (voir par exemple [CD89, Gra83, Eme83]) : il faut se donner un cadre commun pour l'interprétation des formules. Pour cela on a étendu la notion $\pi \models f_L$ utilisée pour les logiques du temps linéaire à la notion $S \models f_L$ du temps arborescent de la manière suivante: $S \models f_L$ ssi $\forall \pi \in S, \pi \models f_L$. Cette définition revient, dans notre cadre, à comparer les formules du temps arborescent à des formules $\mathsf{A}\ f_L$ où f_L est une formule du temps linéaire.

La définition de $S \models f_L$ est aussi motivée par l'idée qu'en général, une spécification avec une logique du temps linéaire est un ensemble de formules qui doivent être vérifiées par **toutes** les exécutions du système.

Pour représenter les logiques composées des formules $\mathsf{A}\ f_L$ où f_L est une formule d'une logique $L(C_1, \ldots, C_n)$ de temps linéaire, on utilise la notation suivante: $B(L(C_1, \ldots, C_n))$.

Un problème de même nature se pose lorsque l'on cherche à comparer des logiques munies d'opérateurs du passé avec des logiques pur-futur. En effet, aucune logique avec passé L_p ne pourra se traduire globalement dans une logique pur-futur L_f puisque L_p pourra faire référence aux états antérieurs à l'état courant. Dans de tels cas la notion de traduction globale a peu d'intérêt, on utilisera la notion de traduction initiale. La question posée sera alors: *étant donnée une formule de* L_p*, existe-t-il une formule de* L_f *équivalente initialement ?* Comme nous l'avons vu précédemment, cette notion de traduction initiale est tout à fait adaptée lorsqu'on considère la spécification des systèmes réactifs puisqu'une spécification est souvent définies à partir d'un état initial [MP89].

2.8 Résultats sur l'expressivité des fragments de $PCTL^*$

Dans cette section, nous allons rappeler les principaux résultats concernant l'expressivité des logiques temporelles que nous avons présentées précédemment. Dans un premier temps, nous donnerons les résultats concernant les logiques pur-futur. Ensuite nous rappelerons les résultats sur l'expressivité des logiques avec passé.

2.8.1 les logiques pur-futur

En ce qui concerne les logiques du temps linéaire, nous avons le résultat suivant qui motive l'introduction de l'opérateur *Until*:

Théorème 2.8.1 *[Kam68]*

$$LTL \prec_g PTL$$

Preuve En fait le théorème 3 du chapitre IV de [Kam68] énonce que l'opérateur U permet de construire des formules qui sont inexprimables avec la logique $LTL + \mathsf{F}_s^{-1} + \mathsf{X}^{-1}$. Pour montrer ce résultat, Kamp utilise le langage du premier ordre sur les modèles linéaires mentionné précédemment.

On peut aussi utiliser les modèles linéaires présentés par [EH85] pour montrer que $\mathsf{E}\, a\ \mathsf{U}\ b$ est inexprimable avec UB. Dans ce cas, on peut montrer que les deux modèles vérifient les mêmes formules de LTL, alors que l'un vérifie $a\,\mathsf{U}\,b$ et l'autre non. On en déduit qu'il n'existe pas de formule de LTL équivalente à $a\,\mathsf{U}\,b$. □

Pour les logiques du temps arborescent, nous avons la hiérarchie suivante:

Théorème 2.8.2 *[EH85, EH86, ES89]*

$$UB^- \prec_g UB \prec_g UB^+ \prec_g CTL \equiv CTL^+ \prec_g ECTL \prec_g ECTL^+ \prec_g CTL^* \tag{2.4}$$

Preuve D'après la définition syntaxique de ces différentes logiques, il est clair que:

$$\begin{array}{l} UB^- \preceq_g UB \preceq_g UB^+ \\ UB \preceq_g CTL \preceq_g CTL^+ \preceq_g ECTL^+ \\ CTL \preceq_g ECTL \preceq_g ECTL^+ \preceq_g CTL^* \end{array}$$

Pour établir 2.4, il reste à montrer que pour chaque résultat $L_1 \prec_g L_2$, il existe une formule de L_2 inexprimable avec L_1. Ces résultats sont récapitulés dans le tableau ci-dessous [ES89]:

Formule	exprimable en	non exprimable en	voir
$\mathsf{AF}\, a$	UB	UB^-	[EH85]
$\mathsf{E}\,(\mathsf{F}\, a \wedge \mathsf{G}\, b)$	UB^+	UB	[EH85]
$\mathsf{E}\,(a\, \mathsf{U}\, b)$	CTL	UB^+	[EH85]
$\mathsf{E}\, \overset{\infty}{\mathsf{F}}\, a$	$ECTL$	CTL	[EH86]
$\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\, a \wedge \overset{\infty}{\mathsf{F}}\, b)$	$ECTL^+$	$ECTL$	[EH86]
$\mathsf{EG}\,(a \vee \mathsf{X}\, a)$	CTL^*	$ECTL^+$	[EH86]

Le résultat $CTL \equiv CTL^+$ est démontré dans [EH85]. Pour avoir une idée de cette preuve [6], il suffit de constater que la formule $\mathsf{E}\,(a\, \mathsf{U}\, b \wedge c\, \mathsf{U}\, d)$ peut se traduire en CTL de la manière suivante: supposons $\pi, n \models \mathsf{E}\,(a\, \mathsf{U}\, b \ \wedge \ c\, \mathsf{U}\, d)$, alors l'état vérifiant b est soit avant celui vérifiant d, soit après, et on en déduit alors:

$$\mathsf{E}\,(a\, \mathsf{U}\, b \wedge c\, \mathsf{U}\, d) \equiv \begin{array}{l} \mathsf{E}\left((a \wedge c)\, \mathsf{U} \left(b \wedge \mathsf{E}\, c\ \mathsf{U}\ d\right)\right) \\ \vee \\ \mathsf{E}\left((a \wedge c)\, \mathsf{U} \left(d \wedge \mathsf{E}\, a\ \mathsf{U}\ b\right)\right) \end{array}$$

□

Ainsi pour toute formule f de CTL^+, il existe une formule f' de CTL équivalente à f et telle que $\mid f' \mid \leq 2^{|f|.log|f|}$ où $\mid f \mid$ représente le nombre de symboles dans f [EH85].

Dans [EH86], Emerson et Halpern montrent (c'est l'objet de leur théorème 6) que la formule de CTL^*, $\mathsf{E}\,(a\, \mathsf{U}\, b \vee a'\, \mathsf{U}\, b')\, \mathsf{U}\, r$, est inexprimable en $ECTL^+$. Nous allons montrer un résultat assez proche de celui-ci que nous utiliserons au chapitre 4, de plus cela permettra de présenter une méthode de preuve utilisée dans [EH86]. Nous allons montrer que la formule $\mathsf{E}\,(s \vee a\, \mathsf{U}\, b)\, \mathsf{U}\, r$ est inexprimable avec $ECTL^+$:

Lemme 2.8.3 *La formule $\mathsf{E}\,(s \vee a\, \mathsf{U} b)\, \mathsf{U} r$ n'est pas exprimable en $ECTL^+$.*

Preuve Considérons les deux familles M_i et N_i de modèles décrits par la figure 2.1. Nous avons, pour tout i: $\alpha_i \models \mathsf{E}\,(s \vee a\, \mathsf{U}\, b)\, \mathsf{U}\, r$ et $a_i \not\models \mathsf{E}\,(s \vee a\, \mathsf{U}\, b)\, \mathsf{U}\, r$. Nous allons montrer que pour toute formule f de $ECTL^+$ il existe i tel que $M_i, \alpha_i \models f$ et $N_i, a_i \not\models f$. Pour cela, nous allons montrer que d'une part α_j et a_j sont indistinguables par toute formule de CTL de taille inférieure à j, et d'autre part que pour toute formule f de $ECTL^+$, il existe une formule $\tilde{f}$ de CTL telle que $s \models f \Leftrightarrow \tilde{f}$ pour tout état s de M_i ou N_i $(\forall i)$.

[6] La preuve du théorème 4.4.14 présente en détail cette traduction

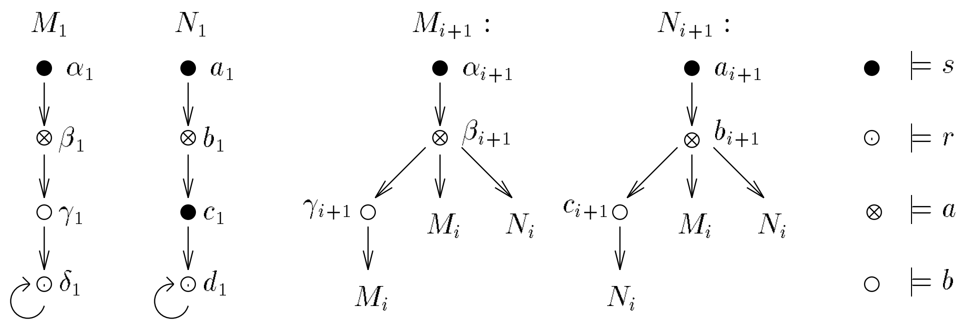


Figure 2.1: Les modèles M_i et N_i

1/ Indistinguabilité de M_i et N_i par CTL

Nous allons montrer que:

$$\forall f \in CTL \;\; \forall i > \mid f \mid \quad \begin{array}{lcl} M_i, \alpha_i \models f & \Leftrightarrow & N_i, a_i \models f \\ M_i, \beta_i \models f & \Leftrightarrow & N_i, b_i \models f \\ M_i, \gamma_i \models f & \Leftrightarrow & N_i, c_i \models f \end{array}$$

La preuve se fait par induction sur la taille de f, notée $\mid f \mid$ et représentant le nombre de symboles dans f:

- $\mid f \mid = 1$: Les états α_i et a_i vérifient les mêmes propositions atomiques (idem pour β_i et b_i, et γ_i et c_i).
- $\mid f \mid = n+1$. On distingue plusieurs cas: $f = \mathsf{EX}\, g$, $\mathsf{E}\, g \;\mathsf{U}\; h$, $\mathsf{A}\, g \;\mathsf{U}\; h$. Les cas $f = g \wedge h$ et $f = \neg g$ sont triviaux.
  - $M_i, \alpha_i \models \mathsf{EX}\, g$: Alors $M_i, \beta_i \models g$ et donc, par h.i. $N_i, b_i \models g$ et $N_i, a_i \models EXg$. On obtient de la même manière la réciproque.
  - $M_i, \alpha_i \models \mathsf{E}\, g \;\mathsf{U}\; h$: On a soit (1) $M_i, \alpha_i \models h$, soit (2) $M_i, \alpha_i \models g$ et: soit (2-a) $M_i, \beta_i \models h$, soit (2-b) $M_i, \beta_i \models g$ et (2-b-1) $M_i, \gamma_i \models h$ ou (2-b-2) $M_{i-1}, \alpha_{i-1} \models \mathsf{E}\, g \;\mathsf{U}\; h$ ou (2-b-3) $N_{i-1}, a_{i-1} \models \mathsf{E}\, g \;\mathsf{U}\; h$.

    Pour les cas (1), (2-a) et (2-b-1), l'hypothèse d'induction nous permet de déduire que $N_i, a_i \models \mathsf{E}\, g \;\mathsf{U}\; h$. Pour les cas (2-b-2) et (2-b-3), on déduit par h.i. $N_i, a_i \models g$ et $N_i, b_i \models g$, ce qui permet d'obtenir $N_i, a_i \models \mathsf{E} g \,\mathsf{U}\, h$ car soit $M_{i-1}, \alpha_{i-1} \models \mathsf{E}\, g \;\mathsf{U}\; h$, soit $N_{i-1}, a_{i-1} \models \mathsf{E}\, g \;\mathsf{U}\; h$.
  - $M_i, \alpha_i \models \mathsf{A} g \mathsf{U} h$. On a soit $M_i, \alpha_i \models h$, soit $M_i, \alpha_i \models g$ et $M_i, \beta_i \models \mathsf{A} g \mathsf{U} h$. Dans ce dernier cas: soit $M_i, \beta_i \models h$, soit $M_{i-1}, \alpha_{i-1} \models \mathsf{A}\, g \;\mathsf{U}\; h$ et $N_{i-1}, a_{i-1} \models \mathsf{A} g \mathsf{U} h$ et $M_i, \gamma_i \models g \vee h$. On obtient par h.i. $N_i, a_i \models \mathsf{A} g \mathsf{U} h$. On obtient de manière identique la réciproque.

On procède de manière identique pour β_i et b_i, γ_i et c_i.

2/ **Montrons que** $\forall f \in ECTL^+, \exists \tilde{f} \in CTL$ **tq** $\forall i, \forall s \in M_i, \quad M_i, s \models f \quad \Leftrightarrow \quad M_i, s \models \tilde{f}$ **et** $\forall s \in N_i, \; N_i, s \models f \quad \Leftrightarrow \quad N_i, s \models \tilde{f}$

Soit $f \in ECTL^+$ de la forme $f = \mathsf{E}\left(p \wedge \overset{\infty}{\mathsf{F}} q_1 \wedge \ldots \wedge \overset{\infty}{\mathsf{F}} q_n \wedge \overset{\infty}{\mathsf{G}} q'\right)$, avec $\mathsf{E} p, q_i, q' \in CTL^+$. Alors, les seuls chemins infinis dans M_i et N_i étant ceux bouclant sur un état, on a: $\forall s \in M_i$, $M_i, s \models f$ ssi $M_i, s \models \mathsf{E}\left[p \wedge \mathsf{F}\left(\mathsf{A}\,\mathsf{G}\,(q_1 \wedge \ldots \wedge q_n \wedge q')\right)\right]$. On a de même pour N_i. La transformation précédente permet (en utilisant la mise sous forme normale disjonctive et la distributivité du E pour la disjonction) de construire [7] pour toute formule f de $ECTL^+$, une formule f' de CTL^+ telle que:

$$\begin{array}{lll} M_i, s \models f & \text{ssi} & M_i, s \models f' \\ N_i, s \models f & \text{ssi} & N_i, s \models f' \end{array}$$

On peut alors utiliser la méthode de [EH85] pour obtenir une formule $\tilde{f}$ de CTL équivalente à f'.

Si il existait une formule f de $ECTL^+$ équivalente à $\mathsf{E}\,(s \vee a\,\mathsf{U}\,b)\,\mathsf{U}\,r$, alors il existerait une formule $\tilde{f}$ de CTL telle que $\forall i$, $M_i, \alpha_i \models \tilde{f} \quad \Leftrightarrow \quad N_i, a_i \models \tilde{f}$, or ceci est en contradiction avec la première partie de cette preuve.

Nous pouvons donc déduire qu'aucune formule de $ECTL^+$ n'est équivalente à la formule $\mathsf{E}\,(s \vee a\,\mathsf{U}\,b)\,\mathsf{U}\,r$. $\square$

Nous allons présenter une autre preuve de ce type concernant les logiques $ECTL$ et $ECTL^+$. Nous savons que la formule de $ECTL^+$ $\mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \overset{\infty}{\mathsf{F}} b)$ est inexprimable en $ECTL$ [EH86], nous allons montrer qu'il en est de même pour $\mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \mathsf{G}\,b)$ qui correspond à la propriété d'équité *strong eventual fairness* $\mathsf{A}\,(\overset{\infty}{\mathsf{F}} a \Rightarrow \mathsf{F}\,b)$ définie dans [Lam80].

Lemme 2.8.4 *La formule de* $ECTL^+$, $\mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \mathsf{G}\,b)$, *est inexprimable en* $ECTL$.

Preuve Pour montrer que $\mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \mathsf{G}\,b)$ est inexprimable en $ECTL$, nous allons utiliser des modèles proches de ceux présentés dans [EH86] pour montrer que $\mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \overset{\infty}{\mathsf{F}} b)$ est inexprimable en $ECTL$. On considère donc les modèles M_1, M_2,…et N_1, N_2,…décrit par la figure 2.2.

Pour tout i, nous avons:

$$a_i \models \mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \mathsf{G}\,b) \;\text{ et }\; \alpha_i \not\models \mathsf{E}\,(\overset{\infty}{\mathsf{F}} a \wedge \mathsf{G}\,b)$$

Nous allons montrer que les états a_i et α_i sont indistinguables par les formules de $ECTL$ de taille au plus i. Pour cela, nous allons d'abord prouver par induction

[7] la preuve se fait facilement par induction sur la hauteur de quantificateur E

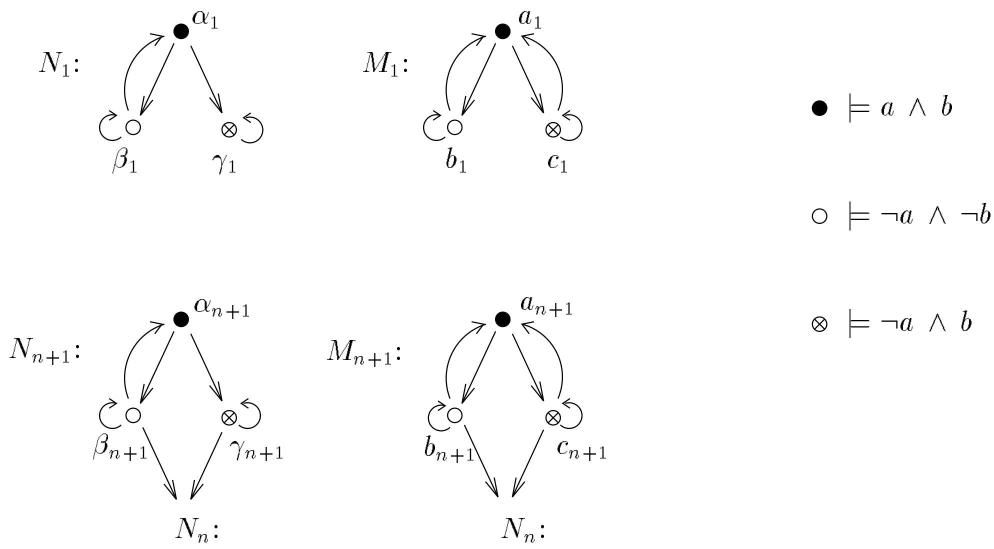


Figure 2.2: $a_i, \alpha_i \models a \wedge b, \quad b_i, \beta_i \models \neg a \wedge \neg b, \quad c_i, \gamma_i \models \neg a \wedge b.$

les trois propositions suivantes qui énoncent que toute formule de $ECTL$ de taille i qui est vraie pour un état α_j (resp. β_j, γ_j) avec $j \geq i$, est vraie pour tous les états α_k (resp. β_k, γ_k) avec $k \geq i$:

$$\begin{array}{lcll}
(\exists j > \mid f \mid,\ \alpha_j \models f) & \text{ssi} & (\forall k \geq \mid f \mid,\ \alpha_k \models f) & (a) \\
(\exists j > \mid f \mid,\ \beta_j \models f) & \text{ssi} & (\forall k \geq \mid f \mid,\ \beta_k \models f) & (b) \\
(\exists j > \mid f \mid,\ \gamma_j \models f) & \text{ssi} & (\forall k \geq \mid f \mid,\ \gamma_k \models f) & (c)
\end{array}$$

Pour établir ces résultats, il suffit de montrer les propriétés suivantes:

$$\begin{array}{rcl}
\alpha_i \models f & \Rightarrow & \alpha_{i+1} \models f \\
\alpha_i \models f \text{ et } i > \mid f \mid & \Rightarrow & \alpha_{i-1} \models f \\
\beta_i \models f & \Rightarrow & \beta_{i+1} \models f \\
\beta_i \models f \text{ et } i > \mid f \mid & \Rightarrow & \beta_{i-1} \models f \\
\gamma_i \models f & \Rightarrow & \gamma_{i+1} \models f \\
\gamma_i \models f \text{ et } i > \mid f \mid & \Rightarrow & \gamma_{i-1} \models f
\end{array}$$

Cette première partie se fait facilement par induction sur f et sur i.

Une fois prouvés (a), (b) et (c), il ne reste plus qu'à montrer que, pour toute formule f de $ECTL$ telle que $\mid f \mid \leq i$, on a:

$$\begin{array}{lcl}
\alpha_i \models f & \text{ssi} & a_i \models f, \\
\beta_i \models f & \text{ssi} & b_i \models f, \\
\gamma_i \models f & \text{ssi} & c_i \models f.
\end{array}$$

On montre ce résultat par induction sur la taille de f:

$\star$ $|f| = 0$: évident.

$\star$ $|f| = i+1$: On veut montrer que $\forall j \geq i+1$, $\alpha_j \models f \Leftrightarrow a_j \models f$, $\beta_j \models f \Leftrightarrow b_j \models f$ et $\gamma_j \models f \Leftrightarrow c_j \models f$. On distingue plusieurs cas:

- $f = \mathsf{EX}\, g$: étant données les structures de M_n et N_n, le seul cas posant problème est celui où $c_j \models \mathsf{EX}\, g$ et $a_j \models g$. Alors par h.i. on sait que $\alpha_j \models g$ et on utilise (c) pour obtenir $\alpha_{j-1} \models g$ (car $|g| \geq j-1$) et donc $\gamma_j \models \mathsf{EX}\, g$.

- $f = \mathsf{E}\, g\ \mathsf{U}\ h$: là encore le seul cas non direct est celui où $c_j \models \mathsf{E}\, g\ \mathsf{U}\ h$ avec $c_j \models g$ et soit $a_j \models h$, soit $a_j \models g$ et $b_j \models h$. Dans ce cas, par h.i. on obtient $\gamma_j \models g$ et $\alpha_j \models h$ (resp. $\alpha_j \models g$ et $\beta_j \models h$), et (a) (resp. (b) et (c)) nous donne: $\alpha_{j-1} \models h$ (resp. $\alpha_{j-1} \models g$ et $\beta_{j-1} \models h$), et donc: $\gamma_j \models \mathsf{E}\, g\ \mathsf{U}\ h$.

- $f = \mathsf{A}\, f\ \mathsf{U}\ g$: on obtient facilement le résultat puisque $\beta_j \models \mathsf{A}\, f\ \mathsf{U}\ g$ ssi $\beta_j \models g$ du fait de l'existence de l'exécution $\beta_j \to \beta_j \to \ldots$ et il en est de même pour γ_j, b_j et c_j.

- $f = \mathsf{E}\, \overset{\infty}{\mathsf{F}}\ g$: le seul cas intéressant est celui où $c_j \models \mathsf{E}\, \overset{\infty}{\mathsf{F}}\ g$ avec le chemin $\pi = c_i \to a_i \to c_i \to a_i \ldots$ vérifiant $\overset{\infty}{\mathsf{F}}\ g$ (c'est en effet le seul chemin n'ayant pas de suffixe commun avec un chemin de N_j). Mais dans ce cas, soit $a_j \models g$ (et donc $\alpha_{j-1} \models g$) , soit $c_j \models g$ (et donc $\gamma_j \models g$) : on en déduit facilement que $\gamma_j \models \mathsf{E}\, \overset{\infty}{\mathsf{F}}\ g$.

Les états α_j et a_j sont donc indistinguables par toute formule de $ECTL$ de taille inférieure à j. On en déduit que la formule $\mathsf{E}\, (\overset{\infty}{\mathsf{F}}\ a \wedge \mathsf{G}\, b)$ est inexprimable avec $ECTL$ $\square$

Ce résultat complète le résultat de [EH86] énonçant que la formule $\mathsf{E}\, (\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$ est inexprimable en $ECTL$: ils montrent bien la différence d'expressivité entre $ECTL$ et $ECTL^+$. Nous verrons au chapitre 4 que $ECTL^+$ est équivalente à la logique $ECTL$ dans laquelle on remplace l'opérateur $\mathsf{E}\, \overset{\infty}{\mathsf{F}}$ par une famille d'opérateurs $\mathsf{E}\, (\mathsf{G}\, h \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\ f_i)$ qui énoncent qu'il existe une exécution le long de laquelle la formule h est toujours vraie et les formules f_i sont vraies infiniment souvent.

Avant de présenter les résultats concernant les logiques avec passé, il faut mentionner les résultats entre logiques du temps arborescent et logiques du temps linéaire. A la section précédente, nous avons vu que comparer L_B du temps arborescent et L_L du temps linéaire, revenait à comparer, en réalité, L_B et $B(L_L)$. Nous avons les résultats suivants:

Théorème 2.8.5 *[EH86]*

$$B(L(\mathsf{F})) \prec_g B(L(\mathsf{U})) \prec_g B(L(\mathsf{X},\mathsf{U})) \prec_g CTL^*$$
$$B(L(\mathsf{F})) \prec_g ECTL^+$$

$$\begin{array}{ll} Et: & B(L(\mathsf{F})) \bowtie UB^- \\ & B(L(\mathsf{F})) \bowtie CTL \\ & B(L(\mathsf{X},\mathsf{U})) \bowtie UB^- \\ & B(L(\mathsf{X},\mathsf{U})) \bowtie ECTL^+ \end{array}$$

Preuve Le résultat $B(L(\mathsf{F})) \prec_g ECTL^+$ est le théorème 4 de [EH86].

La preuve de $B(L(\mathsf{F})) \not\preceq_g CTL$ est basée sur le fait que $\mathsf{A}\overset{\infty}{\mathsf{G}}\ a$ est inexprimable en CTL.

La preuve de $UB^- \not\preceq_g B(L(\mathsf{X},\mathsf{U}))$ est basée sur le fait que la formule $\mathsf{EF}\,a$ est inexprimable avec $B(L(\mathsf{X},\mathsf{U}))$.

La preuve $B(L(\mathsf{U},\mathsf{X})) \not\preceq_g ECTL^+$ est basée sur le fait que $\mathsf{A}\neg\Big((a\mathsf{U}b \vee a'\mathsf{U}b')\mathsf{U}c\Big)$ est inexprimable en $ECTL^+$. □

Les théorèmes que nous venons de présenter concernant la comparaison de l'expressivité des logiques du futur nous donnent la hiérarchie de la figure 2.3.

2.8.2 Les logiques avec passé

L'un des premiers résultats concernant les logiques avec passé a été présenté dans [Kam68] où il est démontré que la logique $L(\mathsf{U}_s, \mathsf{S}_s)$ (i.e. une logique équivalente à $PPTL$) est *expressive complete*, c'est à dire aussi expressive que le langage du premier ordre muni de l'égalité, du prédicat binaire $<$ et des prédicats unaires a, b... interprété sur des modèles linéaires. De plus, Kamp montre que la logique $L(\mathsf{F}_s, \mathsf{X}, \mathsf{F}_s^{-1}, \mathsf{X}^{-1})$ ne vérifie pas cette propriété. En conséquence, nous avons le résultat suivant:

Théorème 2.8.6 *[Kam68]* $L(\mathsf{F}_s, \mathsf{X}, \mathsf{F}_s^{-1}, \mathsf{X}^{-1}) \prec_g L(\mathsf{U}_s, \mathsf{S}_s)$

A la suite de [Kam68], il est démontré dans [GPSS80] que le fragment pur-futur $L(\mathsf{U}_s)$ est *expressive complete* initialement. En conséquence, on a:

Théorème 2.8.7 *[GPSS80]*

$$L(\mathsf{U}_s, \mathsf{S}_s) \equiv_i L(\mathsf{U}_s)$$

De plus, Gabbay et al. montrent le théorème de séparation suivant:

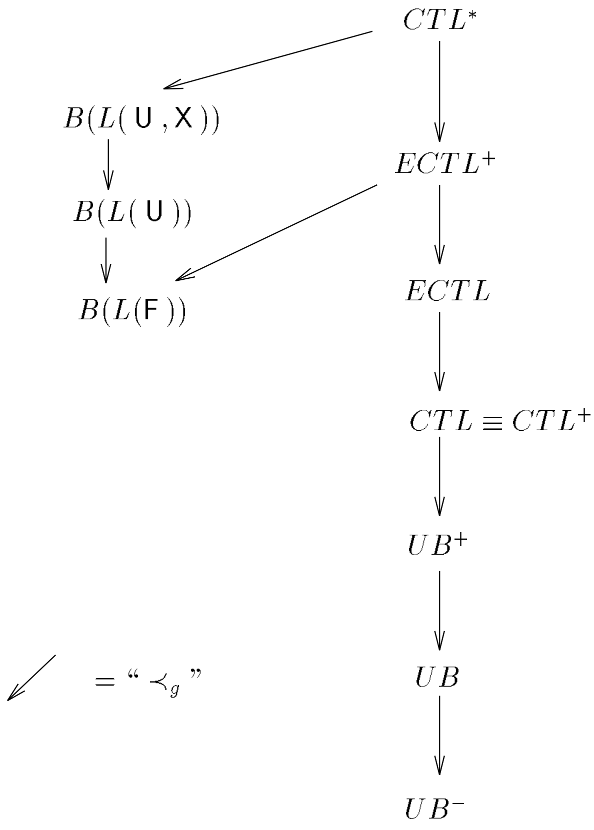


Figure 2.3: Expressivités des fragments de CTL^* [EH86]

Théorème 2.8.8 *Théorème de séparation [GPSS80]*

Toute formule de $L(\mathsf{U}_s, \mathsf{S}_s)$ est équivalente à une combinaison booléenne de formules pur-passé et pur-futur.

Nous reviendrons dans le chapitre 4 sur les techniques de preuves utilisées pour ce résultat et qui sont développées dans [Gab87]. Une conséquence immédiate du corollaire précédent est $PPTL \equiv_i PTL$, cela provient du fait que $PPTL \equiv L(\mathsf{U}_s, \mathsf{S}_s)$ et $PTL \equiv L(\mathsf{U}_s)$. Un résultat similaire (quant à la technique utilisée) concerne la logique $L(\mathsf{U}, \mathsf{S})$ c'est à dire $PTL \backslash \mathsf{X}$:

Théorème 2.8.9 *[MMKR93] $L(\mathsf{U}, \mathsf{S}) \equiv_i L(\mathsf{U})$*

Le théorème de séparation de Gabbay est aussi à la base d'un résultat présenté dans [HT87] concernant la logique $PCTL^*_{HT}$:

Théorème 2.8.10 *[HT87] $PCTL^*_{HT} \equiv_i CTL^*$*

Nous verrons que la preuve de ce théorème s'adapte facilement à notre définition de $PCTL^*$. Enfin, un dernier résultat concernant l'expressivité des logiques avec passé est à mentionner. Il a été présenté dans [MP90] et [Zuc86]:

Théorème 2.8.11 *[MP90, Zuc86]*

Toute formule de $PPTL$ est équivalente initialement à une formule de la forme:

$$\bigwedge_{i=1..n} \left(\mathsf{G}\,\mathsf{F}\,p_i \ \vee\ \mathsf{F}\,\mathsf{G}\,q_i \right)$$

où les p_i et les q_i sont des formules pur-passé de $PPTL$.

Ce dernier résultat a été développé dans le cadre de l'étude de classes de propriétés temporelles pour la spécification des systèmes réactifs. Une conséquence de ce théorème est donc:$L(\mathsf{F}\,,\ \mathsf{S}\,,\mathsf{X}^{-1}\,) \equiv_i PPTL$.

2.9 Conclusion

Après avoir rappelé les différents résultats concernant l'expressivité des fragments de $PCTL^*$, on peut noter que si on trouve dans la littérature une hiérarchie assez fine pour les logiques du futur, il n'en va pas de même pour les logiques avec passé. Ce commentaire vaut tout particulièrement pour les logiques du temps arborescent, pour lesquelles le seul résultat mentionnant une logique avec passé énonce que $PCTL^*_{HT}$ se traduit initialement dans CTL^*. Ainsi, il n'existe dans ce cadre aucun résultat de la forme "$CTL+$passé se traduit initialement dans CTL". La situation est identique pour $ECTL$ ou $ECTL^+$. Pourtant ces logiques sont particulièrement bien adaptées au *model-checking* car elles disposent d'algorithmes très efficaces, à la différence de CTL^* ou des logiques de temps linéaire.

Dans cette situation, si l'on dispose d'une spécification en $CTL +$ passé, on doit la traduire en CTL^* pour la phase de vérification, cela pose évidemment des problèmes d'efficacité.

Au chapitre suivant, nous allons montrer à partir, d'un exemple, l'utilité des opérateurs du passé pour la construction des spécifications. Ensuite, nous aborderons au cours des chapitres suivants la définition de logiques munies d'opérateurs du passé, pouvant se traduire dans des logiques du futur,comme CTL ou $ECTL^+$, pour lesquelles le model-checking est efficace.

Chapitre 3

Spécification en CTL + Passé: l'exemple d'un ascenseur

3.1 Introduction

Au chapitre précédent, nous avons présenté la logique $PCTL^*$ ainsi que les différents fragments classiquement considérés dans la littérature. Avant de nous intéresser à la comparaison de l'expressivité de différentes logiques avec passé, nous allons montrer, à partir d'un exemple, en quoi les opérateurs du passé sont utiles pour la spécification en permettant une expression naturelle et concise des propriétés.

L'exemple que nous avons choisi est un ascenseur. Cet exemple est suffisamment riche (cf. [Bar85, Hal87]) pour nécessiter l'expression de propriétés complexes. Il existe bien évidemment de nombreuses spécifications différentes pour un tel système, nous en proposerons une et nous indiquerons quelques variantes possibles. Nous verrons qu'il est très facile d'adapter une telle spécification en raison du caractère compositionnel de la logique temporelle: une spécification est constituée par un ensemble de formules qui correspondent aux propriétés que le système doit vérifier, changer une de ces propriétés revient à modifier la ou les formules qui lui sont associée(s) (et non toute la spécification).

Cette spécification est abstraite: elle ne suppose aucun choix d'implémentation particulier, elle ne considère qu'un ensemble de propositions atomiques qui correspondent à des paramètres observables (porte ouverte, bouton enfoncé, etc). Cette spécification est complète au sens où elle décrit l'ensemble des propriétés temporelles que doit vérifier notre système [1].

Nous allons tout d'abord donner une description informelle de notre ascenseur. Ensuite, à partir de cette description, nous proposerons d'une part une spécification avec la logique $CTL+$ Passé, et d'autre part une spécification avec la logique CTL. Nous pourrons alors comparer la lisibilité de ces deux spécifications et montrer l'intérêt des opérateurs du passé. Nous avons choisi la logique CTL en raison de l'efficacité des algorithmes de model-checking pour cette logique.

3.2 Description du système

L'ascenseur que nous allons étudier dessert n étages. La cabine comporte n boutons qui permettent de choisir la (ou les) destination(s). A chaque bouton correspond un voyant lumineux qui témoigne, lorsqu'il est allumé, de l'existence d'une requête pour cet étage. De plus, à chaque étage se trouvent un bouton et un voyant lumineux: le bouton permet d'appeler l'ascenseur et le voyant témoigne (lorsqu'il est allumé) de l'existence d'une telle requête. Lorsque la cabine s'arrête à un étage, la porte s'ouvre alors automatiquement.

On peut donner une spécification informelle de cet ascenseur:

1. Quand un bouton est enfoncé, le voyant correspondant s'allume. Quand la requête correspondante est satisfaite, le voyant s'éteint.

2. A chaque étage, la porte n'est jamais ouverte si la cabine n'est pas là.

3. Chaque requête doit être satisfaite "un jour".

4. L'ascenseur ne dessert que les étages pour lesquels existe une requête.

5. Lorsqu'il n'y a pas de requête, la cabine reste à l'étage où elle se trouve.

6. Lorsque la cabine se déplace, elle doit s'arrêter aux étages par lesquels elle passe si une requête les concerne.

7. L'ascenseur doit se déplacer le plus directement possible (lors d'un déplacement de l'étage i vers l'étage j, la cabine ne doit passer que par des étages situés entre i et j, et elle ne doit pas changer de direction au cours de ce déplacement)

[1] Les autres propriétés que l'on pourrait exiger ne sont pas simplement temporelles: elles concernent, par exemple, les aspects temps-réel, la modularité, ...

8. Lorsqu'il existe plusieurs requêtes, l'ascenseur doit traiter prioritairement celle(s) permettant de continuer dans la même direction que le dernier déplacement.

Nous allons utiliser un ensemble de propositions atomiques qui fourniront une description du système à un moment donné. La spécification consistera en un ensemble de formules de logique temporelle qui représenteront les propriétés que doit satisfaire le modèle décrivant le comportement de l'ascenseur. Ce modèle a donc la forme d'une structure de Kripke dont les états sont étiquetés par cet ensemble de propositions atomiques.

Nous allons considérer les propositions atomiques suivantes:

- OpenD_i $(i \in 1, \ldots, n)$ décrit l'état de la porte de l'étage i: un état vérifie OpenD_i ssi la porte de l'étage i est ouverte.
- Call_i $(i \in 1, \ldots, n)$ décrit l'état du bouton d'appel de l'étage i: un état vérifie Call_i ssi le bouton est enfoncé.
- $\mathit{C\text{-}Light}_i$ $(i \in 1, \ldots, n)$ décrit l'état du voyant de l'étage i: un état vérifie $\mathit{C\text{-}Light}_i$ ssi le voyant i est allumé.
- Send_i $(i \in 1, \ldots, n)$ décrit la position du i^{eme} bouton de la cabine: un état vérifie Send_i ssi le bouton est enfoncé.
- $\mathit{S\text{-}Light}_i$ $(i \in 1, \ldots, n)$ décrit l'état du voyant associé au i^{eme} bouton de la cabine: un état vérifie $\mathit{S\text{-}Light}_i$ ssi le voyant est allumé.
- at_i $(i \in 1, \ldots, n)$ décrit la position de la cabine: un état vérifie at_i ssi la cabine se trouve à l'étage i.

Nous aurions pu choisir d'autres propositions atomiques. Notre choix, proche de celui de [Bar85], est motivé par le fait que l'ensemble des propositions que nous considérons sont des propositions dont la valeur de vérité dans un état donné est immédiatement accessible.

Nous allons maintenant présenter une spécification de cet ascenseur avec la logique CTL étendue avec des opérateurs du passé.

3.3 Une spécification avec passé

Nous allons tout d'abord étudier les propriétés qui concernent le lien entre les boutons et les voyants lumineux, l'ouverture des portes et le déplacement continu de l'ascenseur. Il s'agit de propriétés de *safety* [Pnu77, OL82, MP92]. Intuitivement cette classe de propriétés est utilisée pour spécifier que “rien de mal” ne peut arriver. Par opposition on définit les propriétés de *liveness* qui permettent d'exprimer qu'une “propriété désirée” sera un jour satisfaite.

3.3.1 Voyants lumineux et boutons

∘ Nous allons d'abord spécifier que *les voyants (aux étages ou dans la cabine) s'allument chaque fois que c'est nécessaire*: lorsqu'on appuie sur un bouton, alors soit l'ascenseur est là et la porte est ouverte (et la requête est alors immédiatement satisfaite), soit le voyant correspondant s'allume immédiatement. La formule suivante énonce cette propriété pour les boutons d'appel aux étages:

$$\mathsf{AG} \bigwedge_{i=1..n} \left[\mathit{Call}_i \;\Rightarrow\; \left(\mathit{servicing}_i \;\vee\; \mathit{C\text{-}Light}_i\right)\right] \tag{3.5}$$

Avec: $\mathit{servicing}_i \stackrel{\text{def}}{=} \mathit{at}_i \;\wedge\; \mathit{OpenD}_i$. Un état vérifie $\mathit{servicing}_i$, lorsque la cabine dessert l'étage i (la cabine se trouve à cet étage avec la porte ouverte). Pour les boutons de la cabine, nous avons la formule suivante:

$$\mathsf{AG} \bigwedge_{i} \left[\mathit{Send}_i \;\Rightarrow\; \left(\mathit{servicing}_i \;\vee\; \mathit{S\text{-}Light}_i\right)\right] \tag{3.6}$$

Dans la conjonction $\bigwedge_i$, i prend les valeurs de 1 à n. Dans la suite, on omettra ce domaine pour alléger les formules. En revanche, on précisera le domaine des indices lorsqu'il sera différent de $1, \ldots, n$.

∘ Pour exprimer que *les voyants doivent rester allumés tant que la requête n'est pas satisfaite*, nous utilisons la formule suivante:

$$\mathsf{AG} \bigwedge_{i} \left[\mathit{C\text{-}Light}_i \;\Rightarrow\; \mathsf{A}\,(\mathit{C\text{-}Light}_i)\,\mathsf{W}\;(\mathit{servicing}_i)\right] \tag{3.7}$$

L'opérateur $\mathsf{A\_W}$ [2] permet de spécifier que, pour toutes les exécutions possibles, soit le voyant i reste toujours allumé, soit il reste allumé jusqu'à ce que la requête associée soit satisfaite. De même, on a:

$$\mathsf{AG} \bigwedge_{i} \left[\mathit{S\text{-}Light}_i \;\Rightarrow\; \mathsf{A}\,(\mathit{S\text{-}Light}_i)\,\mathsf{W}\;(\mathit{servicing}_i)\right] \tag{3.8}$$

∘ Il faut maintenant spécifier que *les voyants ne s'allument* ***que*** *si c'est nécessaire*, c'est à dire si le bouton correspondant a été actionné. Intuitivement on serait tenté d'utiliser l'opérateur F^{-1} (*il y eu dans le passé*) de la manière suivante: $\mathsf{AG} \bigwedge_i \left(\mathit{C\text{-}Light}_i \;\Rightarrow\; \mathsf{F}^{-1}\mathit{Call}_i\right)$. Pourtant cette formule n'est pas satisfaisante car il suffit, dans le contexte de cette formule, que le bouton ait été enfoncé une fois pour que le voyant puisse s'allumer n'importe quand ensuite. Il faut donc spécifier

[2] $\mathsf{A}\,a\,\mathsf{W}\;b \equiv \neg\mathsf{E}\,(\neg b)\;\mathsf{U}\;(\neg a \;\wedge\; \neg b)$

que si un voyant est allumé, alors il l'est depuis que le bouton correspondant a été actionné:

$$\mathsf{AG} \bigwedge_i \Big[\textit{C-Light}_i \;\Rightarrow\; \big(\textit{C-Light}_i \;\mathsf{S}\; \textit{Call}_i \big) \Big] \tag{3.9}$$

$$\mathsf{AG} \bigwedge_i \Big[\textit{S-Light}_i \;\Rightarrow\; \big(\textit{S-Light}_i \;\mathsf{S}\; \textit{Send}_i \big) \Big] \tag{3.10}$$

Nous verrons au chapitre 5 comment l'opérateur *Now* associé à l'opérateur F^{-1} permet d'exprimer une formule équivalente.

∘ Enfin, nous devons spécifier que *les voyants s'éteignent quand la requête est satisfaite*:

$$\mathsf{AG} \bigwedge_i \Big[\textit{servicing}_i \;\Rightarrow\; \neg \textit{C-Light}_i \,\wedge\, \neg \textit{S-Light}_i \Big] \tag{3.11}$$

Les formules précédentes spécifient le comportement exigé des voyants en réponse aux boutons et à la cabine. Nous pouvons aussi exprimer ce comportement d'une manière plus concise en remarquant que nous avions défini informellement chaque voyant comme le témoin de l'existence d'une requête non encore satisfaite. Ainsi peut-on dire qu'un voyant est allumé si et seulement si la requête qui lui est associée est non encore satisfaite. On peut donc utiliser les formules suivantes:

$$\mathsf{AG} \bigwedge_i \Big[\textit{C-Light}_i \;\Leftrightarrow\; \big(\neg \textit{servicing}_i \big) \;\mathsf{S}\; \big(\textit{Call}_i \,\wedge\, \neg \textit{servicing}_i \big) \Big] \tag{3.12}$$

$$\mathsf{AG} \bigwedge_i \Big[\textit{S-Light}_i \;\Leftrightarrow\; \big(\neg \textit{servicing}_i \big) \;\mathsf{S}\; \big(\textit{Send}_i \,\wedge\, \neg \textit{servicing}_i \big) \Big] \tag{3.13}$$

Ces formules expriment le fait qu'un voyant est allumé ssi il y eu dans le passé une requête (bouton enfoncé) et que depuis, cette requête n'a pas été satisfaite. On peut noter que la conjonction de ces deux formules est équivalente à la conjonction de (3.5),(3.6), (3.7), (3.8), (3.9), (3.10) et (3.11) et bien qu'elles aient une structure un peu plus compliquée, ces deux formules correspondent bien à la définition intuitive du lien entre voyant et bouton.

3.3.2 Les portes

∘ Nous devons spécifier que *si la porte de l'étage i est ouverte, alors la cabine se trouve à l'étage i*:

$$\mathsf{AG} \bigwedge_i \Big[\mathit{OpenD}_i \;\Rightarrow\; \mathit{at}_i \Big] \tag{3.14}$$

3.3.3 La cabine

∘ Pour exprimer que *la cabine ne peut se trouver qu'à un seul étage*, nous utilisons la formule suivante:

$$\mathsf{AG} \bigwedge_i \Big[\mathit{at}_i \;\Rightarrow\; \bigwedge_{j \neq i} \neg \mathit{at}_j \Big] \tag{3.15}$$

∘ Nous devons spécifier que *la cabine se déplace de manière continue*: lorsqu'elle se trouve à l'étage i, elle était précédemment à l'étage $i-1$ ou $i+1$ (excepté lorsqu'elle se trouve à l'étage 1 (resp. n) où elle ne peut venir que de l'étage 2 (resp. $n-1$)):

$$\begin{aligned} &\mathsf{AG} \bigwedge_{i=2,\ldots,n-1} \Big[\mathit{at}_i \;\Rightarrow\; \Big((\mathit{at}_i \;\vee\; \mathit{between\text{-}levels})\; \mathsf{S}\; (\mathit{at}_{i-1} \;\vee\; \mathit{at}_{i+1}) \Big) \Big] \\ &\wedge\; \mathsf{AG} \Big[\mathit{at}_1 \;\Rightarrow\; \Big((\mathit{at}_1 \;\vee\; \mathit{between\text{-}levels})\; \mathsf{S}\; \mathit{at}_2 \Big) \Big] \\ &\wedge\; \mathsf{AG} \Big[\mathit{at}_n \;\Rightarrow\; \Big((\mathit{at}_n \;\vee\; \mathit{between\text{-}levels})\; \mathsf{S}\; \mathit{at}_{n-1} \Big) \Big] \end{aligned} \tag{3.16}$$

Avec: $\mathit{between\text{-}levels} \stackrel{\text{def}}{=} \bigwedge_{i=1,\ldots,n} \neg \mathit{at}_i$. Si un état vérifie $\mathit{between\text{-}levels}$, alors la cabine se trouve entre deux étages. Lorsqu'un état vérifie at_i (i.e. la cabine est à l'étage i), la propriété $\mathit{at}_i \vee \mathit{between\text{-}levels}$ est vraie pour tous les états précédents depuis que la cabine a quitté un étage j: cet étage est donc le dernier étage visité par la cabine avant d'aller à l'étage i. Pour spécifier que la cabine a un déplacement continu, il suffit donc de dire que j est soit $i-1$ soit $i+1$.

3.3.4 Requêtes et comportement de la cabine

Nous venons de définir un ensemble de propriétés de *safety* concernant des propriétés de base. Nous allons maintenant nous intéresser à des propriétés plus complexes qui vont concerner le lien entre les requêtes et le déplacement de la cabine (choix prioritaire, optimisation ...). Tout d'abord nous avons la propriété de *liveness* suivante:

∘ *Toute requête doit être satisfaite.* Nous utilisons la formule suivante:

$$\mathsf{AG} \bigwedge_i \Big[\mathit{Call}_i \;\Rightarrow\; \mathsf{AF}\; \mathit{servicing}_i \Big] \tag{3.17}$$

$$\mathsf{AG} \bigwedge_i \left[\mathit{Send}_i \;\Rightarrow\; \mathsf{AF}\, \mathit{servicing}_i \right] \tag{3.18}$$

L'opérateur AF permet d'exprimer que sur toutes les exécutions possibles à partir de l'état courant (qui vérifie Call_i ou Send_i) il existe un état où la cabine desservira l'étage i.

Les propriétés suivantes sont des propriétés de *safety*.

∘ Tout d'abord, nous devons préciser qu'*il n'y a pas d'arrêt sans requête*. Nous définissons la propriété suivante: $\mathit{request}_i \stackrel{\text{def}}{=} \mathit{C\text{-}Light}_i \vee \mathit{S\text{-}Light}_i$. Un état vérifie $\mathit{request}_i$ si il existe une requête pour l'étage i non satisfaite.

Comme pour les propriétés (3.9) et (3.10), on pourrait être tenté d'utiliser une formule de la forme $\mathsf{AG} \bigwedge_i \left(\mathit{servicing}_i \;\Rightarrow\; \mathsf{F}^{-1} \mathit{request}_i \right)$, mais celle-ci ne convient pas: en effet, il suffirait là aussi qu'un état vérifie $\mathit{request}_i$ pour autoriser à tous les états suivants l'arrêt à l'étage i. Nous verrons au chapitre 5 que l'opérateur *Now* associé à F^{-1} résoud aussi ce problème. Ici nous allons utiliser l'opérateur S :

$$\mathsf{AG} \bigwedge_i \left[\mathit{servicing}_i \;\Rightarrow\; \left(\mathit{servicing}_i \;\mathsf{S}\; \mathit{request}_i \right) \right] \tag{3.19}$$

Cette formule énonce le fait que la cabine dessert l'étage i depuis qu'une requête existait pour cet étage: avant de le desservir, il existait donc une requête.

∘ Pour exprimer que *la cabine reste à l'étage où elle se trouve (i.e. le dernier étage qu'elle a desservi) tant qu'il n'y a pas de requête*, nous utilisons la formule suivante:

$$\mathsf{AG} \bigwedge_i \left[\mathit{servicing}_i \;\Rightarrow\; \mathsf{A}\; (\mathit{servicing}_i) \mathsf{W} \left(\bigvee_{k \neq i} \mathit{request}_k \right) \right] \tag{3.20}$$

∘ Nous devons énoncer que *lors de ses déplacements, la cabine doit s'arrêter aux étages par lesquels elle passe, lorsqu'il existe des requêtes pour ces étages*: lorsqu'elle se déplace [3] de l'étage j à l'étage i, alors elle doit s'arrêter à tout étage k situé entre i et j si il existe une requête pour k. On peut interpréter cette spécification de deux manières: la requête concernant l'étage k peut être prise en compte soit si elle a lieu à tout moment avant le passage de la cabine à l'étage k, soit si elle a lieu seulement avant le départ de la cabine de l'étage j (i.e. le prochain arrêt est fixé lors du départ de l'étage j).

Nous définissons l'abréviation suivante:

$$\mathit{at}\ i\ \mathit{from}\ j \stackrel{\text{def}}{=} \mathit{servicing}_i \;\wedge\; \left(\mathit{servicing}_i \;\vee\; \mathit{Noservice} \right) \mathsf{S}\; \mathit{servicing}_j$$

[3] au sens de quitter l'étage j où la cabine était arrêtée, pour aller s'arrêter à l'étage i

Avec $\mathit{Noservice} \stackrel{\text{def}}{=} \bigwedge_l \neg \mathit{servicing}_l$. Si un état [4] vérifie *at i from j*, alors dans cet état, la cabine est à l'étage i et son précédent arrêt était à l'étage j. Un état vérifie *Noservice*, si aucun étage n'est desservi.

- Si la cabine est partie de l'étage j pour s'arrêter à l'étage i, alors il n'existait pas de requête pour un étage intermédiaire k (situé entre j et i) **au moment du passage de la cabine à l'étage k :**

$$\mathsf{AG} \bigwedge_{i,j \neq i} \left[\mathit{at\ i\ from\ j} \;\Rightarrow\; \left[\left(\bigwedge_{\substack{i<k<j \\ j<k<i}} \left[\mathit{at}_k \;\Rightarrow\; (\neg \mathit{request}_k) \right] \right) \mathsf{S}\ \mathit{servicing}_j \right] \right] \tag{3.21}$$

Le domaine $i < k < j, j < k < i$ permet de considérer les k compris entre i et j sans alourdir les formules en distinguant $i < j$ et $j < i$.

- Si la cabine est partie de l'étage j pour s'arrêter à l'étage i, alors il n'existait pas de requête pour un étage intermédiaire k (situé entre j et i) **au moment du départ de la cabine de l'étage j:**

$$\mathsf{AG} \bigwedge_{i,i \neq i} \left[\mathit{at\,i\,from\,j} \;\Rightarrow\; \left[\left(\mathit{servicing}_i \vee \mathit{Noservice} \right) \mathsf{S} \left(\mathit{servicing}_j \wedge \bigwedge_{\substack{i<k<j \\ j<k<i}} (\neg \mathit{request}_k) \right) \right] \right] \tag{3.22}$$

○ Il faut préciser que *lorsque la cabine se déplace* [5] *de j à i, elle ne passe que par des étages situés entre j et i*:

$$\mathsf{AG} \bigwedge_{i,j \neq i} \left[\mathit{at\ i\ from\ j} \;\Rightarrow\; \left[\left(\big(\bigvee_{\substack{i \leq k \leq j \\ j \leq k \leq i}} \mathit{at}_k \big) \vee \mathit{between\text{-}levels} \right) \mathsf{S}\ \mathit{servicing}_j \right] \right] \tag{3.23}$$

Nous allons définir les propriétés *down* et *up* que nous utiliserons par la suite. Intuitivement, un état vérifie *up* (resp. *down*), lorsque la cabine est en train, ou vient de se déplacer vers le haut (resp. vers le bas):

$$\mathit{up} \stackrel{\text{def}}{=} \bigvee_{i=2,\ldots,n} \left[\left((\mathit{at}_i \vee \mathit{between\text{-}levels})\ \mathsf{S}\ \mathit{at}_{i-1} \right) \wedge \mathsf{A} \left(\mathit{between\text{-}levels}\ \mathsf{U}\ \mathit{at}_i \right) \right] \tag{3.24}$$

[4] En fait, il s'agit d'un état avec une histoire, c'est-à-dire un état de l'arbre représentant les comportements possibles de l'ascenseur

[5] même sens que précédemment

$$\mathit{down} \stackrel{\text{def}}{=} \bigvee_{i=1,\ldots,n-1} \Big[\Big((\mathit{at}_i \ \vee \ \mathit{between\text{-}levels}) \ \mathsf{S} \ \mathit{at}_{i+1} \Big) \ \wedge \ \mathsf{A} \ \Big(\mathit{between\text{-}levels} \ \mathsf{U} \ \mathit{at}_i \Big) \Big] \quad (3.25)$$

Un état vérifie *up* lorsque soit la cabine est à un étage i et vient de $i-1$, soit la cabine se trouve entre deux étages, en provenance de l'étage $i-1$, et se déplace vers l'étage i. On utilise la même démarche pour *down*.

∘ Nous venons d'énoncer que la cabine ne devait passer, lors de ses déplacements, que par les étages qu'il était nécessaire de traverser. Nous devons aussi préciser que *la cabine ne peut pas changer de direction en cours de déplacement*:

$$\mathsf{AG} \ \Big[\Big(\mathit{Noservice} \ \wedge \ \mathit{down} \Big) \ \Rightarrow \ \mathsf{A} \ (\mathit{down}) \ \mathsf{W} \ (\bigvee_i \mathit{servicing}_i) \Big] \qquad (3.26)$$

$$\mathsf{AG} \ \Big[\Big(\mathit{Noservice} \ \wedge \ \mathit{up} \Big) \ \Rightarrow \ \mathsf{A} \ (\mathit{up}) \ \mathsf{W} \ (\bigvee_i \mathit{servicing}_i) \Big] \qquad (3.27)$$

La formule (3.26) énonce que lorsque la cabine se déplace (*Noservice*) vers le bas (*down*), alors elle continuera à descendre tant que elle ne desservira pas un étage (c'est seulement après s'être arrêtée qu'elle peut changer de direction).

∘ Il nous reste à énoncer que *la cabine doit choisir en priorité, les requêtes permettant de continuer dans la même direction*:

$$\mathsf{AG} \bigwedge_i \Big(\Big[\mathit{servicing}_i \wedge \mathit{down} \wedge \bigvee_{k<i} (\mathit{request}_k) \Big] \ \Rightarrow \ \neg \mathsf{E} \Big[(\mathit{servicing}_i \vee \mathit{Noservice}) \ \mathsf{U} \ (\bigvee_{j>i} \mathit{servicing}_j) \Big] \Big) \quad (3.28)$$

$$\mathsf{AG} \bigwedge_i \Big(\Big[\mathit{servicing}_i \ \wedge \ \mathit{up} \ \wedge \ \bigvee_{k>i} (\mathit{request}_k) \Big] \ \Rightarrow \ \neg \mathsf{E} \Big[(\mathit{servicing}_i \ \vee \ \mathit{Noservice}) \ \mathsf{U} \ (\bigvee_{j<i} \mathit{servicing}_j) \Big] \Big) \quad (3.29)$$

La formule (3.28) énonce que si la cabine dessert un étage i en venant d'un étage supérieur (*down*) et si il existe une requête pour un étage j inférieur ($j < i$), alors le prochain étage desservi ne sera pas un étage supérieur (à i). La formule (3.29) énonce une propriété similaire dans le cas où la cabine se déplace vers les étages supérieurs.

On peut aussi exprimer ces propriétés avec les formules suivantes qui expriment le fait que si la cabine dessert un étage i en venant d'un étage supérieur (resp. inférieur) et si il existe une requête pour un étage inférieur (resp. supérieur) que i, alors la cabine ne peut monter (resp. descendre) immédiatement après avoir quitté l'étage i:

$$\mathsf{AG} \bigwedge_i \Big(\Big[\mathit{servicing}_i \ \wedge \ \mathit{down} \ \wedge \ \bigvee_{k<i} (\mathit{request}_k) \Big] \ \Rightarrow \ \neg \mathsf{E} \Big[\mathit{servicing}_i \ \mathsf{U} \ \mathit{up} \Big] \Big) \tag{3.30}$$

$$\mathsf{AG} \bigwedge_i \Big(\Big[\mathit{servicing}_i \ \wedge \ \mathit{up} \ \wedge \ \bigvee_{k>i} (\mathit{request}_k) \Big] \ \Rightarrow \ \neg \mathsf{E} \Big[\mathit{servicing}_i \ \mathsf{U} \ \mathit{down} \Big] \Big) \tag{3.31}$$

3.3.5 Les conditions initiales

A l'état initial, la cabine se trouve à l'étage 1, la porte est ouverte, tous les voyants lumineux sont éteints et les boutons sont relâchés:

$$\mathit{at}_1 \wedge \bigwedge_{i>1} \neg \mathit{at}_i \wedge \mathit{OpenD}_1 \wedge \bigwedge_{i>1} \neg \mathit{OpenD}_i \wedge \bigwedge_i \neg \mathit{C\text{-}Light}_i \wedge \bigwedge_i \neg \mathit{S\text{-}Light}_i \wedge \bigwedge_i \neg \mathit{Call}_i \wedge \bigwedge_i \neg \mathit{Send}_i \tag{3.32}$$

A partir de la spécification informelle de la section 3.2, nous avons donc construit une spécification $SPEC_1$ en $CTL+$ Passé:

$SPEC_1 =$	(3.12) $\wedge$ (3.13)	voyants lumineux et boutons
	$\wedge$ (3.14)	les portes
	$\wedge$ (3.15) $\wedge$ (3.16)	la cabine
	$\wedge$ (3.17) $\wedge$ (3.18) $\wedge$ (3.19) $\wedge$ (3.20) $\wedge$ (3.21)	requêtes et comportement
	$\wedge$ (3.23) $\wedge$ (3.26) $\wedge$ (3.27) $\wedge$ (3.28) $\wedge$ (3.29)	de la cabine
	$\wedge$ (3.32)	conditions initiales

A partir de la même description informelle, nous allons maintenant essayer d'établir une spécification $SPEC_2$ en CTL.

3.4 Une spécification sans passé

Pour construire une spécification sans utiliser d'opérateur du passé, nous allons essayer de remplacer les formules de $SPEC_1$ munies d'opérateurs (S) par des formules de CTL (et donc pur-futur) équivalentes.

3.4.1 Voyants lumineux et boutons

∘ Pour la spécification du comportement des voyants et des boutons, seules les formules (3.9) et (3.10) contiennent des opérateurs du passé. En fait, on veut spécifier que *si un voyant s'allume, c'est que le bouton associé a été enfoncé*. On peut exprimer cette propriété avec les opérateurs de CTL en imposant qu'*un voyant reste éteint jusqu'à ce que le bouton soit enfoncé*:

$$\mathsf{AG} \bigwedge_i \left[(\neg \textit{C-Light}_i) \;\Rightarrow\; \mathsf{A}\,(\neg \textit{C-Light}_i)\,\mathsf{W}\;\textit{Call}_i \right] \tag{3.33}$$

$$\mathsf{AG} \bigwedge_i \left[(\neg \textit{S-Light}_i) \;\Rightarrow\; \mathsf{A}\,(\neg \textit{S-Light}_i)\,\mathsf{W}\;\textit{Send}_i \right] \tag{3.34}$$

Nous sommes donc parvenus à exprimer la même propriété sans opérateur du passé (nous verrons dans l'exemple 4.5.9 du chapitre 4 que les formules (3.9) (resp. (3.10)) et (3.33) (resp (3.34)) sont équivalentes initialement), néanmoins on peut noter que (3.33) et (3.34) sont moins naturelles et moins lisibles que (3.9) et (3.10). De plus, il n'est pas possible de construire facilement des formules du type de (3.12) et (3.13) qui expriment le comportement des voyants en réponse aux boutons et à la cabine de manière concise.

3.4.2 La cabine

∘ La formule (3.16) exprime que la cabine se déplace de manière continue, en imposant que lorsque la cabine se trouve à l'étage i, elle se trouvait précédemment à l'étage $i-1$ ou $i+1$. On peut exprimer une telle propriété avec les opérateurs du futur en imposant des conditions, non pas sur la provenance, mais sur les destinations de la cabine:

$$\begin{aligned} &\mathsf{AG} \bigwedge_{i=2,\ldots,n-1} \left[\textit{at}_i \;\Rightarrow\; \mathsf{A}\left((\textit{at}_i \vee \textit{between-levels})\mathsf{W}\,(\textit{at}_{i-1} \vee \textit{at}_{i+1}) \right) \right] \\ &\wedge\; \mathsf{AG}\left[\textit{at}_1 \;\Rightarrow\; \mathsf{A}\left((\textit{at}_1 \vee \textit{between-levels})\mathsf{W}\,\textit{at}_2 \right) \right] \\ &\wedge\; \mathsf{AG}\left[\textit{at}_n \;\Rightarrow\; \mathsf{A}\left((\textit{at}_n \vee \textit{between-levels})\mathsf{W}\,\textit{at}_{n-1} \right) \right] \end{aligned} \tag{3.35}$$

Une autre manière d'exprimer cette propriété serait d'énoncer que lorsque la cabine se trouve à un étage k avec $k \neq i, i+1, i-1$, alors elle n'atteindra pas l'étage i avant d'être passé par $i-1$ ou $i+1$.

3.4.3 Requêtes et comportement de la cabine

∘ La formule (3.19) qui exprime que *la cabine ne s'arrête que si il y a eu une requête*, utilise l'opérateur S. Les opérateurs du futur permettent d'exprimer la propriété suivante: *un étage n'est pas desservi tant qu'il n'y a pas de requête*:

$$\mathsf{AG} \bigwedge_i \Big[(\neg \mathit{servicing}_i) \;\Rightarrow\; \mathsf{A}\,(\neg \mathit{servicing}_i)\,\mathsf{W}\,\,(\mathit{request}_i)\Big] \tag{3.36}$$

∘ Les formules (3.21) et (3.22) correspondent à deux choix différents pour l'interprétation de la propriété “la cabine doit s'arrêter, lors de ses déplacements, aux étages pour lesquels il existe une requête”. Nous pouvons les exprimer avec CTL^+:

- La formule (3.21) exprime la propriété suivante: *lorsque la cabine se trouve à l'étage i en provenance de l'étage j, alors il n'existait pas de requête pour les étages intermédiaires k (situés entre j et i) au moment où la cabine y est passée.* On peut utiliser la formule suivante:

$$\mathsf{AG} \bigwedge_{j,i\neq j} \Big[\mathit{servicing}_j \;\Rightarrow\; \mathsf{A}\Big(\big[\mathit{servicing}_j \vee \mathit{Noservice}\big]\mathsf{U}(\mathit{servicing}_i) \;\Rightarrow\; \bigwedge_{\substack{j<k<i\\ i<k<j}} \big[(\mathit{at}_k \;\Rightarrow\; \neg \mathit{request}_k)\mathsf{U}\mathit{servicing}_i\big]\Big)\Big] \tag{3.37}$$

Un état où la cabine dessert l'étage j vérifie cette formule, si toutes les exécutions qui correspondent à un déplacement de j vers i, vérifient $(\mathit{at}_k \;\Rightarrow\; \neg \mathit{request}_k)\,\mathsf{U}\,\mathit{servicing}_i$ pour tous les k entre j et i, c'est à dire si aucune requête concernant un étage intermédiaire n'existe au passage de la cabine à cet étage.

On peut noter que la formule (3.37) appartient à la logique CTL^+ (et non à CTL): il est possible de traduire cette formule en CTL mais on perd alors en lisibilité.

- La formule (3.22) exprime la propriété suivante: *si la cabine a quitté l'étage j pour atteindre l'étage i, alors il n'existait pas de requête pour les étages intermédiaires k (situés entre j et i) au moment où la cabine a quitté l'étage j*. On peut utiliser la formule suivante:

$$\mathsf{AG} \bigwedge_{j,i\neq j} \Big[\Big(\mathit{servicing}_j \;\wedge\; \mathsf{E}\,\big[\mathit{servicing}_j \;\vee\; \mathit{Noservice}\big]\,\mathsf{U}\,(\mathit{servicing}_i)\Big) \;\Rightarrow\; \bigwedge_{\substack{j<k<i\\ i<k<j}} (\neg \mathit{request}_k)\Big] \tag{3.38}$$

Cette formule exprime le fait que si la cabine dessert l'étage j et qu'il existe une exécution où l'étage i sera la prochaine destination de la cabine, alors il n'existe pas de requête pour un étage situé entre j et i.

∘ Pour remplacer la formule (3.23) qui énonce que *lorsque la cabine se déplace de j à i, elle ne passe que par des étages situés entre j et i*, on peut utiliser la formule suivante:

$$\mathsf{AG} \bigwedge_{j,i\neq i} \Big[\mathit{servicing}_j \;\Rightarrow\; \mathsf{A}\Big(\big[\mathit{servicing}_j \vee \mathit{Noservice}\big] \,\mathsf{U}\, (\mathit{servicing}_i) \;\Rightarrow\; \big[\mathit{between\text{-}levels} \vee \bigvee_{\substack{j\leq k\leq i\\ i\leq k\leq j}} \mathit{at}_k\big] \,\mathsf{U}\, (\mathit{servicing}_i)\Big)\Big] \tag{3.39}$$

Cette formule énonce que si la cabine dessert l'étage j, alors le long de toutes les exécutions qui correspondent à un déplacement vers l'étage i, la cabine est soit entre deux étages, soit à un étage situé entre j et i. Cette formule appartient à la logique CTL^+.

∘ Les formules (3.26) et (3.27) expriment la propriété suivante: *la cabine ne peut changer de direction qu'après s'être arrêtée*. Pour cela nous avons utilisé les formules *up* et *down* qui utilisent du passé. Nous allons pouvoir exprimer cette propriété en énonçant qu'il n'existe pas d'exécution qui conduit de l'étage i à l'étage $i+1$ (resp. $i-1$) puis de nouveau à l'étage i sans que la cabine ne se soit arrêtée en $i+1$ (resp. $i-1$):

$$\mathsf{AG} \bigwedge_i \Big[\mathit{at}_i \;\Rightarrow\; \neg\mathsf{E}\big(\mathit{at}_i \vee \mathit{between\text{-}levels}\big) \,\mathsf{U}\, \Big(\mathit{at}_{i+1} \wedge \mathsf{E}\big[(\mathit{at}_{i+1} \vee \mathit{between\text{-}levels}) \wedge \mathit{Noservice}\big] \,\mathsf{U}\, \mathit{at}_i\Big)\Big] \tag{3.40}$$

$$\mathsf{AG} \bigwedge_i \Big[\mathit{at}_i \;\Rightarrow\; \neg\mathsf{E}\big(\mathit{at}_i \vee \mathit{between\text{-}levels}\big) \,\mathsf{U}\, \Big(\mathit{at}_{i-1} \wedge \mathsf{E}\big[(\mathit{at}_{i-1} \vee \mathit{between\text{-}levels}) \wedge \mathit{Noservice}\big] \,\mathsf{U}\, \mathit{at}_i\Big)\Big] \tag{3.41}$$

∘ Il nous reste à spécifier que *la cabine doit desservir prioritairement les requêtes qui évitent de changer de direction* (formules (3.28) et (3.29)):

$$\begin{aligned}\mathsf{AG} \Big[\bigwedge_j \; & \bigwedge_{i>j} \Big[\mathit{servicing}_j \;\Rightarrow\; \neg\mathsf{E}\,\big(\mathit{servicing}_j \vee \mathit{Noservice}\big) \,\mathsf{U}\, \Big(\mathit{servicing}_i \wedge \\ & \qquad \mathsf{E}\,\mathit{servicing}_i \,\mathsf{U}\, \Big[\mathit{servicing}_i \wedge \bigvee_{k>i} \mathit{request}_k \wedge \mathsf{E}\,\big(\mathit{servicing}_i \vee \mathit{Noservice}\big) \,\mathsf{U}\, \Big(\bigvee_{k'<i} \mathit{servicing}_{k'}\Big)\Big]\Big)\Big] \\ & \bigwedge_{i<j} \Big[\mathit{servicing}_j \;\Rightarrow\; \neg\mathsf{E}\,\big(\mathit{servicing}_j \vee \mathit{Noservice}\big) \,\mathsf{U}\, \Big(\mathit{servicing}_i \wedge \\ & \qquad \mathsf{E}\,\mathit{servicing}_i \,\mathsf{U}\, \Big[\mathit{servicing}_i \wedge \bigvee_{k<i} \mathit{request}_k \wedge \mathsf{E}\,\big(\mathit{servicing}_i \vee \mathit{Noservice}\big) \,\mathsf{U}\, \Big(\bigvee_{k'>i} \mathit{servicing}_{k'}\Big)\Big]\Big)\Big]\Big]\end{aligned} \tag{3.42}$$

La formule (3.42) exprime le fait que si la cabine dessert l'étage j, alors il n'existe pas d'exécution où la cabine atteint un étage i avec $i > j$ (resp. $i < j$) puis quitte cet étage pour aller vers un étage k' avec $k' < i$ (resp. $k' > i$) alors qu'il existait une requête pour un étage k avec $k > i$ (resp. $k < i$). La sous-formule $\mathit{servicing}_i \wedge \mathsf{E}\, \mathit{servicing}_i \,\mathsf{U}\, (\mathit{servicing}_i \wedge \ldots)$ permet de spécifier que la requête pour un étage k peut exister à tout moment tant que la cabine dessert l'étage i (et pas seulement lorsque celle-ci arrive en i).

Une spécification $SPEC_2$ en CTL de l'ascenseur décrit à la section 3.2 peut donc être définie de la manière suivante:

$SPEC_2 =$	(3.5) $\wedge$ (3.6) $\wedge$ (3.7)	voyants lumineux
	$\wedge$ (3.8) $\wedge$ (3.11) $\wedge$ (3.33) $\wedge$ (3.34)	et boutons
	$\wedge$ (3.14)	les portes
	$\wedge$ (3.15) $\wedge$ (3.35)	la cabine
	$\wedge$ (3.17) $\wedge$ (3.18) $\wedge$ (3.36) $\wedge$ (3.20) $\wedge$ (3.37)	requêtes et comportement
	$\wedge$ (3.39) $\wedge$ (3.40) $\wedge$ (3.41) $\wedge$ (3.42)	de la cabine
	$\wedge$ (3.32)	conditions initiales

3.5 Comparaison des spécifications $SPEC_1$ et $SPEC_2$

A partir d'une description informelle d'un ascenseur, nous avons donc construit deux spécifications en logique temporelle. La première, $SPEC_1$, est composée de formules de $CTL+$ passé. La seconde, $SPEC_2$, est composée de formules de CTL ou CTL^+.

Tout d'abord, nous pouvons remarquer que les formules de $SPEC_1$ sont plus naturelles car elles correspondent mieux à la description informelle que les formules de $SPEC_2$. Ainsi pour spécifier en CTL qu'un voyant ne s'allume que si le bouton a été enfoncé, on énonce qu'un voyant reste éteint tant que le bouton n'est pas enfoncé. L'absence d'opérateurs du passé nous conduit donc à changer de point de vue (en se plaçant dans le temps avant que le voyant ne s'allume) par rapport à la description informelle.

Ainsi peut-on se convaincre relativement facilement de la correspondance entre la description informelle et la spécification $SPEC_1$, mais cela est beaucoup plus difficile en ce qui concerne $SPEC_2$. Ceci plaide fortement en faveur de l'utilisation des logiques temporelles avec passé comme langage de spécification.

D'autre part, pour construire $SPEC_2$, nous avons remplacé les formules de $SPEC_1$ avec passé par des formules pur-futur sensées spécifier la même propriété de la description informelle initiale. Mais si l'on considère, par exemple, le remplacement de la conjonction des formules (3.28) et (3.29) par la formule (3.42), il est particulièrement difficile de se convaincre de leur équivalence tant leur structure est complexe et le point de vue adopté est différent.

3.6 Conclusion

Nous pouvons maintenant faire une synthèse des différentes remarques observées durant ce chapitre et le chapitre précédent. Tout d'abord, la logique $CTL +$ passé permet l'expression de spécifications plus naturelles et concises que les logiques CTL ou CTL^{+}. De plus, en ce qui concerne la vérification automatique, la logique CTL est la mieux adaptée puisqu'elle dispose d'algorithmes très efficaces. Enfin, il apparaît possible mais délicat de passer d'une spécification en $CTL +$ *passé* à une spécification en CTL.

Ces trois remarques soulignent l'intérêt qu'aurait une traduction (implémentable) de $CTL +$ passé dans CTL. Cela permettrait de conjuguer les avantages de $CTL +$ passé pour la spécification et ceux de CTL pour la vérification.

Au cours du chapitre suivant, nous allons nous intéresser à la construction de telles traductions (ou à la preuve de leur impossibilité).

Chapitre 4

Des fragments de $PCTL^*$, expressivité comparée

4.1 Introduction et motivations

Aux cours du chapitre précédent, nous avons constaté l'utilité de la logique temporelle pour la spécification. Dans l'exemple de l'ascenseur, nous avons pu noter l'intérêt pratique qu'apportent les opérateurs du passé pour exprimer simplement les propriétés comportementales. Cette remarque avait déjà été soulignée dans plusieurs études [LPZ85, KVd83]. Néanmoins parmi les logiques temporelles présentées dans la littérature informatique, peu sont pourvues d'opérateurs du passé.

Cette absence s'explique par le fait que les progrès considérables réalisés pour le model-checking qui ont permis la construction d'outils de vérification automatique très efficaces, ne concernent que des logiques d'états (donc pur-futur) comme par exemple CTL. Dès lors, une solution envisageable serait d'utiliser des logiques temporelles avec passé pour la spécification et de se ramener automatiquement, pour la partie vérification, à des formalismes plus adéquats, comme certaines logiques du futur. Il s'agirait donc d'étudier et d'automatiser ce passage ou traduction entre un langage de spécification pratique (permettant une expression aisée des propriétés) et un langage efficace pour la vérification. Une fois défini, ce type de traduction permettrait d'ajouter aux outils de vérifications classiques un module de traduction en amont du model-checking: on spécifierait avec un langage plus expressif, et la vérification se ferait avec un model-checker classique sur la spécification traduite.

Le problème que nous allons aborder dans ce chapitre, concerne la traduction de logiques avec passé dans des logiques pour lesquelles il existe des algorithmes efficaces pour le model-checking comme CTL ou $ECTL^+$.

On peut noter que les résultats de traduction de logiques avec passé vers des logiques du futur, présentés au chapitre 2, ne concernaient que des logiques (PTL et CTL^*) ayant des problèmes de model-checking PSPACE-complets c'est-à-dire n'admettant pas de solutions efficaces. Ainsi l'absence de traduction concernant les logiques CTL ou $ECTL^+$ conduit-elle, lorsqu'on veut traduire CTL + Passé dans une logique du futur, à traduire les formules ainsi construites en CTL^*: on passe donc d'un algorithme de vérification qui dépend linéairement de la taille de la formule et du modèle, à un algorithme dont la complexité est exponentielle.

L'objet de ce chapitre est d'étudier en détail l'introduction d'opérateurs du passé dans l'ensemble des fragments que nous avons présentés au chapitre 2. Nous présenterons d'abord des résultats pour le temps linéaire, notamment pour introduire les techniques de traduction mentionnées au chapitre 2. Ces techniques nous seront utiles pour aborder le cas du temps arborescent. Nous présenterons, ensuite, une série de résultats concernant l'expressivité des logiques du temps arborescent munies d'opérateurs du passé. Plusieurs de ces résultats figurent dans [LS94]. Ce chapitre constitue le cœur technique de la thèse.

4.2 Le temps linéaire

Le théorème de séparation 2.8.8 proposé dans [GPSS80] est un résultat fondamental parmi ceux concernant l'expressivité des logiques temporelles avec passé. Dans un premier temps, nous allons présenter la démarche sous-jacente à ce théorème. Ensuite, nous considérerons l'introduction d'opérateurs du passé dans la logique LTL, ce dernier résultat nous sera utile ultérieurement pour comparer certaines logiques du temps arborescent.

4.2.1 Le théorème de séparation [GPSS80]

Nous présentons ici l'idée générale du théorème de séparation 2.8.8 dont la preuve figure dans [Gab87]. On dira qu'une formule est *séparée* si f est une combinaison booléenne de formules pur-passé et pur-futur (c.-à-d. sans opérateur du passé dans la portée d'opérateurs du futur et sans opérateur du futur dans la portée d'opérateurs du passé).

Le théorème de séparation énonce que toute formule de $L(\mathsf{U}_s, \mathsf{S}_s)$ est équivalente à une formule séparée et montre comment on peut *calculer effectivement* une telle formule. En effet, pour prouver ce théorème, Gabbay donne une stratégie permettant de transformer, à l'aide de règles de réécriture, toute formule de $L(\mathsf{U}_s, \mathsf{S}_s)$ en une formule séparée équivalente.

Chaque règle de réécriture (appelée règle de séparation) a pour objectif d'extraire un opérateur U_s (resp. S_s) de la portée immédiate d'un opérateur S_s (resp. U_s): on distingue différents cas, selon que l'opérateur U_s (resp. S_s) est dans la partie gauche ou droite du S_s (resp. U_s), et selon qu'il est, ou non, précédé par une négation. La distributivité des parties droites (resp. gauche) de *Until* pour la disjonction (resp. la conjonction) permet de simplifier les cas à considérer. Considérons l'équivalence suivante:

$$\alpha\,\mathsf{S}_s\left(\beta \wedge (\gamma\,\mathsf{U}_s\,\delta)\right) \equiv \begin{array}{l} \left[\alpha\,\mathsf{S}_s\,\beta \wedge \gamma\,\mathsf{S}_s\,\beta \wedge \gamma \wedge \gamma\,\mathsf{U}_s\,\delta\right] \\ \vee \left[\delta \wedge (\gamma \wedge \alpha)\,\mathsf{S}_s\,\beta\right] \\ \left[\alpha\,\mathsf{S}_s\left(\delta \wedge \alpha \wedge \gamma\,\mathsf{S}_s\,\beta \wedge \alpha \vee \quad \mathsf{S}_s\,\beta\right)\right] \end{array} \tag{4.43}$$

Nous pouvons utiliser cette équivalence comme une règle de séparation de manière à remplacer les formules ayant un U_s dans la portée d'un S_s (comme la formule de gauche) par une formule équivalente où l'opérateur U_s a été extrait de la portée de S_s.

A travers cet exemple, on peut remarquer que si la formule de droite ne contient plus d'occurrence de U_s dans la portée de S_s, les sous-formules α, β et γ se retrouvent dans la portée de deux S_s emboîtés. Il est clair qu'une stratégie particulière doit être utilisée dans l'emploi des règles de séparation pour permettre la terminaison d'une telle transformation. L'idée clef de cette stratégie consiste à traiter toutes les occurrences d'une sous-formule $\gamma\,\mathsf{U}_s\,\delta$ simultanément: en plus des cas mentionnés ci-dessus, on doit donc considérer les cas où deux sous-formules $\gamma\,\mathsf{U}_s\,\delta$ sont présentes dans la portée du S_s (une dans la partie gauche, une dans la partie droite, et chacune éventuellement précédée par une négation). On considère donc 8 règles différentes pour extraire les sous-formules $\gamma\,\mathsf{U}\,\delta$ de la portée d'un *Since*, et autant pour extraire les sous-formules $\gamma\,\mathsf{S}\,\delta$ de la portée d'un *Until*.

Nous reviendrons plus précisement sur la stratégie utilisée par Gabbay au cours de la section suivante (pour le temps arborescent). Le théorème 2.8.7 énonçant $L(\mathsf{U}_s) \equiv_i L(\mathsf{U}_s, \mathsf{S}_s)$, peut être considéré comme une conséquence immédiate du théorème de séparation. En effet, on peut transformer facilement tout formule séparée de $L(\mathsf{U}_s, \mathsf{S}_s)$ en une formule de $L(\mathsf{U}_s)$ équivalente initialement, puisque toute sous-formule pur-passé est équivalente initialement à une combinaison booléenne de propositions atomiques :

$$\mathsf{X}^{-1} f \;\equiv_i\; \bot \qquad\qquad f\,\mathsf{S}\,g \;\equiv_i\; g \qquad\qquad \mathsf{F}^{-1} f \;\equiv_i\; f$$

Puisque $PPTL \equiv L(\mathsf{U}_s, \mathsf{S}_s)$ et $PTL \equiv L(\mathsf{U}_s)$, on obtient aisément que $PPTL \equiv_i PTL$. On peut néanmoins adapter les règles de séparation pour les opérateurs de $PPTL$, c'est à dire extraire les opérateurs X^{-1} et S (resp. X et U) de la portée d'un X ou d'un U (resp. X^{-1} ou S) [Lar91].

4.2.2 La logique $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$

Nous allons maintenant considérer la logique $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$ obtenue à partir de LTL en ajoutant X^{-1} et F^{-1}. Nous la noterons $PLTL$ et nous allons la comparer avec LTL et PTL.

1 – $LTL \prec_i L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$

Comme $LTL \subseteq PLTL$, il est clair que $LTL \preceq_i PLTL$. Pour montrer $LTL \prec_i PLTL$, il faut prouver qu'il existe au moins une formule de $PLTL$ qui n'est équivalente initialement à aucune formule de LTL.

Nous allons montrer que $a \mathsf{U} b$ est inexprimable initialement en LTL. Ensuite nous montrerons que $a \mathsf{U} b$ est exprimable initialement par $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$, et nous pourrons alors en déduire que $LTL \prec_i L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$.

Proposition 4.2.1 *$a \mathsf{U} b$ est inexprimable initialement avec LTL*

Preuve Nous allons utiliser les modèles proposées dans [EH85] pour montrer que $\mathsf{E}\, a \mathsf{U} b$ est inexprimable en UB^+. On considère deux familles de modèles linéaires (voir figure 4.1) M_{ki} et N_{ki} tels que:

$$M_{ki} = s_0 s_1 \ldots \quad \text{et} \quad \begin{cases} s_j \models a \wedge \neg b & \text{si } \forall t,\ j \neq i + tk \\ s_j \models a \wedge b & \text{si } \exists t \text{ pair } \mid j = i + tk\ \forall t \\ s_j \models \neg a \wedge \neg b & \text{si } \exists t \text{ impair } \mid j = i + tk\ \forall t \end{cases}$$

$$N_{ki} = s'_0 s'_1 \ldots \quad \text{et} \quad \begin{cases} s'_j \models a \wedge \neg b & \text{si } \forall t,\ j \neq i + tk \\ s'_j \models \neg a \wedge \neg b & \text{si } \exists t \text{ pair } \mid j = i + tk\ \forall t \\ s'_j \models a \wedge b & \text{si } \exists t \text{ impair } \mid j = i + tk\ \forall t \end{cases}$$

Il est clair que pour tout $k > 0$ et tout $i > 0$, on a $M_{ki}, s_0 \models a \mathsf{U} b$ et $N_{ki}, s'_0 \not\models a \mathsf{U} b$. De plus nous avons le lemme suivant prouvé dans [EH85]:

Lemme 4.2.2 *$\forall f \in UB^+$, si $\mid f \mid \leq i$ Alors $M_{ki}, s_0 \models f \quad \Leftrightarrow \quad N_{ki}, s'_0 \models f$.*

Preuve Voir page 21 de [EH85]. □

On en déduit le corollaire suivant:

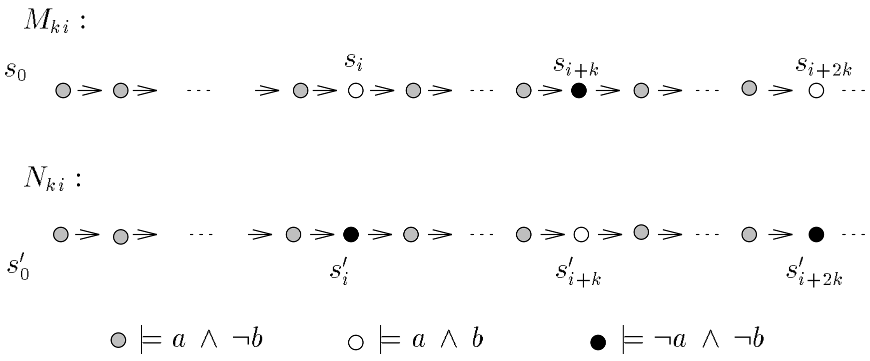


Figure 4.1: Modèles M_{ki} et N_{ki}

Corollaire 4.2.3 *$\forall f \in LTL$, si $|f| \leq i$ Alors $M_{ki}, s_0 \models f \quad \Leftrightarrow \quad N_{ki}, s'_0 \models f$.*

Preuve Supposons qu'il existe une formule f de LTL, de taille inférieure à i et telle que $M_{ki}, s_0 \models f$ et $N_{ki}, s'_0 \not\models f$. Considérons la formule de UB, $\tilde{f}$, obtenue à partir de f en remplaçant chaque occurrence de X par EX et chaque occurrence de F par EF. Comme les modèles M_{ki} et N_{ki} sont linéaires, on a $M_{ki}, s_o \models \tilde{f}$ et $N_{ki}, s'_0 \not\models \tilde{f}$, et ceci contredit le lemme précédent. □

Ce dernier corollaire permet de conclure qu'aucune formule de LTL n'est équivalente initialement à $a\ \mathsf{U}\ b$. □

A partir de la proposition précédente, on peut déduire le théorème suivant puisque la formule $a\ \mathsf{U}\ b$ est facilement exprimable initialement avec la logique $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1})$:

Théorème 4.2.4 *$LTL \prec_i L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1})$*

Preuve La formule $b \vee \mathsf{F}\,(\mathsf{X}\, b \wedge \mathsf{G}^{-1} a)$ de $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1})$ est équivalente initialement à $a\ \mathsf{U}\ b$, et n'est équivalente initialement à aucune formule de LTL d'après la proposition 4.2.1. □

En conséquence, on a bien: $LTL \prec_i L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$.

2 – $L(\mathbf{F}, \mathbf{X}, \mathbf{F}^{-1}, \mathbf{X}^{-1}) \prec_i L(\mathbf{U}, \mathbf{X})$

Kamp a montré que les opérateurs S_s et U_s permettent l'expression de propriétés inexprimables avec la logique $L(\mathsf{F}_s, \mathsf{X}, \mathsf{F}_s^{-1}, \mathsf{X}^{-1})$. Il en résulte que $PLTL \prec_i PTL$ (puisqu'en plus on a $PTL \equiv_i PPTL$). Ici nous allons montrer un résultat plus précis qui nous sera utile dans la suite: nous allons montrer que la formule $\mathsf{F}\,(a \wedge b\,\mathsf{U}\,c)$ est inexprimable avec $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$.

Pour montrer cela, nous utilisons une approche basée sur la théorie des jeux qui nous a été conseillée par Mark Reynolds (Imperial College). Nous allons définir un jeu du même type que ceux utilisés dans [IK87] et [GHR92]. Etant donnés deux modèles linéaires et des pions placés sur ces modèles, ce jeu consiste, pour chacun des deux joueurs, à déplacer les pions à tour de rôle en vérifiant à chaque mouvement un ensemble de conditions sur la place respective des pions. On montrera qu'il y a un lien entre l'existence de méthodes permettant à l'un des deux joueurs de gagner systématiquement et la distinguabilité de ces modèles par certaines logiques de temps linéaire.

Nous allons définir le jeu utilisé et nous démontrerons son lien avec une logique équivalente à $PLTL$. Ensuite nous définirons les deux familles de modèles que nous utiliserons pour montrer que la formule $\mathsf{F}\,(a \wedge b\,\mathsf{U}\,c)$ est inexprimable avec $PLTL$. Plusieurs preuves, longues et techniques, ont été placées en annexe.

Soient M et M' deux modèles linéaires étiquetés par des propositions atomiques. Sur un état de chacun de ces modèles, on dispose un jeton bleu ($bleu_M$ et $bleu_{M'}$) et on place un jeton rouge ($rouge_M$ et $rouge_{M'}$) de la même manière. Considérons deux joueurs, appelés respectivement I et II. Le jeu comporte r tours. Chaque tour se décompose en deux phases: le joueur I déplace un des quatre jetons sur son modèle d'origine, ensuite le joueur II doit déplacer le jeton de même couleur sur l'autre modèle, de manière à ce que: (1) l'ordre des jetons sur chaque modèle soit le même ($bleu_M < rouge_M$ ssi $bleu_{M'} < rouge_{M'}$ et $bleu_M = rouge_M$ ssi $bleu_{M'} = rouge_{M'}$), et (2) les jetons de même couleur doivent être placés sur des états vérifiant les mêmes propositions atomiques.

Si le joueur II ne peut pas déplacer son jeton de manière à vérifier (1) et (2), alors le joueur I a gagné la partie. Si à la fin du r^{eme} tour, les deux conditions sont toujours vérifiées, le joueur II a gagné.

Pour gagner, le joueur II doit donc "imiter" les coups de I, tandis que le joueur I doit essayer d'utiliser la différence entre M et M' pour empêcher II de déplacer correctement (au sens de (1) et (2)) les jetons.

On dira qu'une partie en r tours est *gagnante le joueur* I, si il existe une manière de jouer pour I de telle sorte qu'il ne soit pas possible à II de l'imiter pendant r coups.

De même, on dira qu'une partie en r tours est *gagnante pour le joueur* II, si II peut gagner les parties de r tours quels que soient les coups de I.

Définition 4.2.5 *Etant donnés deux modèles linéaires M et M' (définis par leur ordre $<$ et leurs fonctions d'étiquetage l et l'), $x, y \in M$ et $x', y' \in M'$, on note $G_r(M, x, y; M', x', y')$ le jeu décrit précédemment où x et x' représentent les positions des jetons bleus, et y et y' représentent les positions des jetons rouges. A chaque tour, nous devons avoir:*

$$\begin{array}{lll} x < y & ssi & x' < y' \\ x = y & ssi & x' = y' \\ x = y+1 & ssi & x' = y'+1 \\ y = x+1 & ssi & y' = x'+1 \\ & & \\ l(x) = l'(x') & et & l(y) = l'(y') \end{array}$$

Nous noterons cette condition sur x, y, x', y' de la manière suivante: $x, y \sim x', y'$.

On note $G_r(M, x, y; M', x', y') \in I$ (resp. $G_r(M, x, y; M', x', y') \in II$) le fait que le jeu $G_r(M, x, y; M', x', y')$ est gagnant pour le joueur I (resp. gagnant pour le joueur II).

Nous allons montrer qu'il existe un lien entre l'existence d'une partie gagnante en r tours pour le joueur II (c.-à-d. l'impossibilité de distinguer les deux modèles au moyen de r tours) et l'indistinguabilité des deux modèles pour les formules de la logique $PLTL$. Néanmoins, plutôt que d'utiliser pour cette preuve la logique $PLTL$, nous allons utiliser la logique $L(\mathsf{F}_{\gg}, \mathsf{X}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}^{-1})$ où $\mathsf{F}_{\gg} = \mathsf{X}\,\mathsf{F}_s$ et $\mathsf{F}^{-1}_{\ll} = \mathsf{X}^{-1}\,\mathsf{F}^{-1}_s$. On a clairement $L(\mathsf{F}_{\gg}, \mathsf{X}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}^{-1}) \equiv PLTL$, mais l'emploi de cette logique permet de simplifier la preuve du théorème suivant:

Théorème 4.2.6 *Soient M et M' deux modèles linéaires, $x, y \in M$, $x', y' \in M'$, et $r \geq 0$. On a:*

$$G_r(M, x, y; M', x', y') \in II \quad \Leftrightarrow \quad \begin{cases} x \equiv_{L,r} x' \\ y \equiv_{L,r} y' \\ x, y \sim x', y' \end{cases}$$

où L dénote $L(\mathsf{F}_{\gg}, \mathsf{X}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}^{-1})$ et $q \equiv_{L,r} q'$ énonce que q et q' vérifient les mêmes formules de L ayant une hauteur d'opérateurs temporels d'au plus r.

Preuve Montrons $\Rightarrow$. On procède par induction sur r. On note $||\phi||$ la hauteur d'opérateurs temporels de ϕ. De plus, pour représenter le déplacement du jeton bleu (resp. rouge) de M de la position z vers z', on utilise la notation suivante: $M, z \rightsquigarrow_b z'$ (resp. $M, z \rightsquigarrow_r z'$).

- $r = 0$: on a donc: $x, y \sim x', y'$ ce qui entraîne $x \equiv_{L,0} x'$ et $y \equiv_{L,0} y'$.
- $r = r'+1$: Donc $G_{r'+1}(M, x, y, M', x', y') \in II$. Nous allons montrer que toute formule de la forme $\mathsf{F}_{\gg}\,\phi$, $\mathsf{F}^{-1}_{\ll}\,\phi$, $\mathsf{X}\,\phi$ ou $\mathsf{X}^{-1}\,\phi$ (avec $||\phi|| \leq r'$) vraie en x (resp. y, x', y') est vraie en x' (resp. y', x, y).

  Supposons $x \models \mathsf{F}_{\gg}\,\phi$ avec $||\phi|| \leq r'$. Alors il existe $k > 1$ tel que $x + k \models \phi$. Si I joue $M, y \rightsquigarrow_r x + k$, alors il existe un coup $M', y' \rightsquigarrow_r x' + k'$ (avec

$k' > 1$) pour II tel que $G_{r'}(M, x, x+k; M', x', x'+k') \in II$. Et par h.i. on a: $x + k \equiv_{L,r'} x' + k'$. D'où $x' \models \mathsf{F}_{\gg} \phi$.

Supposons $x \models \mathsf{X}\phi$. Par hypothèse, on sait que $G_{r'}(M, x, x+1; M', y, y+1) \in II$. D'où $x + 1 \equiv_{L,r'} x' + 1$. On obtient donc: $x' \models \mathsf{X}\,\phi$.

On fait de même pour les autres cas ($\mathsf{F}^{-1}_{\ll}\phi$ et $\mathsf{X}^{-1}\,\phi$) et le traitement des opérateurs booléens est trivial.

Montrons $\Leftarrow$. On procède par induction sur r:

- $r = 0$: Comme $x, y \sim x', y'$, on a: $G_0(M, x, y; M', x', y') \in II$.
- $r = r' + 1$: Supposons que $x \equiv_{L,r'+1} x'$, $y \equiv_{L,r'+1} y'$, et $x, y \sim x', y'$. Nous devons montrer que $G_{r'+1}(M, x, y; M', x', y') \in II$. Supposons que I joue $M : x \rightsquigarrow_b z$. Si z est y (resp. $y-1$ ou $y+1$), la réponse de II doit être y' (resp. $y'-1$ ou $y'+1$) et par h.i. (car $y - 1 \equiv_{L,r'} y' - 1$ et $y + 1 \equiv_{L,r'} y' + 1$ d'après $y \equiv_{L,r'+1} y'$) on obtient des jeux gagnants en r' tours pour II.

  Considérons le cas où $z \neq y, y-1, y+1$. Supposons que pour tous les coups z' de II, on ait $G_{r'}(M, z, y; M', z', y') \in I$. Dans les cas où $z, y \sim z', y'$, on a alors $z \not\equiv_{L,r'} z'$. Supposons que $z > y + 1$, on a alors: $\forall z' > y' + 1$, $\exists \varphi_{z'}$ tq $||\varphi_{z'}|| \leq r'$ et $z \models \varphi_{z'}$ et $z' \not\models \varphi_{z'}$. L'ensemble des formules distinctes (modulo simplifications booléennes) $\varphi_{z'}$ est fini, soit N son cardinal et φ_i ($i = 1..N$) une énumération de cet ensemble. On définit alors la formule $\psi = \bigwedge_{i=1..N} \varphi_i$ et on a $||\psi|| \leq r'$.

  On a clairement : $z \models \psi$ alors que : $\forall z' > y' + 1 \in M'$, $z' \not\models \psi$, d'où on obtient $y \models \mathsf{F}_{\gg}\psi$ mais $y' \not\models \mathsf{F}_{\gg}\psi$. Cela contredit notre hypothèse initiale. On utilise $\mathsf{F}^{-1}_{\ll}$ dans le cas où $z < y$.

$\square$

Nous allons maintenant définir deux familles de modèles M_n et M'_n qui seront utilisées par la suite pour montrer qu'aucune formule de la logique $L(\mathsf{F}_{\gg}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}, \mathsf{X}^{-1})$ n'est équivalente à $a \wedge b\,\mathsf{U}\,c$.

Définition 4.2.7 *On définit deux séquences d'états S et T de la manière suivante (voir figure 4.2):*

$$S = \alpha_0\alpha_1\ldots\alpha_{2n+2} \quad tq \begin{cases} \alpha_0 \models a \wedge b \\ \alpha_i \models b \wedge \neg c \ \ (\forall i \mid \ 0 < i < 2n+2 \ \wedge \ i \neq n+1) \\ \alpha_{n+1} \models \neg b \wedge \neg c \\ \alpha_{2n+2} \models c \end{cases}$$

$$T = \beta_0\beta_1\ldots\beta_{n+1} \quad tq \begin{cases} \beta_0 \models a \wedge b \\ \beta_i \models b \wedge \neg c \ \forall\ 0 < i < n+1 \\ \beta_{n+1} \models c \end{cases}$$

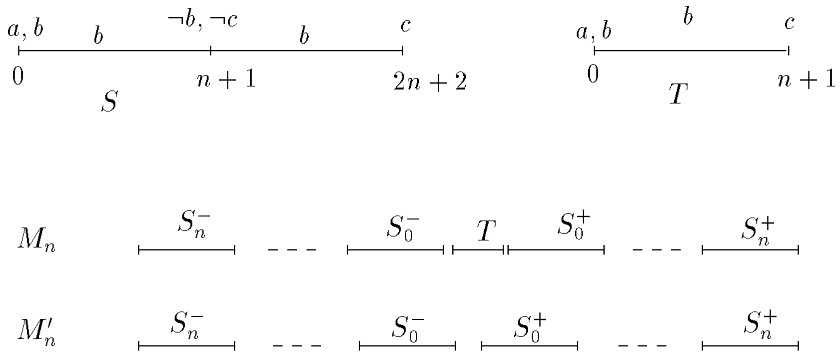


Figure 4.2: Modèles M_n et M'_n

Etant donné un entier n, on définit les modèles M_n et M'_n de la manière suivante: $M_n = S^n.S.T.S.S^n$ et $M'_n = S^n.S.S.S^n$.

Remarque Pour distinguer les différentes composantes de M_n et M'_n, on utilisera la notation suivante: $M_n = S_n^- \ldots S_0^-.T.S_0^+ \ldots S_n^+$ et $M'_n = S_n^- \ldots S_0^-.S_0^+ \ldots S_n^+$. De plus, $\alpha_i^{\varepsilon j}$ dénote l'état α_i $(0 \leq i \leq 2n+2)$ de S_j^ε $(\varepsilon \in \{+,-\}$ et $0 \leq j \leq n+1)$. □

Si on considère le jeu défini précédemment sur les modèles M_n et M'_n, on peut démontrer qu'il existe des jeux gagnants pour le joueur II. Nous avons notamment le théorème suivant:

Théorème 4.2.8 *Soit n un entier, pour $r \leq \frac{n}{2}$, on a:*

$$G_r(M_n, \alpha_0^{-n}, \alpha_0^{-n}; M'_n, \alpha_0^{-n}, \alpha_0^{-n}) \in II$$

Preuve Ce théorème correspond à un cas particulier du lemme A.0.4 démontré en annexe. □

Une conséquence directe de ce résultat est:

Corollaire 4.2.9 *$\mathsf{F}(a \wedge b \,\mathsf{U}\, c)$ est inexprimable initialement en PLTL*

Preuve Le théorème 4.2.8 implique que $M_n, \alpha_0^{-n} \equiv_{L,r} M'_n, \alpha_0^{-n}$ pour tout $r \leq \frac{n}{2}$. Mais on a $M_n, \alpha_0^{-n} \models \mathsf{F}\,(a \wedge b \,\mathsf{U}\, c)$ et $M'_n, \alpha_0^{-n} \not\models \mathsf{F}\,(a \wedge b \,\mathsf{U}\, c)$. On en déduit que la formule $\mathsf{F}\,(a \wedge b \,\mathsf{U}\, c)$ est inexprimable avec $L = L(\mathsf{F}_\gg, \mathsf{X}, \mathsf{F}_\ll^{-1}, \mathsf{X}^{-1})$ et comme $L \equiv PLTL$, cette formule est inexprimable avec $PLTL$. □

Concernant la logique LTL, on peut aussi remarquer que l'ajout des opérateurs X^{-1} et S permet d'atteindre l'expressivité de PTL: ceci est une conséquence directe du théorème 2.8.11 dû à Manna et Pnueli [MP90].

En conclusion, on peut noter que si l'introduction d'opérateurs du passé à PTL ou $PCTL^*_{HT}$ n'augmente pas leur expressivité au regard de l'équivalence initiale, il n'en va pas de même pour certaines logiques. Il est donc utile d'examiner en détail, pour chaque logique L, si $L \equiv_i L +$ Passé.

4.3 Logiques de temps arborescent et opérateurs S et X^{-1}

Nous allons, à présent, nous intéresser aux logiques du temps arborescent munies d'opérateurs du passé. Dans la littérature, le seul résultat concernant ces logiques est celui de Hafer et Thomas énonçant que $PCTL^*_{HT}$ se traduit initialement dans CTL^*. Avant de montrer comment on peut adapter ce théorème à notre définition de $PCTL^*$, nous allons tout d'abord définir une nouvelle notion de séparation. Ensuite nous étudierons l'introduction des opérateurs X^{-1} et S dans les différentes logiques que nous avons présentées au chapitre 2.

4.3.1 La séparation pour le temps arborescent

Le théorème de séparation 2.8.8 pour la logique $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ constitue un résultat important pour les logiques du temps linéaire. Il serait intéressant de disposer de résultats similaires pour le cas du temps arborescent. Cependant la différence de structure entre le temps linéaire et le temps arborescent empêche l'utilisation de la même notion de séparation. En général, une formule de temps arborescent ayant des opérateurs du passé n'est pas séparable. En effet dans le cas du temps linéaire, une formule ne fait référence qu'aux seuls états d'une exécution π et les opérateurs du passé peuvent ainsi décrire précisément le préfixe $\pi_{|n}$ de l'état courant. Mais dans le cas du temps arborescent, des opérateurs du futur dans la portée d'opérateur du passé peuvent faire référence aux *futurs possibles à partir d'un état du passé* auxquels les seuls opérateurs du passé ne peuvent pas faire référence (puisque ces états ne font pas partie de la séquence $\pi(0) \ldots \pi(n)$).

Ainsi la formule $\mathsf{F}^{-1}\mathsf{EF}\, a$ exprime-t-elle le fait qu'il existe un état dans le passé à partir duquel il était possible d'atteindre un état vérifiant a (voir figure 4.3)

Les opérateurs du passé X^{-1} et S expriment des propriétés sur les états du passé et ne peuvent pas, sans l'utilisation d'opérateurs du futur, caractériser des états qu'il aurait été possible d'atteindre depuis les états du passé. Dans le temps linéaire, cette restriction ne pose pas de problème: chaque état a un unique successeur, et la formule $\mathsf{F}^{-1}\mathsf{EF}\, a$ est équivalente à $\mathsf{F}^{-1}a \vee \mathsf{F}\, a$, car il n'y a plus de notion de choix ou de futur “possible”.

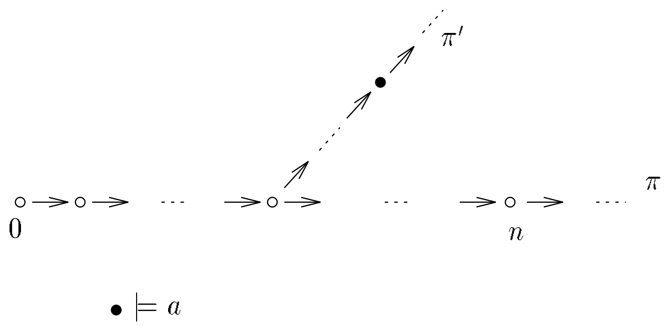


Figure 4.3: $\pi, n \models \mathsf{F}^{-1}\mathsf{EF}\ a$

De manière générale, les formules de temps arborescent ne sont donc pas séparables selon la définition utilisée pour le temps linéaire. En fait, nous allons définir une nouvelle notion de séparation:

Définition 4.3.1 *On dit qu'une formule f de $PCTL^*$ est* séparée *si et seulement si aucun opérateur du passé n'est dans la portée d'un opérateur du futur.*

Selon cette définition, dans une formule séparée, on autorise la présence d'opérateurs du futur dans la portée d'opérateurs du passé. Pour distinguer cette notion de séparation de celle utilisée pour le temps linéaire, nous appellerons cette dernière *séparation complète*, et on dira qu'une formule qui est une combinaison booléenne de formules pur-passé ou pur-futur, est *complètement séparée*. Notre définition est plus générale que celle utilisée pour le temps linéaire: toute formule complètement séparée est une formule séparée. Mais la réciproque est fausse: la formule $\mathsf{F}^{-1}\mathsf{EF}\ a$ est une formule séparée mais n'est pas complètement séparée. De plus, on a:

Proposition 4.3.2
- *Si une formule est séparée, alors toutes ses sous-formules le sont aussi.*
- *Si f et g sont des formules séparées, il en est de même pour $\neg f$, $f \wedge g$, $f\,\mathsf{S}\,g$, $\mathsf{X}^{-1} f$ et $\mathsf{E} f$.*

Preuve évidente. □

Lorsque, pour chaque formule f d'une logique L, il existe une formule de L, séparée, équivalente à f, alors on dit que la logique L est *séparable*. On désigne par L^{sep} le fragment de L composé des formules séparées. Ainsi L est séparable si et seulement si $L \equiv L^{sep}$.

Pour définir nos méthodes de séparation, nous avons besoin de *contextes*, c'est à dire des formules avec des variables qui représentent des sous-formules. Ainsi on notera $f[x]$ le contexte f ayant une variable x, et $f[g]$ représentera la formule obtenue, à partir de f, en remplaçant toutes les occurrences de x par g.

On étend les définitions précédentes aux contextes: on dira que $f[x_1, \ldots, x_n]$ est séparé lorsqu'aucun opérateur du passé n'est dans la portée d'opérateur du futur. On parlera de contexte pur-passé lorsque le contexte ne comportera aucun opérateur du futur, et de contexte pur-futur lorsqu'il ne comportera aucun opérateur du passé. La définition d'une formule séparée peut s'exprimer de la façon suivante: une formule f est séparée ssi f est de la forme suivante: $g[\varphi_1, \ldots, \varphi_n]$ où $g[x_1, \ldots, x_n]$ est un contexte pur-passé et où les φ_i sont des formules pur-futur.

De même, on utilisera le lemme de séparation de Gabbay pour les contextes de $PPTL$: tout contexte $f[x_1, \ldots, x_n]$ de $PPTL$ est équivalent à un contexte complètement séparé $\tilde{f}[x_1, \ldots, x_n]$.

Nous avons le théorème suivant:

Théorème 4.3.3 *Soit L une logique de la forme $B(C_1, \ldots, C_n)$, $L(C_1, \ldots, C_n)$ ou $B(L(C_1, \ldots, C_n))$ où $C_i \in \{ \mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}_s, \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \wedge, \neg\}$.*

Toute formule séparée de L est équivalente initialement à une formule pur-futur (de L).

Preuve Soit h une formule séparée. D'après la définition 4.3.1, h est équivalente à une formule de la forme $\varphi^-[g_1^+, \ldots, g_n^+]$, où φ^- est un contexte n-aire pur-passé et les g_j^+ sont pur-futur. On peut représenter h sous la forme suivante où les zones blanches correspondent à des sous-formules pur-futur et la zone grise à un contexte pur-passé:

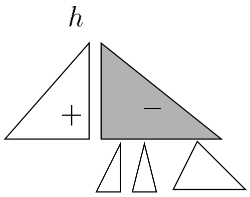


Figure 4.4: La formule h est séparée

Considérons la transformation Φ définie de manière inductive comme suit:

$$
\begin{array}{lcll}
\Phi(p) & = & p & \forall p \in Prop \\
\Phi(f \wedge g) & = & \Phi(f) \wedge \Phi(g) & \\
\Phi(\neg f) & = & \neg \Phi(f) & \\
\Phi(\mathsf{E}\, f) & = & \mathsf{E}\, \Phi(f) & \\
\Phi(f\, \mathsf{U}\, g) & = & f\, \mathsf{U}\, g & \\
\Phi(\mathsf{X}\, f) & = & \mathsf{X}\, f & \\
\Phi(f\, \mathsf{S}\, g) & = & \Phi(g) & \\
\Phi(\mathsf{X}^{-1}\, f) & = & \perp &
\end{array}
$$

Il est clair que pour toute formule h, $\Phi(f) \equiv_i h$: en effet, la seule action de la transformation Φ est de remplacer $f\,\mathsf{S}\,g$ par g et $\mathsf{X}^{-1} f$ par $\bot$ et cela seulement pour les opérateurs du passé qui se trouvent hors de la portée des opérateurs du futur (toute sous-formule ayant un X ou un U à la racine reste inchangée).

Si h est séparée, alors les seuls opérateurs du passé présents dans h sont hors de la portée d'opérateur du futur: $\Phi(h)$ est donc une formule pur-futur. $\square$

Ce théorème confirme que la notion de séparation que nous avons choisie est suffisante pour la traduction des logiques avec passé dans leur fragment pur-futur: si une logique L [1] est séparable, alors chacune de ces formules est initialement équivalente à une formule pur-futur de L.

Ainsi pourra-t-on déduire de tout résultat énonçant qu'une logique L (vérifiant les conditions du théorème 4.3.3) est séparable, le fait que L se traduise initialement dans L^+ où L^+ dénote le fragment pur-futur de L. Pour montrer les résultats du type $L \equiv_i L^+$, on essaiera donc toujours de montrer d'abord que L est séparable.

4.3.2 la logique $PCTL^*$

Pour montrer que $PCTL^*$ est équivalente initialement à CTL^*, nous allons considérer un lemme de séparation pour $PCTL^*$.

Lemme 4.3.4 *Toute formule de $PCTL^*$ est équivalente à une formule séparée (de $PCTL^*$).*

Preuve Nous suivons ici une démarche proche de celle utilisée dans [HT87].

Nous allons faire une preuve par induction sur la hauteur de quantificateurs E dans la formule f de $PCTL^*$ à séparer.

Supposons que f soit de la forme $\mathsf{E}\,g$ où g est une formule linéaire (sans quantificateur E). Alors, d'après le théorème de séparation de Gabbay, g est équivalente à une formule complètement séparée, c'est à dire:

$$\begin{array}{llll} & g & \equiv & \bigvee_i \left(g_i^- \wedge g_i^+\right) \qquad \text{où } g_i^+ \in L(\mathsf{U},\mathsf{X}) \text{ et } g_i^- \in L(\mathsf{S},\mathsf{X}^{-1}) \\ \text{donc} & f & \equiv & \mathsf{E}\left(\bigvee_i (g_i^- \wedge g_i^+)\right) \\ \text{d'où} & f & \equiv & \bigvee_i \left(\mathsf{E}\,(g_i^- \wedge g_i^+)\right) \end{array}$$

Les g_i^- étant des formules pur passé, on a: $f \equiv \bigvee_i \left(g_i^- \wedge \mathsf{E}\,g_i^+\right)$. De plus, les sous-formules $\mathsf{E}\,g_i^+$ sont des formules de CTL^*. D'autre part, les g_i^- étant des formules pur-passé, $\bigvee_i \left(g_i^- \wedge \mathsf{E}\,g_i^+\right)$ est une formule séparée.

[1] du même type que celle du théorème 4.3.3

Soit f une formule de la forme $\mathsf{E}\, g$ où g est une formule quelconque de $PCTL^*$. Alors, par h.i. on sait que f est équivalente à $\mathsf{E}\, \tilde{g}$ où toutes les sous-formules $\mathsf{E}\, h_i$ de $\tilde{g}$ sont des formules de CTL^*. On considère alors le contexte de $PCTL^*$ obtenu à partir de $\tilde{g}$, en remplaçant chaque $\mathsf{E}\, h_i$ par une variable x_i. On applique alors le théorème de séparation à ce contexte, et on obtient, après substitution des x_i par les $\mathsf{E}\, h_i$, une formule g' séparée (puisque les $\mathsf{E}\, h_i$ sont pur-futur) équivalente à $\tilde{g}$ (et donc à g). $\mathsf{E}\, g'$ est donc une formule séparée équivalente à f. $\square$

On en déduit immédiatement:

Théorème 4.3.5 $PCTL^* \equiv_i CTL^*$

Preuve On utilise le théorème 4.3.3. $\square$

Ce résultat qui est une variante de celui montré dans [HT87], est important puisqu'il énonce que CTL^* est suffisamment expressive pour exprimer initialement toutes les propriétés exprimable avec $PCTL^*$. Mais ce résultat ne résoud pas notre problème de recherche de logiques munies d'opérateurs du passé pouvant se traduire dans des fragments pur-futur ayant un algorithme de Model-Checking efficace, puisque CTL^* ne dispose pas d'un tel algorithme.

4.3.3 La logique $UB + \mathsf{X}^{-1}$, S

L'ajout des opérateurs X^{-1} et S à CTL permet de construire des formules qui ne sont pas traduisibles dans CTL. En fait, l'ajout de X^{-1} à la logique $B(\mathsf{F})$ et l'ajout de S à la logique UB^- suffisent pour sortir du pouvoir d'expression de $ECTL^+$.

Le simple fait de disposer des opérateurs EG et X^{-1} permet de construire des formules inexprimables en $ECTL^+$. Nous avons le théorème suivant:

Théorème 4.3.6 $B(\mathsf{F}, \mathsf{X}^{-1}) \not\leq_i ECTL^+$

Preuve La formule $\mathsf{EG}\,(a \vee \mathsf{X}\, a)$ de CTL^* est inexprimable en $ECTL^+$ [EH86]. Mais la formule de $B(\mathsf{F}, \mathsf{X}^{-1})$, $\mathsf{EG}\,(a \vee \mathsf{X}^{-1}\, a \vee \neg \mathsf{X}^{-1}\, \top)$, est équivalente initialement à $\mathsf{EG}\,(a \vee \mathsf{X}\, a)$. $\square$

On en déduit donc $UB + \mathsf{X}^{-1} \not\leq_i ECTL^+$ et à fortiori $CTL + \mathsf{X}^{-1} \not\leq_i ECTL^+$.

D'autre part, nous avons le lemme suivant:

Lemme 4.3.7

$$\begin{aligned}
\mathsf{E}\,(s \;\vee\; a\,\mathsf{U}\,b)\,\mathsf{U}\,r \quad &\equiv_i \\
& r \;\vee\; \mathsf{EF}\Big[\mathsf{EX}\,b \;\wedge\; a\,\mathsf{S}\big[r \;\wedge\; a \;\wedge\; \neg\mathsf{F}^{-1}\big(\neg a \;\wedge\; \neg b \;\wedge\; (\neg b)\,\mathsf{S}\,(\neg b \;\wedge\; \neg s)\big)\big]\Big] \\
& \vee\; \mathsf{EF}\Big[\mathsf{EX}\,(r \;\wedge\; b) \;\wedge\; \neg\mathsf{F}^{-1}\big(\neg a \;\wedge\; \neg b \;\wedge\; (\neg b)\,\mathsf{S}\,(\neg b \;\wedge\; \neg s)\big)\Big] \\
& \vee\; \mathsf{EF}\Big[\neg\mathsf{F}^{-1}\big(\neg a \;\wedge\; \neg b \;\wedge\; (\neg b)\,\mathsf{S}\,(\neg b \;\wedge\; \neg s)\big) \;\wedge\; \mathsf{EX}\,(r) \;\wedge\; \neg(\neg b)\,\mathsf{S}\,(\neg b \;\wedge\; \neg s)\Big]
\end{aligned}$$

Preuve Si $\mathsf{E}\,(s \;\vee\; a\,\mathsf{U}\,b)\,\mathsf{U}\,r$ est vrai dans un état donné, alors cela signifie que l'on peut atteindre un état vérifiant r et qu'avant, $(s \;\vee\; a\,\mathsf{U}\,b)$ a été vraie, c'est à dire qu'il n'existe pas d'état vérifiant $\neg s \;\wedge\; \neg(a\,\mathsf{U}\,b)$. Mais un état π, k vérifie $\neg s \;\wedge\; \neg(a\,\mathsf{U}\,b)$ si et seulement si il existe un état π, l avec $l \geq k$, tel que π, l vérifie $\neg a \;\wedge\; \neg b \;\wedge\; (\neg b)\,\mathsf{S}\,(\neg b \;\wedge\; \neg s)$.

Les termes de la disjonction représentent les différents cas selon que l'état vérifiant r vérifie ou b, $a \;\wedge\; \mathsf{EX}\,(\mathsf{E}\,a\,\mathsf{U}\,b)$ ou $\neg\mathsf{E}\,a\,\mathsf{U}\,b$. □

On en déduit le théorème suivant:

Théorème 4.3.8 $UB^- + \mathsf{S} \not\leq_i ECTL^+$

Preuve La preuve est basée sur le fait que $\mathsf{E}\,(s \vee a\,\mathsf{U}\,b)\,\mathsf{U}\,r$ est inexprimable avec $ECTL^+$ (lemme 2.8.3) et exprimable initialement avec $UB^- + \mathsf{S}$ 4.3.7. □

Intuitivement, ce résultat est dû au fait que les opérateurs U emboîtés font référence à la même séquence: mais en $ECTL^+$, entre deux opérateurs U emboîtés, il y a toujours un quantificateur E ou A, ce qui rend impossible l'expression de $\mathsf{E}\,(s \;\vee\; a\,\mathsf{U}\,b)\,\mathsf{U}\,r$. Mais l'opérateur S ne quantifie pas (il n'y a qu'un passé), et l'emboîtement de deux S permet ainsi de faire référence à des séquences au sein d'une séquence englobante et d'exprimer cette propriété.

Ces deux résultats (théorèmes 4.3.6 et 4.3.8) montrent malheureusement que l'introduction des opérateurs du passé dans une logique aussi peu expressive que UB, compromet les possibilités de traduction dans la logique $ECTL^+$. En conséquence, l'introduction des opérateurs X^{-1} ou S dans CTL aura le même effet et interdira la traduction dans $ECTL^+$. Avant de considérer d'autres façons d'introduire les opérateurs du passé dans les logiques du temps arborescent, il nous reste à étudier la logique $B(\mathsf{X}\,)$ qui est encore moins expressive que UB.

4.3.4 La logique $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$

Nous allons considérer la logique $B(\mathsf{X})$ augmentée des opérateurs X^{-1} et S. Les formules de cette logique sont donc construites à l'aide des opérateurs temporels EX, AX, S et X^{-1}. Nous avons le résultat suivant:

Lemme 4.3.9 *Toute formule de $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$ est équivalente à une formule séparée (de $B(\mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$).*

Preuve Soit f une formule de $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$. La preuve se fait par induction structurelle sur f. Les cas où f est a, $f_1 \wedge f_2$ ou $\neg f$ sont triviaux.

- $f = \mathsf{X}^{-1} f_1$ ou $f = f_1 \mathsf{S} f_2$. Par hypothèse d'induction, f_1 et f_2 sont équivalentes à des formules séparées f_1' et f_2': il est clair que $\mathsf{X}^{-1} f_1'$ et $f_1' \mathsf{S} f_2'$ sont donc séparées (cf. proposition 4.3.2).

- $f = \mathsf{EX}\, g$: Par h.i. on sait que g est équivalente à une formule séparée. On a donc: $g \equiv \bigvee_i (g_i^+ \wedge g_{i,1}^{-+} \wedge \ldots \wedge g_{i,n_i}^{-+})$ où les $g_{i,j}^{-+}$ sont des formules séparées ayant un X^{-1} ou un S à la racine. Nous allons transformer ces formules de manière à obtenir une formule séparée $\tilde{g}$ équivalente à g et telle que chaque opérateur S présent dans $\tilde{g}$ soit dans la portée d'un opérateur X^{-1}: nous allons utiliser l'équivalence suivante pour "déplier" les sous-formules $\alpha \mathsf{S} \beta$:

$$\alpha \mathsf{S} \beta \;\equiv\; \beta \vee \Big(\alpha \wedge \mathsf{X}^{-1} (\alpha \mathsf{S} \beta)\Big)$$

  Nous appliquons cette transformation jusqu'à ce que chaque opérateur *Since* soit dans la portée d'un *Previous*.

  Ayant obtenue la formule séparée $\tilde{g} = \bigvee_i (g_i^+ \wedge \tilde{g}_{i,1}^{-+} \wedge \ldots \wedge \tilde{g}_{i,n_i'}^{-+})$ où les $\tilde{g}_{i,j}^{-+}$ ont un X^{-1} à la racine, on peut regrouper les $\tilde{g}_{i,j}^{-+}$ de la manière suivante:

$$\begin{array}{lcl} \mathsf{X}^{-1} f \wedge \mathsf{X}^{-1} g & \equiv & \mathsf{X}^{-1} (f \wedge g) \\ \neg \mathsf{X}^{-1} f & \equiv & \neg \mathsf{X}^{-1} \top \vee \mathsf{X}^{-1} (\neg f) \end{array}$$

  On obtient finalement une formule g' équivalente à g de la forme:

$$\begin{array}{rcl} g' & = & \bigvee_i \Big(g_i'^{+} \wedge \mathsf{X}^{-1} g_i'^{-+}\Big) \\ \text{ou } g' & = & \bigvee_i \Big(g_i'^{+} \wedge \mathsf{X}^{-1} g_i'^{-+} \wedge \neg \mathsf{X}^{-1} \top\Big) \end{array}$$

  La présence des $\neg \mathsf{X}^{-1} \top$ dépend de la présence de sous-formule(s) $\neg \mathsf{X}^{-1} f$ dans la formule g. A partir de g', on obtient facilement une sous-formule séparée équivalente à $\mathsf{EX}\, g$ avec:

$$\begin{array}{lcl} \mathsf{EX}\,(\bigvee_i f_i) & \equiv & \bigvee_i \mathsf{EX}\, f_i \\ \mathsf{EX}\,(f \wedge \mathsf{X}^{-1}\, g) & \equiv & g \wedge \mathsf{EX}\, f \\ \mathsf{EX}\,(f \wedge \neg \mathsf{X}^{-1}\, \top) & \equiv & \bot \end{array}$$

A partir d'une sous-formule $f = \mathsf{EX}\, g$, nous avons donc construit une formule séparée équivalente.

□

On obtient donc le théorème suivant:

Théorème 4.3.10 $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S}) \equiv_i B(\mathsf{X})$

Preuve Application du théorème 4.3.3 et du lemme précédent. □

Ce résultat est donc positif puisqu'il énonce la possibilité de traduire dans la logique $B(\mathsf{X})$ les formules de $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$. Néanmoins son intérêt pour la spécification est limité en raison de la faible expressivité de la logique obtenue: on est loin du pouvoir expressif de CTL ou de $ECTL^+$.

Dans le même cadre, on peut aussi considérer la logique $B(L(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S}))$ constituer des formules de la forme $\mathsf{A}\, f$ où f est une formule de $L(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$. On obtient aisément le résultat suivant:

Théorème 4.3.11 $B(L(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})) \equiv_i B(L(\mathsf{X}))$

Preuve La logique $L(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$ est séparable (on peut procéder de la même manière que pour la logique $B(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})$ en simplifiant le dernier cas d'induction). On peut donc obtenir une formule séparée que l'on peut transformer en une formule de $L(\mathsf{X})$ équivalente initialement à f. On en déduit $B(L(\mathsf{X}, \mathsf{X}^{-1}, \mathsf{S})) \equiv_i B(L(\mathsf{X}))$. □

4.3.5 La logique $ECTL^+ + \mathsf{S} + \mathsf{X}^{-1}$

Avant de tirer un premier bilan concernant l'introduction des opérateurs du passé dans les logiques du temps arborescent, nous allons présenter un autre résultat intéressant du point de vue théorique qui montre une nouvelle fois le fort pouvoir expressif des opérateurs X^{-1} et S.

Nous avons vu que l'ajout des opérateurs X^{-1} et S à UB augmente sensiblement l'expressivité, au point de permettre l'énoncé de propriétés inexprimables avec $ECTL^+$. De ce fait, la logique obtenue à partir de $ECTL^+$ en ajoutant X^{-1} et S ne peut pas se traduire dans $ECTL^+$ et nous avons même le résultat suivant:

Théorème 4.3.12 $B(L(X, U, X^{-1}, S)) \prec_i ECTL^+ + X^{-1}, S$

Preuve Ce résultat utilise le théorème 2.8.11 dû à [MP90] qui énonce que toute formule de $PPTL$ (i.e. $L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})$) est équivalente initialement à une formule de la forme:

$$\bigwedge_{i=1..n} \Big(\mathsf{G}\,\mathsf{F}\, p_i \ \vee\ \mathsf{F}\,\mathsf{G}\, q_i\Big) \tag{4.44}$$

où p_i et q_i sont des formules pur-passé (i.e. de $L(\mathsf{X}^{-1}, \mathsf{S})$).

Toute formule de $B(L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S}))$ est de la forme: $\mathsf{A}\, f$ où f est une formule $PPTL$. On a donc:

$$\mathsf{A}\, f \ \equiv_i\ \mathsf{A}\Big[\bigwedge_{i=1..n} \Big(\mathsf{G}\,\mathsf{F}\, p_i \ \vee\ \mathsf{F}\,\mathsf{G}\, q_i\Big)\Big] \tag{4.45}$$

La formule de droite est une formule de $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$ puisque chaque p_i et q_i est pur-passé (et donc $p_i, q_i \in ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$), $\mathsf{G}\,\mathsf{F} = \overset{\infty}{\mathsf{F}}$ et $\mathsf{F}\,\mathsf{G} = \overset{\infty}{\mathsf{G}}$.

On a donc: $B(L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})) \preceq_i ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$. D'autre part on a $B(L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})) \equiv_i B(L(\mathsf{X}, \mathsf{U}))$ (conséquence de $PPTL \equiv_i PTL$) et $B(L(\mathsf{X}, \mathsf{U})) \bowtie ECTL^+$ (i.e. $ECTL^+ \not\preceq_i B(L(\mathsf{X}, \mathsf{U}))$), on en déduit $B(L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})) \prec_i ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$. □

Ce résultat est important car il souligne combien les opérateurs X^{-1} et S peuvent enrichir l'expressivité des logiques. En effet, la logique $B(L(\mathsf{U}, X, \mathsf{X}^{-1}, \mathsf{S}))$ (qui est équivalente initialement à $B(PTL)$) est très expressive. Parmi les logiques considérées ici, seules $PCTL^*$, CTL^* et $ECTL^+ + \mathsf{X}^{-1} + \mathsf{S}$ disposent d'une expressivité supérieure.

4.3.6 Premier bilan:

Les résultats précédents sont décrits par la figure 4.5.

Dans ce schéma, une flèche reliant une logique L à une logique L' signifie que $L' \preceq_i L'$. Les résultats que nous avons présentés et les trois conjectures suivantes permettent de conclure qu'aucune flèche ne peut être ajoutée dans le schéma (excepté par transitivité):

Conjecture 4.3.13

- $ECTL^+ + S, X^{-1} \not\preceq_i CTL^*$
- $UB^- + S \not\preceq_i UB$
- $UB + X^{-1} \not\preceq_i UB^+$

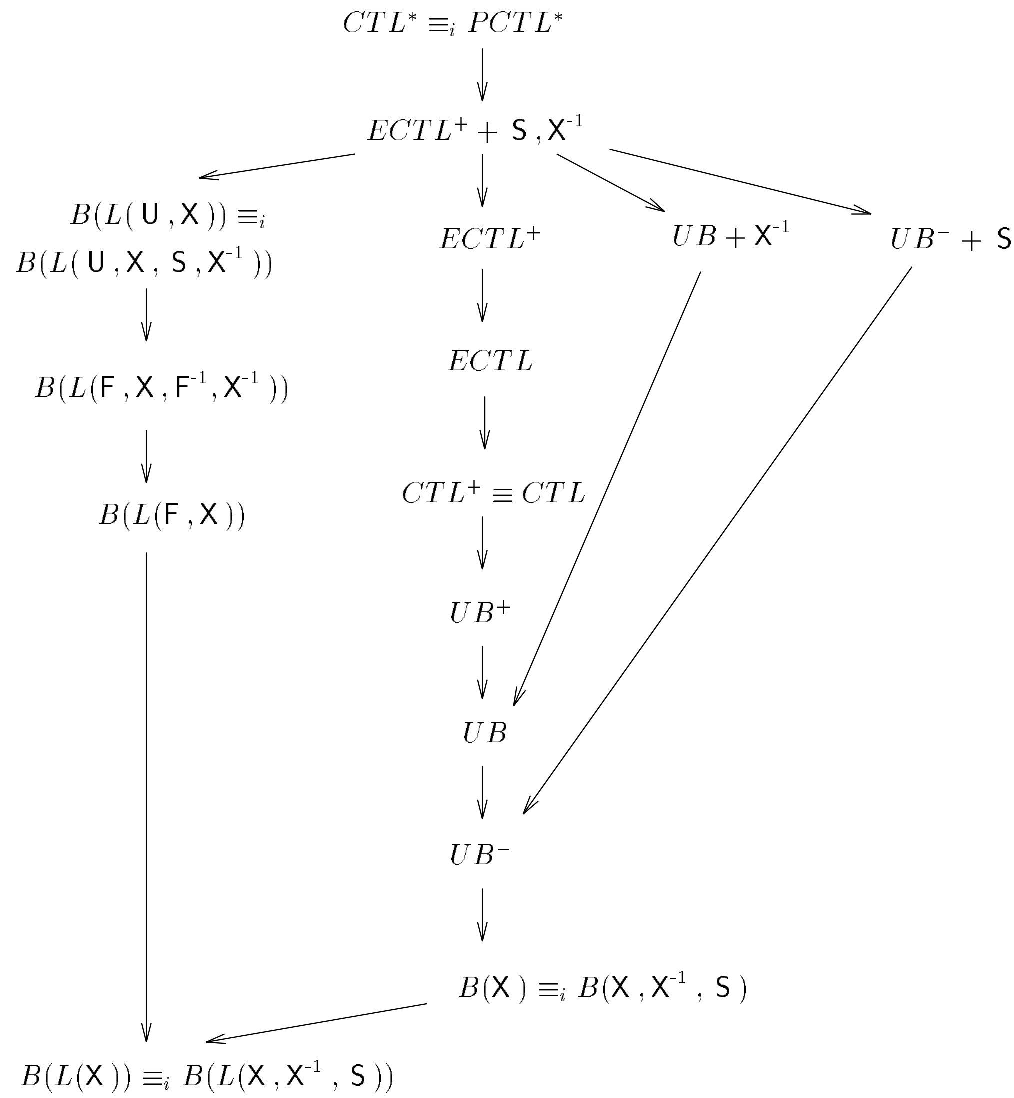


Figure 4.5: Les logiques de temps arborescent avec X^{-1} et S

Nous reviendrons au chapitre suivant sur la première conjecture, l'idée étant que la traduction de CTL^* nécessiterait de pouvoir traduire globalement toute formule de $B(L(\mathsf{X}, \mathsf{U}))$ dans $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$, ce qui n'est pas (à notre avis) possible. La seconde conjecture est basée sur l'idée que la formule $\mathsf{EG}\, a$ serait inexprimable en $UB^- + \mathsf{S}$ où l'on ne dispose que des opérateurs EF, EX, AG et S. La troisième conjecture repose sur l'idée que $E(\mathsf{F}\, a \wedge \mathsf{G}\, b)$ ne peut pas se traduire en $UB + \mathsf{X}^{-1}$.

Les résultats que nous venons de présenter concernant l'expressivité de $UB + \mathsf{X}^{-1}$ et $UB + \mathsf{S}$ montrent que l'ajout des opérateurs X^{-1} et S à la logique UB, et à fortiori à CTL, empêche la traduction dans les fragments de CTL^* ayant un algorithme de Model-Checking efficace (CTL, $ECTL$, $ECTL^+$).

Cela pourrait laisser penser qu'il n'est pas possible d'insérer des opérateurs du passé dans des logiques du temps arborescent sans perdre la possibilité de se ramener, après traduction, à des logiques comme CTL ou $ECTL^+$. Pourtant on peut suggérer au moins deux façons de contourner partiellement les problèmes posés par nos premiers résultats: d'une part, on peut s'intéresser à des opérateurs du passé "moins forts" comme F^{-1} ou F_s^{-1}, et d'autre part on peut chercher si des restrictions sur la place de S et X^{-1} dans les formules permettent de "bonnes traductions" (dans CTL ou $ECTL^+$). Il se trouve que ces deux approches aboutissent, et nous allons maintenant les présenter.

Si l'opérateur F^{-1} est plus faible que le *Since*, on peut noter qu'il est utile dans bien des cas, car il correspond à l'assertion "Il y a eu dans le passé ..." assez naturelle dans une spécification. En ce qui concerne le fait de chercher à insérer différemment X^{-1} et S pour permettre la traduction, cette démarche est motivée par la forme générale d'une spécification (comme celle de l'ascenseur du chapitre précédent) qui est constituée d'un ensemble de formules assez simples: dans ce cas, un emploi même limité d'opérateurs du passé peut s'avérer utile pour construire une spécification.

4.4 Logiques de temps arborescent avec opérateurs F_s^{-1},F^{-1}

Dans un premier temps, nous allons étudier l'insertion des opérateurs F_s^{-1} et F^{-1} dans les différentes logiques de temps arborescent présentées au chapitre 2.

Bien que les méthodes soient très proches, nous avons choisi de présenter à la fois les résultats concernant l'introduction de F^{-1} et ceux concernant l'introduction de F_s^{-1}. Ce choix est motivé par les intérêts respectifs de ces deux opérateurs.

L'opérateur F^{-1} est souvent utilisé dans la littérature et sa définition est assez naturelle dans une étude sur les logiques munies des opérateurs U et S.

L'opérateur F_s^{-1} a une expressivité plus riche et permet ainsi d'exprimer des propriétés importantes. Reprenons l'exemple du distributeur de billet: dans une spécification, on pourrait souhaiter exprimer que *à chaque fois que la carte est*

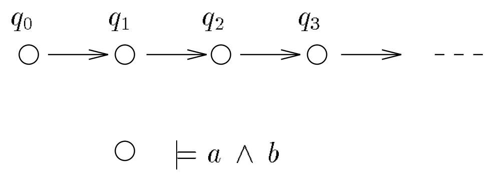


Figure 4.6: Modèle M

bloquée, alors, c'est qu'il y a eu trois erreurs de code précédemment. Cette propriété est exprimable directement avec $CTL + \mathsf{F}^{-1}_s$ de la manière suivante:

$$\mathsf{AG}\left[\mathit{blocage} \;\; \Rightarrow \;\; \mathsf{F}^{-1}_s\Big(\mathit{erreur} \wedge \mathsf{F}^{-1}_s(\mathit{erreur} \wedge \mathsf{F}^{-1}_s \mathit{erreur})\Big)\right]$$

Pourtant il n'existe pas de formule de $CTL + \mathsf{F}^{-1}$ équivalente globalement à cette formule de $CTL + \mathsf{F}^{-1}_s$. En fait, on a même:

Lemme 4.4.1 *Il n'existe pas de formule de* $CTL + \mathsf{F}^{-1}$ *équivalente globalement à* $\mathsf{AG}\,(a \;\Rightarrow\; \mathsf{F}^{-1}_s b)$

Preuve Considérons le modèle linéaire M où tous les états q_i sont étiquetés par les propositions atomiques a et b (cf. figure 4.6).

On peut facilement montrer, par induction structurelle sur les formules de $CTL+\mathsf{F}^{-1}$, que pour tout i et tout j, on a: $q_i \equiv_{CTL+\mathsf{F}^{-1}} q_j$.

Mais il est clair que pour tout $i > 1$, on a $q_i \models \mathsf{AG}\,(a \;\Rightarrow\; \mathsf{F}^{-1}_s b)$ mais $q_0 \not\models \mathsf{AG}\,(a \;\Rightarrow\; \mathsf{F}^{-1}_s b)$ (puisque $q_0 \not\models \mathsf{F}^{-1}_s b$). $\mathsf{AG}\,(a \;\Rightarrow\; \mathsf{F}^{-1}_s b)$ est donc inexprimable en $CTL + \mathsf{F}^{-1}$. $\square$

Puisque F^{-1} s'exprime facilement avec F^{-1}_s, on peut déduire que $CTL + \mathsf{F}^{-1} \prec_g CTL + \mathsf{F}^{-1}_s$. Avec le même modèle, on pourrait montrer que $\mathsf{AG}\,(a \;\Rightarrow\; \mathsf{F}^{-1}_s(b \wedge \mathsf{F}^{-1}_s(b \wedge \mathsf{F}^{-1}_s b)))$ ou $\neg\mathsf{F}^{-1}_s\top$ (qui permet de distinguer l'état initial) sont inexprimables en $CTL + \mathsf{F}^{-1}$.

Dans cette section, nous considérerons toujours les deux opérateurs F^{-1} et F^{-1}_s et nous chercherons à les insérer dans les différentes logiques du temps arborescent.

4.4.1 $UB + \mathsf{F}^{-1}, \mathsf{F}^{-1}_s$

Même si la logique UB est moins intéressante que CTL en tant que langage de spécification, nous allons, dans un premier temps, étudier l'effet en terme d'expressivité de l'ajout de l'opérateur F^{-1} à la logique UB. Ensuite nous nous intéresserons à la logique $CTL + \mathsf{F}^{-1}$. Nous avons le résultat suivant:

Théorème 4.4.2 $UB \prec_i UB + \mathsf{F}^{-1}$

Preuve La preuve de ce théorème est basée sur le fait que $\mathsf{E}\, a \,\mathsf{U}\, b$ est inexprimable avec UB^+ [EH86], tandis qu'il existe une formule de $UB + \mathsf{F}^{-1}$ équivalente initialement à $\mathsf{E}\, a \,\mathsf{U}\, b$:

$$\mathsf{E}\, a \,\mathsf{U}\, b \equiv_i b \vee \mathsf{EF}\left(\mathsf{EX}\, b \wedge \mathsf{G}^{-1} a\right)$$

$\square$

Nous avons donc le corollaire suivant:

Corollaire 4.4.3 $UB + \mathsf{F}^{-1}$ *n'est pas séparable.*

Preuve Si $UB + \mathsf{F}^{-1}$ était séparable, le théorème 4.3.3 entraînerait: $UB + \mathsf{F}^{-1} \equiv_i UB$. $\square$

Mais si l'expressivité de $UB + \mathsf{F}^{-1}$ dépasse celle de UB, elle reste en deça de l'expressivité de CTL:

Théorème 4.4.4 $UB + \mathsf{F}^{-1} \prec_i CTL$

Preuve Nous verrons ultérieurement la preuve de $UB + \mathsf{F}^{-1} \preceq_i CTL$: il s'agit d'un corollaire immédiat du théorème qui énonce que $CTL + \mathsf{F}^{-1}$ se traduit initialement dans CTL.

Considérons la direction $CTL \not\preceq_i UB + \mathsf{F}^{-1}$. Supposons qu' il existe une formule f de $UB + \mathsf{F}^{-1}$, équivalente initialement à $\mathsf{EF}\,(a \wedge \mathsf{E}\, b \,\mathsf{U}\, c)$, alors la formule $\tilde{f}$ de $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1})$ obtenue en supprimant les quantificateurs E, serait équivalente (initialement) à la formule de PTL $\mathsf{F}\,(a \wedge b \,\mathsf{U}\, c)$. D'après le corollaire 4.2.9, il n'existe pas une telle formule dans $L(\mathsf{F}, \mathsf{X}, \mathsf{F}^{-1}, \mathsf{X}^{-1})$. La formule $\mathsf{EF}\,(a \wedge \mathsf{E}\, b \,\mathsf{U}\, c)$ n'est donc équivalente initialement à aucune formule de $UB + \mathsf{F}^{-1}$. On a donc: $UB + \mathsf{F}^{-1} \prec_i CTL$. $\square$

En ce qui concerne l'opérateur F_s^{-1}, nous obtenons les mêmes résultats: $UB \prec_i UB + \mathsf{F}_s^{-1} \prec_i CTL$. Mais on a: $UB + \mathsf{F}^{-1} \prec_g UB + \mathsf{F}_s^{-1}$ (conséquence du lemme 4.4.1).

Concernant l'insertion de F^{-1} et F_s^{-1} dans les logiques UB^- et UB^+, nous avons clairement $UB^+ \not\prec_i UB^- + \mathsf{F}^{-1}$ puisque la formule $\mathsf{E}\, a \,\mathsf{U}\, b$, inexprimable en UB^+, est exprimable initialement en UB^- (lemme 4.4.2). On en conclut donc $UB^+ + \mathsf{F}^{-1} \succ_i UB^+$ et $UB^- + \mathsf{F}^{-1} \succ_i UB^-$. On a de même pour F_s^{-1}.

4.4.2 La logique $CTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$

Nous allons maintenant montrer que la logique CTL augmentée des opérateurs F^{-1} et F_s^{-1} peut se traduire (initialement) dans CTL. Nous avons le lemme suivant:

Lemme 4.4.5 *Toute formule de $CTL + \mathsf{F}^{-1}$ est équivalente à une formule séparée (de $CTL + \mathsf{F}^{-1}$).*

Preuve La preuve ressemble à celle du lemme de séparation de Gabbay. Nous allons procéder en plusieurs étapes que nous distinguerons par autant de lemmes.

Lemme 4.4.6 *Si $f[x]$ est un contexte CTL, alors $f[\mathsf{F}^{-1}x]$ est équivalent à un contexte séparé $\tilde{f}[x, \mathsf{F}^{-1}x]$ où $\tilde{f}[x, y]$ est un contexte CTL.*

Preuve On procède par induction structurelle sur $f[x]$. Si $f[x]$ est a, $f_1[x] \wedge f_2[x]$, ou $\neg f_1[x]$, on obtient facilement le résultat. Nous allons considérer les cas où $f[x]$ est de la forme $\mathsf{E}\,\varphi[x]\ \mathsf{U}\ \psi[x]$, $\mathsf{EG}\,\varphi[x]$ ou $\mathsf{EX}\,\varphi[x]$.

- $f[x] = \mathsf{E}\varphi[x]\,\mathsf{U}\,\psi[x]$: Par h.i. on sait que $\varphi[\mathsf{F}^{-1}x]$ et $\psi[\mathsf{F}^{-1}x]$ sont équivalents respectivement aux contextes séparés $\tilde{\varphi}[x, \mathsf{F}^{-1}x]$ et $\tilde{\psi}[x, \mathsf{F}^{-1}x]$ où $\tilde{\varphi}[x, y]$ et $\tilde{\psi}[x, y]$ sont des contextes CTL. Dans $\tilde{\varphi}$ et $\tilde{\psi}$, les occurrences de $\mathsf{F}^{-1}x$ ne peuvent apparaître que dans la portée d'opérateurs booléens. On a donc:

$$\begin{aligned} \varphi[x, \mathsf{F}^{-1}x] &\equiv \Big((\mathsf{F}^{-1}x \wedge \alpha) \vee (\neg\mathsf{F}^{-1}x \wedge \beta) \vee \gamma\Big) \\ \psi[x, \mathsf{F}^{-1}x] &\equiv \Big((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\Big) \end{aligned}$$

  Où les α, β, γ, α', β' et γ' sont des contextes CTL (ne contenant donc aucun F^{-1}). Pour obtenir ces sous-formules, on regroupe les occurrences de $\mathsf{F}^{-1}x$ et $\neg\mathsf{F}^{-1}x$. On a donc:

$$f[\mathsf{F}^{-1}x] \equiv \mathsf{E}\Big((\mathsf{F}^{-1}x\wedge\alpha)\vee(\neg\mathsf{F}^{-1}x\wedge\beta)\vee\gamma\Big)\,\mathsf{U}\Big((\mathsf{F}^{-1}x\wedge\alpha')\vee(\neg\mathsf{F}^{-1}x\wedge\beta')\vee\gamma'\Big) \tag{4.46}$$

  Nous utilisons l'équivalence suivante comme une règle de séparation pour extraire $\mathsf{F}^{-1}x$ de la portée de l'opérateur U:

$$\boxed{\begin{array}{l} \mathsf{E}\Big((\mathsf{F}^{-1}x \wedge \alpha) \vee (\neg\mathsf{F}^{-1}x \wedge \beta) \vee \gamma\Big)\,\mathsf{U}\Big((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\Big) \\ \qquad\qquad\quad \mathsf{F}^{-1}x \ \wedge\ \mathsf{E}\,(\alpha \vee \gamma)\,\mathsf{U}\,(\alpha' \vee \gamma') \\ \qquad \equiv \ \vee\ \neg\mathsf{F}^{-1}x \ \wedge\ \mathsf{E}\Big(\neg x \wedge (\beta \vee \gamma)\Big)\,\mathsf{U}\Big(x \wedge \mathsf{E}\,(\alpha \vee \gamma)\,\mathsf{U}\,(\alpha' \vee \gamma')\Big) \\ \qquad\qquad\quad \vee\ \neg\mathsf{F}^{-1}x \ \wedge\ \mathsf{E}\Big(\neg x \wedge (\beta \vee \gamma)\Big)\,\mathsf{U}\Big(\neg x \wedge (\beta' \vee \gamma')\Big) \end{array}} \tag{4.47}$$

La validité de cette équivalence s'explique de la manière suivante: on distingue trois cas selon que le contexte $\mathsf{F}^{-1}x$ (où x est fixé) est vrai à l'état courant (et donc vrai pour tous les états au-delà), selon que $\mathsf{F}^{-1}x$ est faux à l'état courant et vrai à l'état vérifiant la partie droite du *Until* (dans ce cas on passe par un état vérifiant x), et enfin selon que $\mathsf{F}^{-1}x$ est faux à l'état courant et faux à l'état vérifiant la partie droite du *Until*.

On remarque ainsi que la formule de droite n'a pas de sous-formule $\mathsf{F}^{-1}x$ dans la portée d'opérateur du futur. On pourrait aussi considérer plusieurs règles plus simples (ne considérant qu'un $\mathsf{F}^{-1}x$ dans la partie droite) et utiliser la distributivité de la partie droite du E_U pour la disjonction (voir [LS94]).

- $f[x] = \mathsf{EG}\,\varphi[x]$: De manière identique (par h.i. et par regroupement des occurrences de $\mathsf{F}^{-1}x$) , on a:

$$f[\mathsf{F}^{-1}x] \equiv \mathsf{EG}\left((\mathsf{F}^{-1}x \,\wedge\, \alpha) \,\vee\, (\neg\mathsf{F}^{-1}x \,\wedge\, \beta) \,\vee\, \gamma\right) \tag{4.48}$$

On peut alors utiliser l'équivalence suivante comme règle de séparation pour extraire $\mathsf{F}^{-1}x$ hors de la portée de l'opérateur EG :

$$\boxed{\begin{array}{ll} \mathsf{EG}\left((\mathsf{F}^{-1}x \,\wedge\, \alpha) \,\vee\, (\neg\mathsf{F}^{-1}x \,\wedge\, \beta) \,\vee\, \gamma\right) & \\ & \mathsf{F}^{-1}x \,\wedge\, \mathsf{EG}\,(\alpha \vee \gamma) \\ \equiv & \vee\; \neg\mathsf{F}^{-1}x \,\wedge\, \mathsf{E}\left(\neg x \wedge (\beta \vee \gamma)\right)\mathsf{U}\left(x \wedge \mathsf{EG}\,(\alpha \vee \gamma)\right) \\ & \vee\; \neg\mathsf{F}^{-1}x \,\wedge\, \mathsf{EG}\left(\neg x \wedge (\beta \vee \gamma)\right) \end{array}} \tag{4.49}$$

- $f[x] = \mathsf{EX}\,\varphi[x]$: comme pour les deux cas précédents, on considère le contexte $\mathsf{EX}\left(\tilde{\varphi}[x, \mathsf{F}^{-1}x]\right)$ équivalent à $f[\mathsf{F}^{-1}x]$ où $\tilde{\varphi}[x, \mathsf{F}^{-1}x]$ est un contexte séparé et $\tilde{\varphi}[x, y]$ un contexte CTL. Avec la distributivité de EX pour la disjonction, il suffit de considérer les équivalences suivantes que l'on utilisera comme règles de séparation pour extraire les F^{-1} de la portée de l'opérateur EX :

$$\boxed{\begin{array}{lll} \mathsf{EX}\,(\alpha \wedge \mathsf{F}^{-1}x) & \equiv & \mathsf{EX}\,(\alpha \wedge x) \;\vee\; \mathsf{F}^{-1}x \wedge \mathsf{EX}\,\alpha \\ \mathsf{EX}\,(\alpha \wedge \neg\mathsf{F}^{-1}x) & \equiv & \neg\mathsf{F}^{-1}x \wedge \mathsf{EX}\,(\alpha \wedge \neg x) \end{array}} \tag{4.50}$$

Dans tous les cas, nous avons pu construire un contexte séparé $\tilde{f}[x, \mathsf{F}^{-1}x]$ et tel que $\tilde{f}[x, y]$ soit un contexte CTL. □

Lemme 4.4.7 *Si $f[x_1, \ldots, x_n]$ est un contexte CTL, alors $f[\mathsf{F}^{-1}x_1, \ldots, \mathsf{F}^{-1}x_n]$ est équivalent à un contexte séparé $\tilde{f}[x_1, \mathsf{F}^{-1}x_1, \ldots, x_n, \mathsf{F}^{-1}x_n]$ où $\tilde{f}[x_1, y_1, \ldots, x_n, y_n]$ est un contexte CTL.*

Preuve La preuve se fait par induction sur n. Le lemme 4.4.6 correspond au cas $n = 1$.

Pour $n = n' + 1$, on applique le lemme 4.4.6 au contexte $f[\mathsf{F}^{-1}x_1, x_2, \ldots, x_n]$ et on obtient ainsi un contexte séparé équivalent, $\tilde{f}[x_1, \mathsf{F}^{-1}x_1, x_2, \ldots, x_n]$ où $\tilde{f}[x_1, x_1', x_2, \ldots x_n]$ est un contexte CTL. Donc $\tilde{f}[x_1, \mathsf{F}^{-1}x_1, x_2, \ldots x_n]$ est une combinaison booléenne de $\mathsf{F}^{-1}x_1$ et de contexte CTL de la forme $g[x_1, \ldots x_n]$. Chaque contexte $g[x_1, \mathsf{F}^{-1}x_2, \ldots, \mathsf{F}^{-1}x_n]$ est séparable par h.i. $\square$

Lemme 4.4.8 *Si $f[x_1, \ldots, x_n]$ est un contexte CTL et $\varphi_1^-, \ldots, \varphi_n^-$ sont des formules pur-passé de $CTL + \mathsf{F}^{-1}$, alors $f[\varphi_1^-, \ldots, \varphi_n^-]$ est équivalente à une formule séparée de $CTL + \mathsf{F}^{-1}$.*

Preuve La preuve se fait par induction sur la hauteur de F^{-1} (le nombre maximum de F^{-1} emboîtés) dans $f[\varphi_1^-, \ldots, \varphi_n^-]$. Il suffit de constater que cette formule peut être vue comme $g[\mathsf{F}^{-1}\psi_1, \ldots, \mathsf{F}^{-1}\psi_m]$ où g est un contexte CTL et où chaque ψ_i est une formule pur-passé ayant une hauteur de F^{-1} inférieure à celle de φ_i^- (puisque chaque ψ_j est une sous-formule d'un φ_l^- apparaissant dans la portée d'un F^{-1}).

Par le lemme précédent, on sait construire un contexte séparé $\tilde{g}[x_1, \mathsf{F}^{-1}x_1, \ldots, x_m, \mathsf{F}^{-1}x_m]$ où $\tilde{g}[y_1, z_1, \ldots, y_m, z_m]$ est un contexte CTL. On sait que les z_i ne sont pas dans la portée d'opérateur temporel, et on peut donc utiliser l'hypothèse d'induction pour séparer les sous-formules de $\tilde{g}[\psi_1, \mathsf{F}^{-1}\psi_1, \ldots, \psi_m, \mathsf{F}^{-1}\psi_m]$ qui sont de la forme $h[\psi_1, \ldots, \psi_m]$ avec $h[y_1, \ldots, y_m]$ contexte CTL. $\square$

Lemme 4.4.9 *Si $f[x_1, \ldots, x_n]$ est un contexte CTL et $\psi_1, \ldots, \psi_n$ sont des formules séparées de $CTL + \mathsf{F}^{-1}$, alors $f[\psi_1, \ldots, \psi_n]$ est équivalente à une formule séparée de $CTL + \mathsf{F}^{-1}$.*

Preuve Chaque ψ_i est de la forme $g_i[\varphi_{i,1}, \ldots, \varphi_{i,m_i}]$ où $g[x_1, \ldots, x_{m_i}]$ est un contexte pur-passé de $CTL + \mathsf{F}^{-1}$ et chaque $\varphi_{i,j}$ est une formule CTL. Il suffit alors d'appliquer le lemme précédent au contexte $f[g_1[x_{1,1}, \ldots, x_{1,m_1}], \ldots, g_n[x_{n,1}, \ldots, x_{n,m_n}]]$ pour obtenir un contexte séparé équivalent $\tilde{f}[x_{1,1}, \ldots, x_{n,m_n}]$. On a bien $f[\psi_1, \ldots, \psi_n] \equiv \tilde{f}[\varphi_{1,1}, \ldots, \varphi_{n,m_n}]$ et comme les $\varphi_{i,j}$ sont des formules pur-futur, la formule $\tilde{f}[\varphi_{1,1}, \ldots, \varphi_{n,m_n}]$ est séparée. $\square$

Ce dernier lemme nous permet de prouver facilement le lemme de séparation pour toute formule f de $CTL + \mathsf{F}^{-1}$ par induction structurelle sur f:

- Si f est de la forme a, $g \wedge h$, $\neg g$, $\mathsf{E}\, g\ \mathsf{U}\ h$ ou $\mathsf{A}\, f\ \mathsf{U}\ g$, alors le dernier lemme et l'hypothèse d'induction permettent de conclure que f est séparable.

- Si f est de la forme $\mathsf{F}^{-1}g$, alors comme g est équivalente à une formule $\tilde{g}$ séparée (par h.i.), on a $f \equiv \mathsf{F}^{-1}\tilde{g}$ et $\mathsf{F}^{-1}\tilde{g}$ est séparée.

On obtient ainsi la preuve du lemme 4.4.5. $\square$

Dans la suite nous réutiliserons ce schéma de preuve en l'adaptant à d'autres logiques, néanmoins nous ne détaillerons plus les différentes étapes (lemmes 4.4.6, 4.4.7, 4.4.8 et 4.4.9), nous nous contenterons de donner les règles de séparation nécessaires pour la logique considérée en s'assurant que celles-ci remplissent bien les conditions requises à l'application de la première étape (lemme 4.4.6).

On obtient le résultat suivant à partir du lemme 4.4.5 et du théorème 4.3.3:

Théorème 4.4.10 *$CTL + \mathsf{F}^{-1} \equiv_i CTL$*

On obtient de manière équivalente:

Lemme 4.4.11 *Toute formule de $CTL + \mathsf{F}_s^{-1}$ est équivalente à une formule séparée (de $CTL + \mathsf{F}_s^{-1}$).*

Preuve La preuve se fait de manière semblable à celle du lemme énonçant la séparation de $CTL + \mathsf{F}^{-1}$. Il suffit de donner les équivalences qui seront utilisée comme règles de séparation pour F_s^{-1}:

$$
\begin{array}{l}
\mathsf{E}\left((\mathsf{F}_s^{-1}x \wedge \alpha) \vee (\neg\mathsf{F}_s^{-1}x \wedge \beta) \vee \gamma\right) \mathsf{U} \left((\mathsf{F}_s^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}_s^{-1}x \wedge \beta') \vee \gamma'\right) \\
\quad \equiv \begin{array}[t]{l}
\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{E}\,(\alpha \vee \gamma)\,\mathsf{U}\,(\alpha' \vee \gamma') \\
\vee\; \neg\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{E}\left(\neg x \wedge (\beta \vee \gamma)\right) \mathsf{U} \\
\qquad\qquad \left[x \wedge (\beta \vee \gamma) \wedge \mathsf{EX}\left(\mathsf{E}\,(\alpha \vee \gamma)\,\mathsf{U}\,(\alpha' \vee \gamma')\right)\right] \\
\vee\; \neg\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{E}\left(\neg x \wedge (\beta \vee \gamma)\right) \mathsf{U} \left(\beta' \vee \gamma'\right)
\end{array}
\end{array}
$$

$$
\begin{array}{l}
\mathsf{EG}\left((\mathsf{F}_s^{-1}x \wedge \alpha) \vee (\neg\mathsf{F}_s^{-1}x \wedge \beta) \vee \gamma\right) \\
\quad \equiv \begin{array}[t]{l}
\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{EG}\,(\alpha \vee \gamma) \\
\vee\; \neg\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{E}\left(\neg x \wedge (\beta \vee \gamma)\right) \mathsf{U} \left(x \wedge (\beta \vee \gamma) \wedge \mathsf{EX}\,\mathsf{EG}\,(\alpha \vee \gamma)\right) \\
\vee\; \neg\mathsf{F}_s^{-1}x \;\wedge\; \mathsf{EG}\left(\neg x \wedge (\beta \vee \gamma)\right)
\end{array}
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{EX}\,(\alpha \wedge \mathsf{F}_s^{-1}x) & \equiv & \mathsf{EX}\,\alpha \;\wedge\; (x \vee \mathsf{F}_s^{-1}x) \\
\mathsf{EX}\,(\alpha \wedge \neg\mathsf{F}_s^{-1}x) & \equiv & \neg\mathsf{F}_s^{-1}x \wedge \neg x \wedge \mathsf{EX}\,\alpha
\end{array}
$$

Dans ces équivalences, $\mathsf{F}_s^{-1}x$ n'apparaît plus, dans les formules de gauche, dans la portée d'un opérateur $\mathsf{E\_U}$, EG ou EX: on peut donc facilement adapter la preuve obtenue pour la séparation de $CTL + \mathsf{F}^{-1}$ à $CTL + \mathsf{F}_s^{-1}$. $\square$

On a donc le théorème suivant:

Théorème 4.4.12 *$CTL + \mathsf{F}_s^{-1} \equiv_i CTL$*

Preuve Le lemme précédent et le théorème 4.3.3 permettent de conclure que $CTL + \mathsf{F}_s^{-1} \equiv_i CTL$. □

Comme $\mathsf{F}^{-1} f \equiv f \vee \mathsf{F}_s^{-1} f$, on a aussi:

Théorème 4.4.13 $CTL + \mathsf{F}_s^{-1}, \mathsf{F}^{-1} \equiv_i CTL$

4.4.3 La logique $CTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$

Nous allons maintenant considérer la logique $CTL^+ + \mathsf{F}^{-1}$, c'est à dire la logique où l'on autorise dans la portée des quantificateurs E, les combinaisons booléennes de formules de chemin $\alpha \,\mathsf{U}\, \beta$, $\mathsf{X}\, \alpha$ et $\mathsf{F}^{-1}\alpha$ (avec α et β des formules de $CTL^+ + \mathsf{F}^{-1}$). Le résultat précédent concernant CTL peut être étendu à la logique CTL^+; pour cela, nous allons tout d'abord montrer que la logique $CTL^+ + \mathsf{F}^{-1}$ se traduit dans $CTL + \mathsf{F}^{-1}$ et en conséquence se traduit initialement dans CTL.

Théorème 4.4.14 $CTL^+ + \mathsf{F}^{-1} \equiv CTL + \mathsf{F}^{-1}$

Preuve Considérons une formule h de $CTL^+ + \mathsf{F}^{-1}$. Nous allons faire une preuve par induction sur la hauteur de quantificateurs E dans h. Si h ne contient aucun quantificateur E, alors h est aussi une formule de $CTL + \mathsf{F}^{-1}$.

Puisque la disjonction est distributive pour le quantificateur E, il suffit de considérer une formule h de la forme suivante:

$$h = \mathsf{E}\left(\bigwedge_i (f_i \,\mathsf{U}\, g_i) \;\wedge\; \bigwedge_i \neg(f_i' \,\mathsf{U}\, g_i') \;\wedge\; \bigwedge_i \mathsf{X}\, h_i \;\wedge\; \bigwedge_i \neg\mathsf{X}\, h_i' \;\wedge\; \bigwedge_i \mathsf{F}^{-1} k_i \;\wedge\; \bigwedge_i \neg\mathsf{F}^{-1} k_i'\right)$$

De plus, par h.i., on peut supposer que les sous-formules f_i, g_i,...sont des formules de $CTL + \mathsf{F}^{-1}$. Comme $\mathsf{E}\,(\varphi \wedge \varphi^-) \equiv \varphi^- \wedge \mathsf{E}\,\varphi$ si φ^- est une formule d'histoire, nous avons l'équivalence suivante:

$$h \equiv \mathsf{E}\left[\bigwedge_i (f_i \,\mathsf{U}\, g_i) \;\wedge\; \bigwedge_i \neg(f_i' \,\mathsf{U}\, g_i') \;\wedge\; \bigwedge_i \mathsf{X}\, h_i \;\wedge\; \bigwedge_i \neg\mathsf{X}\, h_i'\right] \;\wedge\; \bigwedge_i \mathsf{F}^{-1} k_i \;\wedge\; \bigwedge_i \neg\mathsf{F}^{-1} k_i'$$

On utilise alors les mêmes méthodes que celles utilisées dans [EH85] pour montrer $CTL^+ \equiv CTL$:

$$\mathsf{E}\left(\bigwedge_i (f_i \,\mathsf{U}\, g_i) \;\wedge\; \bigwedge_i \neg(f_i' \,\mathsf{U}\, g_i') \;\wedge\; \bigwedge_i \mathsf{X}\, h_i \;\wedge\; \bigwedge_i \neg\mathsf{X}\, h_i'\right) \;\equiv$$

$$\equiv \mathsf{E}\,(\bigwedge_{i=1\ldots n} \varphi_i \,\mathsf{U}\, \psi_i \;\wedge\; \mathsf{G}\,\alpha \;\wedge\; \mathsf{X}\,\alpha')$$

$$\equiv \bigwedge_{J \subseteq \{1,\ldots,n\}} \left[\bigwedge_{i \in J} \psi_i \;\wedge\; \bigwedge_{i \notin J} \varphi_i \;\wedge\; \alpha \;\wedge\; \mathsf{EX}\left[\alpha' \;\wedge\; \mathsf{E}\left(\bigwedge_{i \notin J} \varphi_i \,\mathsf{U}\, \psi_i \;\wedge\; \mathsf{G}\,\alpha\right)\right]\right]$$

De plus, on a :

$$\mathsf{E}\Big(\bigwedge_{i=1,\ldots,n} a_i \,\mathsf{U}\, b_i \;\wedge\; \mathsf{G}\, c\Big) \;\equiv$$
$$\bigvee_{\pi\in Permut(n)} \mathsf{E}\,\Big(\bigwedge_i a_i \;\wedge\; c\Big)\,\mathsf{U}\,\Big[b_{\pi(1)} \;\wedge\; \mathsf{E}\,(\bigwedge_{i\neq\pi(1)} a_i \;\wedge\; c)\,\mathsf{U}\,\Big(\ldots \mathsf{E}\,(a_{\pi(n)} \;\wedge\; c)\,\mathsf{U}\,(b_{\pi(n)} \;\wedge\; \mathsf{EG}\, c)\Big)\Big]$$

où $Permut(n)$ est l'ensemble des permutations sur $1,\ldots,n$

Nous avons donc pu construire une formule équivalente à h et telle que tous les opérateurs X et U soient dans la portée directe d'un quantificateur E : h est donc équivalente à une formule de $CTL + \mathsf{F}^{-1}$. □

Lemme 4.4.15 *Toute formule de $CTL^{+}{+}\mathsf{F}^{-1}$ est équivalente à une formule séparée de $CTL + \mathsf{F}^{-1}$.*

Preuve Directe à partir du théorème 4.4.14 et du lemme 4.4.5. □

En conséquence, on obtient le théorème suivant:

Théorème 4.4.16 $CTL^{+} + \mathsf{F}^{-1} \equiv_i CTL$

De la même manière que le théorème 4.4.14, on obtient:

Théorème 4.4.17 $CTL^{+} + \mathsf{F}_s^{-1} \;\equiv\; CTL + \mathsf{F}_s^{-1}$

On a aussi:

Lemme 4.4.18 *Toute formule de $CTL^{+}{+}\mathsf{F}_s^{-1}$ est équivalente à une formule séparée de $CTL + \mathsf{F}_s^{-1}$.*

Comme pour $CTL + \mathsf{F}_s^{-1}$, on peut déduire les deux théorèmes suivants:

Théorème 4.4.19 $CTL^{+} + \mathsf{F}_s^{-1} \equiv_i CTL$

Théorème 4.4.20 $CTL^{+} + \mathsf{F}_s^{-1}, \mathsf{F}^{-1} \equiv_i CTL$

4.4.4 La logique $ECTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$

Nous allons maintenant étudier la logique $ECTL$ augmentée de l'opérateur F^{-1}. Les formules de cette logique peuvent être vues comme étant construites à l'aide des opérateurs $\mathsf{E}\,\_\,\mathsf{U}$, $\mathsf{A}\,\_\,\mathsf{U}$, $\mathsf{E}\,\overset{\infty}{\mathsf{F}}$, $\mathsf{A}\,\overset{\infty}{\mathsf{F}}$ et F^{-1}. En fait, on sait qu'il suffit de considérer les seuls $\mathsf{E}\,\_\mathsf{U}$, AF , $\mathsf{E}\,\overset{\infty}{\mathsf{F}}$ et F^{-1} puisque d'une part, l'opérateur $\mathsf{A}\,\_\mathsf{U}$ peut être défini à partir de $\mathsf{E}\,\_\,\mathsf{U}$ et AF et d'autre part, $\mathsf{A}\,\overset{\infty}{\mathsf{F}}\ f \equiv \mathsf{AG\,AF}\ f$.

Nous allons voir qu'ajouter l'opérateur F^{-1} augmente l'expressivité de la logique $ECTL$: l'opérateur F^{-1} associé à $\mathsf{E}\,\overset{\infty}{\mathsf{F}}$ permet d'exprimer des propriétés pour lesquelles il n'existe aucun équivalent en $ECTL$. Ainsi avons-nous le théorème suivant:

Théorème 4.4.21 $ECTL \ \prec_i \ ECTL + \mathsf{F}^{-1}$

Preuve La preuve de ce théorème est basée sur le fait qu'il n'existe pas de formule de $ECTL$ équivalente à la formule de $ECTL^+$: $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a\ \wedge\ \mathsf{G}\,b)$ (lemme 2.8.4), alors que l'on a:

$$\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a\ \wedge\ \mathsf{G}\,b) \equiv_i \mathsf{E}\,\overset{\infty}{\mathsf{F}}\ (a\ \wedge\ \mathsf{G}^{-1}b)$$

Cette équivalence est valide puisque la formule de droite est vraie ssi il existe un chemin où il y a une infinité d'états qui vérifient a et pour chacun d'eux b est vrai depuis l'origine. □

L'ajout de l'opérateur F^{-1} permet donc d'augmenter l'expressivité de $ECTL$. Le fait de pouvoir exprimer initialement la formule $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \mathsf{G}\,b)$ permet à $ECTL + \mathsf{F}^{-1}$ d'exprimer initialement (à la différence de $ECTL$) une des propriétés d'équité [2] définies dans [Lam80]: $\mathsf{A}\,(\overset{\infty}{\mathsf{F}}\ a\ \Rightarrow\ \mathsf{F}\,b)$ ($\equiv \neg\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a\ \wedge\ \mathsf{G}\,\neg b)$) qui exprime que sur tous les chemins où a est vérifiée infiniment souvent, il existe un état où b est vraie.

Dans la hiérarchie des expressivités de [Eme90], on trouve au delà de $ECTL$, la logique $ECTL^+$. Nous allons maintenant constater que l'ajout de F^{-1} à $ECTL$ ne permet pas d'atteindre le pouvoir expressif de $ECTL^+$:

Théorème 4.4.22 $ECTL + \mathsf{F}^{-1} \ \prec_i \ ECTL^+$

Preuve Cette preuve se décompose en deux étapes: d'abord nous allons montrer que $ECTL + \mathsf{F}^{-1}$ se traduit initialement dans une logique du futur L^+, et ensuite nous montrerons $L^+ \prec_i ECTL^+$. Nous aurons donc:

$$ECTL + \mathsf{F}^{-1} \ \preceq_i \ L^+ \ \prec_i \ ECTL^+$$

La logique L^+ que nous utilisons est un fragment de $ECTL^+$ et contient $ECTL$:

[2] la propriété *strong eventual fairness.*

Définition 4.4.23 *Logique* L^+

$$L^+ \ni f, g ::= a \mid f \wedge g \mid \neg f \mid \mathsf{E} f \, \mathsf{U} g \mid \mathsf{A} f \, \mathsf{U} g \mid \mathsf{EX} f \mid \mathsf{E}(\overset{\infty}{\mathsf{F}} f \wedge \mathsf{G} g)$$

Nous avons donc: $ECTL \preceq_g L^+$ et $ECTL + \mathsf{F}^{-1} \preceq_g L^+ + \mathsf{F}^{-1}$. Nous allons maintenant montrer que la logique $L^+ + \mathsf{F}^{-1}$ est séparable:

Lemme 4.4.24 *Toute formule de $L^+ + \mathsf{F}^{-1}$ est équivalente à une formule séparée (de $L^+ + \mathsf{F}^{-1}$).*

Preuve Pour établir ce lemme, il suffit de donner une règle de séparation adaptée aux formules de la forme $\mathsf{E}(\overset{\infty}{\mathsf{F}} f \wedge \mathsf{G} g)$ de manière à étendre la preuve du lemme de séparation pour CTL. Nous avons l'équivalence suivante qui va nous permettre de limiter le nombre de cas à considérer pour la séparation de $L^+ + \mathsf{F}^{-1}$:

$$\mathsf{E}\left(\overset{\infty}{\mathsf{F}} (\alpha \vee \beta) \wedge \mathsf{G}\gamma\right) \equiv \mathsf{E}(\overset{\infty}{\mathsf{F}} \alpha \wedge \mathsf{G}\gamma) \vee \mathsf{E}(\overset{\infty}{\mathsf{F}} \beta \wedge \mathsf{G}\gamma)$$

En utilisant la même approche que celle du lemme 4.4.6, on essaie donc de trouver des formules séparées équivalentes aux deux formules suivantes:

$$\mathsf{E}\left[\overset{\infty}{\mathsf{F}} (\alpha \wedge \mathsf{F}^{-1}x) \wedge \mathsf{G}\left((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\right)\right]$$

$$\mathsf{E}\left[\overset{\infty}{\mathsf{F}} (\alpha \wedge \neg\mathsf{F}^{-1}x) \wedge \mathsf{G}\left((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\right)\right]$$

Nous avons les deux équivalences suivantes que nous pourrons utiliser comme règles de séparation pour l'opérateur $\mathsf{E}(\overset{\infty}{\mathsf{F}} f \wedge \mathsf{G} g)$:

$$\boxed{\begin{aligned}
&\mathsf{E}\left[\overset{\infty}{\mathsf{F}} (\alpha \wedge \mathsf{F}^{-1}x) \wedge \mathsf{G}\left((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\right)\right] \\
&\qquad \equiv \mathsf{F}^{-1}x \wedge \mathsf{E}\left(\overset{\infty}{\mathsf{F}} \alpha \wedge \mathsf{G}(\alpha' \vee \gamma')\right) \\
&\qquad\quad \vee \neg\mathsf{F}^{-1}x \wedge \mathsf{E}\left[\left(\neg x \wedge (\beta' \vee \gamma')\right) \mathsf{U} \left(x \wedge \mathsf{E}(\overset{\infty}{\mathsf{F}} \alpha \wedge \mathsf{G}(\alpha' \vee \gamma'))\right)\right] \\
\hline
&\mathsf{E}\left[\overset{\infty}{\mathsf{F}} (\alpha \wedge \neg\mathsf{F}^{-1}x) \wedge \mathsf{G}\left((\mathsf{F}^{-1}x \wedge \alpha') \vee (\neg\mathsf{F}^{-1}x \wedge \beta') \vee \gamma'\right)\right] \\
&\qquad \equiv \neg\mathsf{F}^{-1}x \wedge \mathsf{E}\left[\overset{\infty}{\mathsf{F}} \alpha \wedge \mathsf{G}\left(\neg x \wedge (\beta' \vee \gamma')\right)\right]
\end{aligned}} \tag{4.51}$$

La validité de ces deux équivalences se prouve de la même manière que celle des équivalences concernant les opérateurs de $CTL + \mathsf{F}^{-1}$.

Ces deux règles de séparation permettent ainsi d'étendre le lemme de séparation de CTL à la logique $L^+ + \mathsf{F}^{-1}$. □

On a donc clairement: $ECTL + \mathsf{F}^{-1} \preceq_i L^+$ (puisque $ECTL + \mathsf{F}^{-1} \subseteq L^+ + \mathsf{F}^{-1}$ et $L^+ + \mathsf{F}^{-1} \preceq_i L^+$).

Pour prouver $L^+ \prec_i ECTL^+$, nous allons montrer que la formule de $ECTL^+$, $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$ est inexprimable avec L^+:

Lemme 4.4.25 *La formule $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$ est inexprimable avec L^+.*

Preuve Pour montrer ce lemme, il suffit de reprendre la preuve proposée dans [EH86] qui établit que $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$ est inexprimable avec $ECTL$ et de l'étendre à L^+. On considère donc les mêmes modèles que [EH86]:

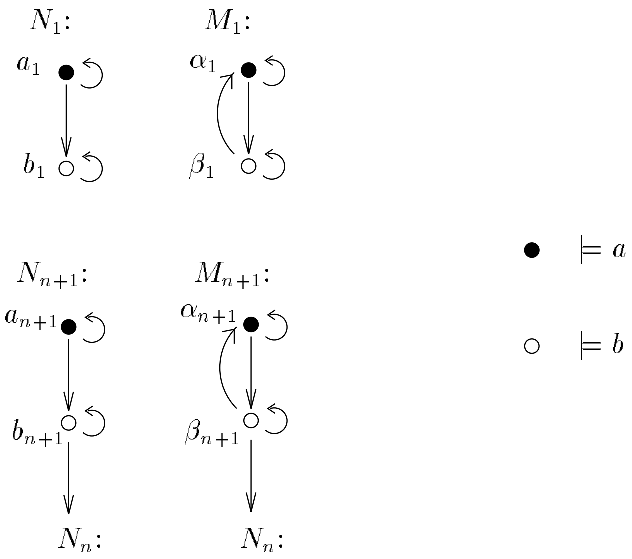


Figure 4.7: $\alpha_i \models \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$, et $a_i \not\models \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ a \wedge \overset{\infty}{\mathsf{F}}\ b)$

Nous avons les résultats suivants où f est une formule de L^+:

$$\begin{array}{lcl}(\exists j \geq |f|,\ a_j \models f) & \text{ssi} & (\forall k \geq |f|,\ a_k \models f)\\ (\exists j \geq |f|,\ b_j \models f) & \text{ssi} & (\forall k \geq |f|,\ b_k \models f)\end{array}$$

On montre ces résultats par induction sur la taille de f. On reprend la preuve de [EH86] pour les cas $f = a$, $f = \mathsf{EX}\, g$, $f = \mathsf{E}\, g\ \mathsf{U}\ h$ et $f = \mathsf{A}\, g\ \mathsf{U}\ h$. Il ne reste à envisager que le cas où $f = \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ g \wedge \mathsf{G}\, h)$: Si $a_i \models f$, alors $a_i \models h$ (et donc $a_{i+1} \models h$ et $a_{i-1} \models h$ si $|f| \geq i$) et soit $a_i \models g$ (d'où $a_{i-1} \models g$), soit $b_i \models h \wedge g$, soit $b_i \models h$ et $a_{i-1} \models f$. Dans tous les cas, on obtient $a_{i-1} \models f$ si $|f| > i$ et $a_{i+1} \models f$.

Il ne reste plus qu'à montrer que:

$$\forall f \in L^+, \forall i \geq |f|, \qquad a_i \models f \;\Leftrightarrow\; \alpha_i \models f \quad \text{et} \quad b_i \models f \;\Leftrightarrow\; \beta_i \models f$$

On montre ces résultats par induction sur la taille de f. On reprend la preuve de [EH86] pour les cas $f = a$, $f = \mathsf{EX}\, g$, $f = \mathsf{E}\, g \;\mathsf{U}\; h$ et $f = \mathsf{A}\, g \;\mathsf{U}\; h$. Il ne reste à envisager que le cas où $f = \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; g \wedge \mathsf{G}\, h)$: Le cas intéressant est celui où $\beta_i \models f$ avec le chemin $\alpha_i \to \beta_i \to \alpha_i \ldots$ (c'est le seul chemin de M_i ayant un suffixe qui n'existe pour aucun chemin de N_i) vérifiant $(\overset{\infty}{\mathsf{F}}\; g \wedge \mathsf{G}\, h)$. Dans ce cas, α_i et β_i vérifient h, et soit $\alpha_i \models g$ (et donc $a_i \models g$ et $a_{i-1} \models g$), soit $\beta_i \models g$ (donc $b_i \models g$) : on a donc bien $b_i \models \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; g \wedge \mathsf{G}\, h)$.

Comme $\alpha_i \models \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; a \wedge \overset{\infty}{\mathsf{F}}\; b)$ et $a_i \not\models \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; a \wedge \overset{\infty}{\mathsf{F}}\; b)$, aucune formule de L^+ n'est équivalente à $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; a \wedge \overset{\infty}{\mathsf{F}}\; b)$. Et donc: $\mathrm{L}^+ \prec_g ECTL^+$. $\square$

Nous avons donc montré que d'une part, $ECTL + \mathsf{F}^{-1}$ se traduisait initialement dans L^+, et que d'autre part L^+ était moins expressive que $ECTL^+$. On en déduit donc que $ECTL + \mathsf{F}^{-1} \prec_i ECTL^+$. $\square$

Remarque Pour montrer $ECTL^+ \not\preceq_i ECTL + \mathsf{F}^{-1}$, nous devions montrer qu'il existait une formule de $ECTL^+$ ($\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; a \wedge \overset{\infty}{\mathsf{F}}\; b)$) qui n'était pas exprimable avec $ECTL + \mathsf{F}^{-1}$: pour cela, nous avons fait intervenir une logique du futur L^+ plus expressive que $ECTL$ et telle que $ECTL + \mathsf{F}^{-1}$ se traduise dans L^+. Le problème qui restait, était de montrer que la formule $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; a \wedge \overset{\infty}{\mathsf{F}}\; b)$ était inexprimable avec L^+ (le F^{-1} était inutile puisque $L^+ \equiv_i L^+ + \mathsf{F}^{-1}$), et pour cela nous avons considéré les modèles N_n et M_n.

Ce passage par une logique pur-futur pour montrer l'impossibilité d'exprimer une certaine propriété est motivé par le fait que la définition inductive des modèles et les hypothèses d'induction concernant les équivalences entre les états sont beaucoup plus simples dans ce cas que lorsqu'on considère une logique mélangeant passé et futur. $\square$

Nous avons les mêmes résultats pour $ECTL + \mathsf{F}_s^{-1}$:

Théorème 4.4.26 $ECTL \prec_i ECTL + \mathsf{F}_s^{-1}$

Preuve La preuve est identique à celle du théorème 4.4.21 car:

$$\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\; p \wedge \mathsf{G}\, q) \equiv_i \mathsf{E}\,\overset{\infty}{\mathsf{F}}\; (p \wedge \mathsf{G}^{-1} q)$$

$\square$

Théorème 4.4.27 $ECTL + \mathsf{F}_s^{-1} \prec_i ECTL^+$

Preuve La preuve se fait de manière semblable à celle du théorème 4.4.22. Nous utilisons ici la logique $L^+ + \mathsf{F}_s^{-1}$ au lieu de $L^+ + \mathsf{F}^{-1}$. On adapte alors le théorème de séparation de $CTL + \mathsf{F}_s^{-1}$ à cette nouvelle logique grâce à l'équivalence suivante:

$$\begin{array}{|l|}
\hline
\mathsf{E}\left[\overset{\infty}{\mathsf{F}}\ (\alpha \wedge \mathsf{F}_s^{-1} x) \ \wedge\ \mathsf{G}\left((\mathsf{F}_s^{-1} x \wedge \alpha') \vee (\neg \mathsf{F}_s^{-1} x \wedge \beta') \vee \gamma'\right)\right] \\
\qquad \equiv\ \mathsf{F}^{-1} x \ \wedge\ \mathsf{E}\left(\overset{\infty}{\mathsf{F}}\ \alpha \ \wedge\ \mathsf{G}\,(\alpha' \vee \gamma')\right) \\
\qquad\quad \vee\ \neg \mathsf{F}^{-1} x \ \wedge\ \mathsf{E}\left[\left(\neg x \wedge (\beta' \vee \gamma')\right) \mathsf{U} \left(x \wedge (\beta' \vee \gamma') \wedge \mathsf{EX}\left\{\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ \alpha \ \wedge\ \mathsf{G}\,(\alpha' \vee \gamma'))\right\}\right)\right] \\
\hline
\mathsf{E}\left[\overset{\infty}{\mathsf{F}}\ (\alpha \wedge \neg \mathsf{F}_s^{-1} x) \ \wedge\ \mathsf{G}\left((\mathsf{F}_s^{-1} x \wedge \alpha') \vee (\neg \mathsf{F}_s^{-1} x \wedge \beta') \vee \gamma'\right)\right] \\
\qquad \equiv\ \neg \mathsf{F}_s^{-1} x \ \wedge\ \mathsf{E}\left[\overset{\infty}{\mathsf{F}}\ \alpha \ \wedge\ \mathsf{G}\left(\neg x \wedge (\beta' \vee \gamma')\right)\right] \\
\hline
\end{array}$$

□

Nous proposons la conjecture suivante, sur laquelle nous reviendrons au chapitre 5:

Conjecture 4.4.28 $L^+ \not\preceq_i ECTL + \mathsf{F}_s^{-1}$

Cette conjecture repose sur l'idée que la formule $\mathsf{E}\ \overset{\infty}{\mathsf{F}}\ (a \wedge \mathsf{G}^{-1} b)$ énonce qu'il existe une exécution **depuis l'état initial** le long de laquelle b est toujours vraie et où a est vraie infiniment souvent. La logique L^+ permet d'exprimer qu'il existe une exécution le long de laquelle b est toujours vraie *depuis l'état courant* et où a est vraie infiniment souvent. $\mathsf{E}\ \overset{\infty}{\mathsf{F}}\ (a \wedge \mathsf{G}^{-1} b)$ remonte à l'état initial et il ne semble pas possible d'exprimer avec $ECTL + \mathsf{F}_s^{-1}$ la formule $\mathsf{EF}\,(a \wedge \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ b \wedge \mathsf{G}\,c))$.

Puisque l'expressivité de $ECTL + \mathsf{F}^{-1}$ ne dépasse pas l'expressivité de $ECTL^+$, nous devons chercher si il est possible de traduire initialement $ECTL^+ + \mathsf{F}^{-1}$ dans $ECTL^+$.

4.4.5 La logique $ECTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$

L'ajout de l'opérateur F^{-1} à la logique $ECTL^+$ n'augmente pas son expressivité. Avant de montrer que $ECTL^+ + \mathsf{F}^{-1}$ est séparable, nous allons définir le fragment $\mathcal{F}_{ECTL^+}$ qui a la même expressivité que $ECTL^+$:

Définition 4.4.29 *La logique $\mathcal{F}_{ECTL^+}$:*

$$\mathcal{F}_{ECTL^+} \ni f, g \ ::= \ a \mid f \wedge g \mid \neg f \mid E\, f\, U\, g \mid A\, f\, U\, g \mid EX\, f \mid E\,(G\, f \wedge \bigwedge_i \overset{\infty}{F}\ g_i)$$

On a bien $\mathcal{F}_{ECTL^+} \subseteq ECTL^+$ et de plus, nous avons le lemme suivant:

Lemme 4.4.30 *Toute formule de $ECTL^+$ est équivalente à une formule de $\mathcal{F}_{ECTL^+}$.*

Preuve Ce lemme est basé sur les mêmes techniques que celles utilisées pour montrer $CTL^+ \equiv CTL$ [EH85]. Soit f une formule de $ECTL^+$ de la forme suivante:

$$f \;=\; \mathsf{E}\left(\begin{array}{c} \bigwedge_{i=1..n'} a'_i \,\mathsf{U}\, b'_i \;\wedge\; \bigwedge_{i=1..n''} \neg(a''_i \,\mathsf{U}\, b''_i) \\ \wedge\; \bigwedge_i \overset{\infty}{\mathsf{F}}\, d_i \;\wedge\; \bigwedge_i \neg\, \overset{\infty}{\mathsf{F}}\, d'_i \;\wedge\; \bigwedge_i \mathsf{X}\, c_i \;\wedge\; \bigwedge_i \neg\mathsf{X}\, c'_i \end{array}\right)$$

$$f \;\equiv\; \mathsf{E}\Big(\bigwedge_{i=1..n} a_i \,\mathsf{U}\, b_i \;\wedge\; \mathsf{G}\, c \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{F}}\, d_i \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{G}}\, d''_i \;\wedge\; \mathsf{X}\, c'\Big)$$

De plus:

$$f \equiv \bigwedge_{J\subseteq\{1,\ldots,n\}} \left(\begin{array}{c} \bigwedge_{i\in J} b_i \;\wedge\; \bigwedge_{i\notin J} a_i \;\wedge\; c \;\wedge \\ \wedge\, \mathsf{EX}\Big[c' \;\wedge\; \mathsf{E}\Big(\bigwedge_{i\notin J} a_i \,\mathsf{U}\, b_i \;\wedge\; \mathsf{G}\, c \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{F}}\, d_i \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{G}}\, d''_i\Big)\Big] \end{array}\right) \tag{4.52}$$

On a aussi:

$$\mathsf{E}\Big(\bigwedge_{i=1,\ldots,n} a_i \,\mathsf{U}\, b_i \;\wedge\; \mathsf{G}\, c \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{F}}\, d_i \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{G}}\, d''_i\Big) \;\equiv$$
$$\bigvee_{\pi\in Permut(n)} \mathsf{E}\;\Big(\bigwedge_i a_i \;\wedge\; c\Big)\,\mathsf{U}\,\Big[b_{\pi(1)} \;\wedge\; \mathsf{E}\,(\bigwedge_{i\neq\pi(1)} a_i \;\wedge\; c)\,\mathsf{U}\,\Big(\ldots \mathsf{E}\,(a_{\pi(n)} \;\wedge\; c)\,\mathsf{U}\left[\begin{array}{c} b_{\pi(n)} \;\wedge\; \mathsf{E}\Big(\mathsf{G}\, c\, \wedge \\ \bigwedge_i \overset{\infty}{\mathsf{F}}\, d_i \;\wedge\; \bigwedge_i \overset{\infty}{\mathsf{G}}\, d''_i\Big)\end{array}\right]\Big)\Big]$$

où $Permut(n)$ est l'ensemble des permutations sur $1,\ldots,n$

□

Ce lemme s'appuie sur le fait que la valeur de vérité des sous-formules ayant un opérateur $\overset{\infty}{\mathsf{F}}$ ou $\overset{\infty}{\mathsf{G}}$ à la racine dépend uniquement du suffixe de l'exécution: on peut donc les "repousser" dans les parties droites des *Until*. Emerson et Lei ont appelé cette caractéristique *obliviousness property* [EL87].

On peut établir un lemme similaire pour les logiques $ECTL^+ + \mathsf{F}^{-1}$ et $\mathcal{F}_{ECTL^+} + \mathsf{F}^{-1}$:

Lemme 4.4.31 *Toute formule de $ECTL^+ + \mathsf{F}^{-1}$ est équivalente à une formule de $\mathcal{F}_{ECTL^+} + \mathsf{F}^{-1}$.*

Preuve La preuve est identique à celle du lemme précédent. □

Pour montrer que $ECTL^{+} + \mathsf{F}^{-1}$ est séparable, il suffit donc de montrer que $\mathcal{F}_{ECTL^{+}} + \mathsf{F}^{-1}$ est séparable. Nous avons le lemme suivant:

Lemme 4.4.32 *Toute formule de $\mathcal{F}_{ECTL^{+}} + \mathsf{F}^{-1}$ est équivalente à une formule séparée (de $\mathcal{F}_{ECTL^{+}} + \mathsf{F}^{-1}$).*

Preuve On utilise le même procédé que pour la séparation de CTL. Il nous suffit de construire une règle adéquate pour traiter les formules de la forme: $\mathsf{E}\,(\mathsf{G}\,f \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\, g_i)$:

$$\boxed{\begin{array}{l} \mathsf{E}\left[\mathsf{G}\left((\mathsf{F}^{-1}x \,\wedge\, \alpha) \,\vee\, (\neg \mathsf{F}^{-1}x \,\wedge\, \beta) \,\vee\, \gamma\right) \,\wedge\, \bigwedge_i \overset{\infty}{\mathsf{F}}\, \left((\mathsf{F}^{-1}x \,\wedge\, \alpha_i) \,\vee\, (\neg \mathsf{F}^{-1}x \,\wedge\, \beta_i) \,\vee\, \gamma_i\right)\right] \;\equiv \\ \qquad \mathsf{F}^{-1}x \,\wedge\, \mathsf{E}\left[\mathsf{G}\,(\alpha \,\vee\, \gamma) \,\wedge\, \bigwedge_i \overset{\infty}{\mathsf{F}}\, (\alpha_i \,\vee\, \gamma_i)\right] \\ \qquad \vee \neg \mathsf{F}^{-1}x \,\wedge\, \mathsf{E}\left(\neg x \,\wedge\, (\beta \,\vee\, \gamma)\right) \mathsf{U} \left(x \,\wedge\, \mathsf{E}\left[\mathsf{G}\,(\alpha \,\vee\, \gamma) \,\wedge\, \bigwedge_i \overset{\infty}{\mathsf{F}}\, (\alpha_i \,\vee\, \gamma_i)\right]\right) \\ \qquad \vee \neg \mathsf{F}^{-1}x \,\wedge\, \mathsf{E}\left[\mathsf{G}\,(\neg x \,\wedge\, (\beta \,\vee\, \gamma)) \,\wedge\, \bigwedge_i \overset{\infty}{\mathsf{F}}\, (\beta_i \,\vee\, \gamma_i)\right] \end{array}} \tag{4.53}$$

Dans la formule de droite, les sous-formules $\mathsf{F}^{-1}x$ ne sont plus dans la portée d'opérateur du futur. On peut donc utiliser cette règle et garder la même démarche (lemmes 4.4.6, 4.4.7, 4.4.8 et 4.4.9) que celle établie pour CTL. □

A partir des lemmes 4.4.32 et 4.4.30, on déduit directement le lemme suivant:

Lemme 4.4.33 *Toute formule de $ECTL^{+} + \mathsf{F}^{-1}$ est équivalente à une formule séparée (de $ECTL^{+} + \mathsf{F}^{-1}$).*

Nous avons donc le théorème suivant:

Théorème 4.4.34 $ECTL^{+} + \mathsf{F}^{-1} \;\equiv_i\; ECTL^{+}$

Preuve directe à partir du lemme précédent et du théorème 4.3.3. □

Nous avons le même résultat avec l'opérateur F_s^{-1}:

Lemme 4.4.35 *Toute formule de $ECTL^{+} + \mathsf{F}_s^{-1}$ est équivalente à une formule séparée (de $ECTL^{+} + \mathsf{F}_s^{-1}$).*

Preuve La seule différence réside dans la règle de séparation utilisée. Ici nous avons:

$$\boxed{\begin{array}{l}\mathsf{E}\left[\mathsf{G}\left((\mathsf{F}_s^{-1}x \wedge \alpha) \vee (\neg\mathsf{F}_s^{-1}x \wedge \beta) \vee \gamma\right) \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\left((\mathsf{F}_s^{-1}x \wedge \alpha_i) \vee (\neg\mathsf{F}_s^{-1}x \wedge \beta_i) \vee \gamma_i\right)\right] \equiv \\ \quad \mathsf{F}_s^{-1}x \wedge \mathsf{E}\left[\mathsf{G}\,(\alpha \vee \gamma) \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\,(\alpha_i \vee \gamma_i)\right] \\ \quad \vee \neg\mathsf{F}_s^{-1}x \wedge \mathsf{E}\left(\neg x \wedge (\beta \vee \gamma)\right)\mathsf{U}\left(x \wedge (\beta \vee \gamma) \wedge \mathsf{EX}\,\mathsf{E}\left[\mathsf{G}\,(\alpha \vee \gamma) \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\,(\alpha_i \vee \gamma_i)\right]\right) \\ \quad \vee \neg\mathsf{F}_s^{-1}x \wedge \mathsf{E}\left[\mathsf{G}\,(\neg x \wedge (\beta \vee \gamma)) \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\,(\beta_i \vee \gamma_i)\right]\end{array}}$$

□

On obtient donc les théorèmes suivants concernant $ECTL^+ + \mathsf{F}_s^{-1}$ et $ECTL^+ + \mathsf{F}_s^{-1}, \mathsf{F}^{-1}$:

Théorème 4.4.36 $ECTL^+ + F_s^{-1} \equiv_i ECTL^+$

Théorème 4.4.37 $ECTL^+ + F_s^{-1}, F^{-1} \equiv_i ECTL^+$

Preuve semblable à celle du théorème 4.4.13. □

4.4.6 Bilan de l'introduction de F^{-1} et F_s^{-1}

Les résultats présentés dans cette section sont résumés par la figure 4.8.

Si les conjectures suivantes sont vraies, alors le schéma est complet au sens où on ne peut pas rajouter de flèche (excepté par transitivité).

Conjecture 4.4.38
- $UB^+ \not\leq_i UB + F^{-1}$
- $UB \not\leq_i UB^- + F^{-1}$

La première conjecture est basée sur l'idée que $\mathsf{EF}\left(a \wedge \mathsf{E}\,(\mathsf{F}\,b \wedge \mathsf{G}\,c)\right)$ nous semble inexprimable en $UB + \mathsf{F}^{-1}$. La seconde conjecture est motivée par la formule $\mathsf{EG}\,a$ qui paraît inexprimable en $UB^- + \mathsf{F}^{-1}$.

Les résultats les plus intéressants concernent les logiques CTL et $ECTL^+$ puisqu'il est possible de traduire $CTL + \mathsf{F}_s^{-1}, \mathsf{F}^{-1}$ (resp. $ECTL^+ + \mathsf{F}_s^{-1}, \mathsf{F}^{-1}$) dans CTL (resp. $ECTL^+$). On peut aussi noter que ces traductions sont basées sur un ensemble de règles de séparation, ce qui rend facile leur implémentation.

Une question importante qui n'est pas abordée dans ce travail, concerne la complexité des traductions. Par complexité, on entend ici la taille des formules résultant de la traduction (qui a tendance à exploser). Il est difficile de mesurer précisément cette taille, il est clair que dans le pire des cas elle est très élevée, et probablement non-élémentaire. Les traductions proposées sont proches, de ce point de vue, de

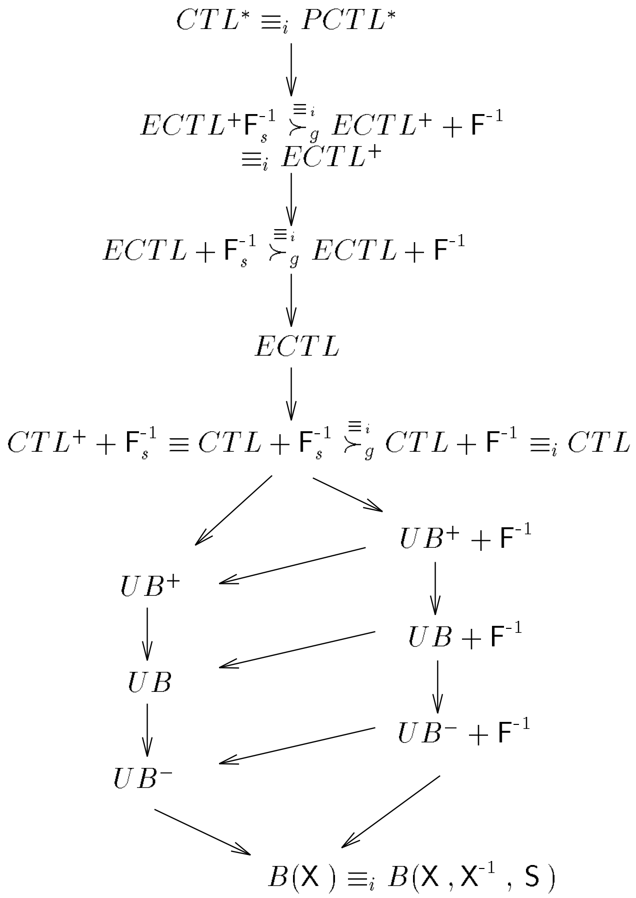


Figure 4.8: Hiérarchie du temps arborescent avec F^{-1},F_s^{-1}

la traduction pour le temps linéaire qui est réputée comme étant non-élémentaire (même si ce problème demeure ouvert à l'heure actuelle). En fait, notre objectif étant la traduction de spécifications, une étude très précise de cette complexité n'aurait pas un grand intérêt. Par contre, les questions importantes qui se posent, concernent d'une part l'applicabilité de ces traductions en raison de leur complexité et d'autre part l'existence de traductions plus efficaces.

Dans la pratique, lorsqu'on considère des spécifications typiques, la traduction peut être utilisée: la taille de la spécification traduite n'est pas prohibitive pour le model-checking. Ceci s'explique par le fait que les formules sont alors suffisamment simples, c'est-à-dire sans emboîtement important de modalités du passé, et sans un grand nombre d'alternations passé/futur qui sont responsables dans le cas général de la plus forte explosion de la taille. L'exemple du chapitre 3 souligne cet aspect: une traduction (à l'aide de règles de séparation [3]) de la spécification avec passé serait

[3] nous n'avons pas encore présenté de règles permettant la traduction de cette spécification qui utilise des opérateurs *Since*: c'est l'objet de la section suivante.

approximativement de même taille que celle de la spécification pur-futur proposée.

L'utilisation de simplifications booléennes et l'emploi d'autres règles de séparation (par exemple le fait de différencier plus de cas comme dans [LS94]) permettraient de diminuer un peu la taille des traductions. Néanmoins, la traduction de la formule $\mathsf{EF}\,(\mathsf{F}^{-1}a_1 \wedge \ldots \wedge \mathsf{F}^{-1}a_n)$ pose des problèmes similaires à ceux rencontrés pour la traduction de la formule de CTL^+ $\mathsf{E}\,(\mathsf{F}\,a_1 \wedge \ldots \wedge \mathsf{F}\,a_n)$. On sait que la traduction de CTL^+ dans CTL donne lieu à une explosion de la taille (si f est une formule de CTL^+, alors on sait que la taille de la formule CTL f' équivalente à f est bornée par $2^{(|f|.log(|f|))}$). Nous conjecturons qu'il n'existe pas de formule de CTL équivalente initialement à $\mathsf{E}\,(\mathsf{F}^{-1}a_1 \wedge \ldots \wedge \mathsf{F}^{-1}a_n)$ de taille en $\mathcal{O}(2^n)$. Ceci souligne aussi la difficulté d'isoler un fragment performant du point de vue de la complexité car la formule $\mathsf{E}\,(\mathsf{F}^{-1}a_1 \wedge \ldots \wedge \mathsf{F}^{-1}a_n)$ ne contient pas d'emboîtement important futur/passé [4].

En fait, la complexité des traductions souligne la richesse d'expressivité des opérateurs du passé: ils permettent d'énoncer de manière concise des propriétés complexes qui nécessitent des formules pur-futur de grande taille. Leur utilisation pour la spécification n'est donc pas à mettre en cause sur cette base [5], il s'agit plutôt de savoir quelles propriétés souhaitons-nous énoncer dans une spécification.

4.5 Une introduction partielle des opérateurs X^{-1} et S

Après avoir constaté que l'ajout des opérateurs *Previous* et *Since*, dans les logiques de temps arborescent (exceptée CTL^*), empêchait leur traduction dans des fragments comme CTL ou $ECTL^+$, nous avions présenté deux directions pour construire, malgré tout, des logiques du temps arborescent munies d'opérateurs du passé et "traduisibles" dans CTL ou $ECTL^+$: chercher à insérer des opérateurs plus faibles (F^{-1} ou F_s^{-1}), c'est ce que nous venons d'étudier, et d'autre part essayer de trouver des limitations dans l'emploi des opérateurs X^{-1} et S pour obtenir des formules séparables. Nous allons maintenant étudier cette seconde orientation.

L'objectif est donc, par exemple, de chercher des contraintes syntaxiques dans l'emploi des opérateurs X^{-1} et S. Pour cela, nous allons chercher à quelles conditions des règles de séparation semblables à celles présentées pour la séparation de $CTL+\mathsf{F}^{-1}$, peuvent être construites pour les opérateurs S et X^{-1}.

[4] Notons aussi que pour les formules de ce type utilisées dans les spécifications, n a une valeur faible.

[5] L'énoncé direct en logique pur-futur se révelerait finalement aussi couteux et augmenterait le risque d'erreur dans la conception de la spécification.

4.5.1 La logique CTL_p

Dans un premier temps, nous allons nous intéresser à la logique $CTL + \mathsf{S} + \mathsf{X}^{-1}$. La question est donc *existe-t-il des cas où l'on peut extraire (au sens des règles de séparation pour* $CTL + \mathsf{F}^{-1}\ldots$*) les opérateurs* X^{-1} *ou* S *de la portée d'opérateurs du futur?*

Concernant X^{-1}, nous avons les équivalences suivantes qui peuvent être utilisées comme règles de séparation pour extraire un X^{-1} de la partie droite d'un opérateur E_U et de la portée d'un opérateur EX :

$\mathsf{E}\Big(\varphi\,\mathsf{U}\,(\alpha \wedge \mathsf{X}^{-1}\,x)\Big)$	$\equiv$	$\alpha \wedge \mathsf{X}^{-1}\,x$ $\vee\,\mathsf{E}\Big(\varphi\,\mathsf{U}\,(\varphi \wedge x \wedge \mathsf{EX}\,\alpha)\Big)$	$R1$
$\mathsf{E}\Big(\varphi\,\mathsf{U}\,(\alpha \wedge \neg\mathsf{X}^{-1}\,x)\Big)$	$\equiv$	$\alpha \wedge \neg\mathsf{X}^{-1}\,x$ $\vee\;\mathsf{E}\Big(\varphi\,\mathsf{U}\,(\varphi \wedge \neg x \wedge \mathsf{EX}\,\alpha)\Big)$	$R2$
$\mathsf{EX}\,(\alpha \wedge \mathsf{X}^{-1}\,x)$	$\equiv$	$x \wedge \mathsf{EX}\,\alpha$	$R3$
$\mathsf{EX}\,(\alpha \wedge \neg\mathsf{X}^{-1}\,x)$	$\equiv$	$\neg x \wedge \mathsf{EX}\,\alpha$	$R4$

On peut donc séparer les formules ayant un $\mathsf{X}^{-1}\,x$ en partie droite de *Until*: la valeur de vérité de $\mathsf{X}^{-1}\,x$ ne dépend alors que de la valeur de vérité de x à l'état précédant l'état vérifiant la partie droite du *Until.* Il n'y a pas non plus de difficulté à extraire les sous-formules $\mathsf{X}^{-1}\,x$ de la portée d'un opérateur EX.

Mais nous ne pouvons pas séparer les formules où $\mathsf{X}^{-1}\,x$ (ou $\neg\mathsf{X}^{-1}\,x$) apparaît dans la partie gauche d'un *Until* ou dans la portée d'un EG car la valeur de vérité de $\mathsf{X}^{-1}\,x$ le long d'une séquence d'états nécessite de considérer la valeur de vérité de x à chaque état pour connaître la valeur de vérité de $\mathsf{X}^{-1}\,x$ à l'état **suivant sur la séquence**: il n'est pas possible de l'exprimer sans les opérateurs X^{-1} et X qui, étant donnée une exécution, permettent de faire référence respectivement à l'état précédent et à l'état suivant **le long de cette** exécution, à la différence de EX qui fait référence à un état successeur sur **une** exécution issue de l'état courant.

Pour l'opérateur *Since*, nous avons les équivalences suivantes qui peuvent être utilisées comme des règles de séparation pour extraire l'opérateur S de la portée d'un opérateur E_U ou EX :

$\mathsf{E}\left(\varphi\,\mathsf{U}\,(\alpha \wedge x\,\mathsf{S}\,y)\right)$	$\equiv$	$\mathsf{E}\,\varphi\,\mathsf{U}\,(\alpha \wedge y)$ $\vee\, \mathsf{E}\,\varphi\,\mathsf{U}\left(\varphi \wedge y \wedge \mathsf{EX}\,\mathsf{E}\,(\varphi \wedge x)\,\mathsf{U}\,(\alpha \wedge x)\right)$ $\vee\, x\,\mathsf{S}\,y \wedge \left(\alpha \vee \mathsf{E}\,(\varphi \wedge x)\,\mathsf{U}\,(\alpha \wedge x)\right)$	$R'1$
$\mathsf{E}\left(\varphi\,\mathsf{U}\,(\alpha \wedge \neg(x\,\mathsf{S}\,y))\right)$	$\equiv$	$\mathsf{E}\,\varphi\,\mathsf{U}\,(\alpha \wedge \neg x \wedge \neg y)$ $\vee\, \mathsf{E}\,\varphi\,\mathsf{U}\left(\varphi \wedge \neg x \wedge \neg y \wedge \mathsf{E}\,(\varphi \wedge \neg y)\,\mathsf{U}\,(\alpha \wedge \neg y)\right)$ $\vee\, \neg(x\,\mathsf{S}\,y) \wedge \left(\alpha \vee \mathsf{E}\,(\varphi \wedge \neg y)\,\mathsf{U}\,(\alpha \wedge \neg y)\right)$	$R'2$
$\mathsf{EX}\,(\alpha \wedge x\,\mathsf{S}\,y)$	$\equiv$	$\mathsf{EX}\,(\alpha \wedge y)$ $\vee\, x\,\mathsf{S}\,y \wedge \mathsf{EX}\,(\alpha \wedge x)$	$R'3$
$\mathsf{EX}\,(\alpha \wedge \neg(x\,\mathsf{S}\,y))$	$\equiv$	$\mathsf{EX}\,(\alpha \wedge \neg x \wedge \neg y)$ $\vee\, \neg(x\,\mathsf{S}\,y) \wedge \mathsf{EX}\,(\alpha \wedge \neg y)$	$R'4$

De la même manière que pour *Previous*, on ne peut pas séparer les formules ayant un opérateur *Since* dans la partie gauche d'une formule $\mathsf{E}\,\varphi\,\mathsf{U}\,\psi$ car il faudrait utiliser directement (i.e. sans quantificateur E) un autre opérateur U pour caractériser des séquences à l'intérieur de la séquence vérifiant $\varphi\,\mathsf{U}\,\psi$. On a un problème similaire avec l'opérateur EG.

Ces premiers éléments fournissent donc une condition nécessaire pour qu'une formule de $CTL + \mathsf{S} + \mathsf{X}^{-1}$ soit séparable par les mêmes méthodes que celles décrites pour la séparation de $CTL + \mathsf{F}^{-1}$: *les opérateurs X^{-1} et S (exceptés F^{-1} et F_s^{-1}) ne doivent pas apparaître dans la partie gauche d'un opérateur E_U, ni dans la portée d'un opérateur EG.*

Mais cette condition n'est pas suffisante. En effet:

1. La règle de séparation $R'1$ fait intervenir la sous-formule x dans la partie gauche d'un opérateur E_U: La présence d'opérateur S dans x pourrait donc empêcher la séparation à une étape ultérieure.

2. La règle de séparation $R'2$ fait intervenir la sous-formule $\neg y$ dans la partie gauche d'un opérateur E_U: La présence d'opérateur S dans y pourrait donc empêcher la séparation à une étape ultérieure.

3. Les règles de séparation utilisées pour séparer les formules ayant $\mathsf{F}^{-1}x$ ou $\mathsf{F}_s^{-1}x$ dans la portée d'un opérateur E_U font intervenir la sous-formule x dans la partie gauche d'un opérateur E_U: La présence d'opérateur S dans la portée d'un F^{-1} ou F_s^{-1} pourrait donc empêcher la séparation à une étape ultérieure.

A partir de ces remarques, on peut définir un fragment CTL_p de $CTL + \mathsf{X}^{-1} + \mathsf{S}$ en restreignant l'usage des opérateurs S et X^{-1} :

Définition 4.5.1 *La logique CTL_p*

$$CTL_p \ni f, g ::= a \mid f \wedge g \mid \neg f \mid \mathsf{EX}\, f \mid \mathsf{E}\, \varphi\, \mathsf{U}\, f \mid \mathsf{A}\, \varphi\, \mathsf{U}\, \psi \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1}\, f$$

où $\varphi, \psi \in CTL + \mathsf{F}_s^{-1}, \mathsf{F}^{-1}$

La logique CTL_p est donc le fragment de $CTL + \mathsf{X}^{-1} + \mathsf{S}$ où les opérateurs S et X^{-1} n'apparaissent ni dans la portée d'un opérateur S, ni dans la portée d'un opérateur $\mathsf{A\_U}$, ni dans une partie gauche d'un opérateur $\mathsf{E\_U}$. L'opérateur F^{-1} peut être utilisé sans contraintes particulières (comme dans $CTL + \mathsf{F}^{-1}$). Clairement, on a : $CTL + \mathsf{F}^{-1} \subseteq CTL_p$.

On peut aussi noter que la formule $\mathsf{A}\, f \mathsf{W}\, \varphi$ appartient à la logique CTL_p puisque $\mathsf{A}\, a \mathsf{W}\, b \equiv \neg \mathsf{E}\, (\neg b)\, \mathsf{U}\, (\neg b \wedge \neg a)$.

Exemple 4.5.2

- *$\mathsf{AG}\,(a\, \mathsf{S}\, b \Rightarrow \mathsf{E}\, d\, \mathsf{U}\, c) \in CTL_p$ car la sous-formule $a\, \mathsf{S}\, b$ apparaît dans la portée d'un opérateur EF ($\mathsf{AG} = \neg \mathsf{EF} \neg$) et donc dans la partie droite d'un $\mathsf{E\_U}$.*
- *$\mathsf{AG}\,(a\, \mathsf{S}\,(b \wedge b\, \mathsf{S}\, c) \Rightarrow d) \notin CTL_p$ car il y a deux S emboîtés.*
- *$\mathsf{E}\left((a \wedge \mathsf{F}^{-1} c)\, \mathsf{U}\,(c \wedge a\, \mathsf{S}\, d)\right) \in CTL_p$.*
- *$\mathsf{E}\left((a \wedge \mathsf{F}^{-1} c)\, \mathsf{U}\,(\mathsf{X}^{-1}\, \mathsf{X}^{-1}\, c \wedge a\, \mathsf{S}\, d)\right) \in CTL_p$.*

Nous avons le lemme suivant:

Lemme 4.5.3 *Toute formule de CTL_p est équivalente à une formule séparée (de CTL_p).*

Preuve Nous allons reprendre la même démarche que celle utilisée pour la séparation de $CTL + \mathsf{F}^{-1}$:

Lemme 4.5.4 *Soit $f[x]$ un contexte CTL. Nous avons les deux résultats suivants:*

1. *Si $f[x\, \mathsf{S}\, y]$ est un contexte de CTL_p, alors $f[x\, \mathsf{S}\, y]$ est équivalent à un contexte séparé $\tilde{f}[x, y, x\, \mathsf{S}\, y]$ de CTL_p où $\tilde{f}[x, y, z]$ est un contexte CTL.*
2. *Si $f[\mathsf{X}^{-1}\, x]$ est un contexte de CTL_p, alors $f[\mathsf{X}^{-1}\, x]$ est équivalent à un contexte séparé $\tilde{f}[x, \mathsf{X}^{-1}\, x]$ de CTL_p où $\tilde{f}[x, y]$ est un contexte CTL.*

Preuve On procède par induction structurelle sur $f[x]$. Si $f[x]$ est a, $f_1[x] \wedge f_2[x]$, ou $\neg f_1[x]$, on obtient facilement le résultat. Nous allons considérer les cas où $f[x]$ est de la forme $\mathsf{E}\,\varphi[x]\ \mathsf{U}\ \psi[x]$, $\mathsf{EG}\,\varphi[x]$ ou $\mathsf{EX}\,\psi[x]$. Mais comme $f[x\ \mathsf{S}\ y]$ et $f[\mathsf{X}^{-1}\,x]$ sont des contextes de CTL_p, on sait que $x\ \mathsf{S}\ y$ et $\mathsf{X}^{-1}\,x$ n'apparaissent ni dans la partie gauche d'un *Until*, ni dans la portée de EG. Le contexte $f[x]$ est donc de la forme $\mathsf{E}\,\varphi\ \mathsf{U}\ \psi[x]$ ou $\mathsf{EX}\,\psi[x]$. Nous allons nous intéresser au premier cas (le second est identique):

Le sous-contexte $\psi[x\mathsf{S}y]$ est séparable (par h.i.), et, par transformation booléenne, on obtient l'équivalence suivante où les ψ_i, ψ'_i et ψ''_i ne contiennent pas d'opérateur S:

$$\begin{aligned} f[x\ \mathsf{S}\ y] &\equiv \mathsf{E}\,\varphi\ \mathsf{U}\ \Big(\bigvee_i(\psi_i\ \wedge\ x\ \mathsf{S}\ y)\ \vee\ \bigvee_i(\psi'_i\ \wedge\ \neg x\ \mathsf{S}\ y)\ \vee\ \psi''_i\Big) \\ &\equiv \bigvee_i \mathsf{E}\,\varphi\ \mathsf{U}\ \Big(\psi_i\ \wedge\ x\ \mathsf{S}\ y\Big)\ \vee\ \bigvee_i \mathsf{E}\,\varphi\ \mathsf{U}\ \Big(\psi'_i\ \wedge\ x\ \mathsf{S}\ y\Big)\ \vee\ \bigvee_i \mathsf{E}\,\varphi\ \mathsf{U}\ \psi''_i \end{aligned}$$

On peut alors utiliser les règles $(R'1)$ et $(R'2)$ pour séparer la dernière formule. On procède de la même façon pour séparer $f[\mathsf{X}^{-1}\,x]$ et pour traiter le cas où $f = \mathsf{EX}\,\psi[x]$. Dans tous les cas, le contexte séparé obtenu est un contexte de CTL_p. □

Lemme 4.5.5 *Soit $f[x_1,\ldots,x_n]$ un contexte CTL.*

Alors si $f[x_1\,\mathsf{S}\,y_1,\ldots,\mathsf{X}^{-1}\,x_i,\ldots,\mathsf{F}_s^{-1}x_j,\ldots]$ est un contexte de CTL_p, il est équivalent à un contexte séparé $\tilde{f}[x_1,y_1,x_1\,\mathsf{S}\,y_1,\ldots,x_i,\mathsf{X}^{-1}\,x_i,\ldots,x_j,\mathsf{F}_s^{-1}x_j,\ldots]$ de CTL_p où $\tilde{f}[x_1,y_1,\ldots,x_n,y_n]$ est un contexte CTL.

Preuve On procède par induction sur n. Le lemme précédent et le lemme 4.4.6 permettent de traiter le cas $n = 1$. Soit $n = n'+1$. Si $f[x_1\mathsf{S}y_1,\ldots,\mathsf{X}^{-1}x_i,\ldots,\mathsf{F}_s^{-1}x_j,\ldots]$ est un contexte de CTL_p, alors cela entraîne que dans le contexte $f[x_1,\ldots,x_n]$, aucune variable x_l $(l < j)$ n'apparaît en partie gauche de *Until* ou dans la portée d'un opérateur EG.

On considère le contexte $f[x_1\ \mathsf{S}\ y_1, x_2,\ldots,x_n]$ (on procéderait de manière identique avec $\mathsf{X}^{-1}\,x_1$, et le cas $\mathsf{F}_s^{-1}x_1$ se traiterait de la même manière qu'au lemme 4.4.7). D'après le lemme 4.5.4, il existe un contexte séparé $\tilde{f}[x_1,y_1,x_1\ \mathsf{S}\ y_1,x_2,\ldots x_n]$ de CTL_p tel que $\tilde{f}[x_1,y_1,z_1,x_2,\ldots,x_n]$ soit un contexte CTL_p. En fait, ce dernier contexte est obtenu à partir de $f[x_1,\ldots,x_n]$ par l'application de règles de distributivité booléenne et par les règles $(R'1)$ et $(R'2)$: à chacune de ces étapes, les seules variables pouvant être placées dans la partie gauche d'un *Until* sont x_1 et y_1. Aucune variable ne peut être placée dans la portée d'un EG. On peut donc utiliser l'hypothèse d'induction pour séparer $\tilde{f}[x_1,y_1,z_1,x_2\ \mathsf{S}\ y_2,\ldots\mathsf{X}^{-1}\,x_i,\ldots\mathsf{F}_s^{-1}x_j]$. □

Lemme 4.5.6 *Soient $f[x_1,\ldots,x_n]$ un contexte CTL et $\varphi_1^-,\ldots,\ \varphi_n^-$ des formules pur-passé (i.e. de $L(\mathsf{X}^{-1}\ ,\ \mathsf{S}\,))$. Alors si $f[\varphi_1^-,\ldots,\varphi_n^-]$ est une formule de CTL_p, $f[\varphi_1^-,\ldots,\varphi_n^-]$ est équivalente à une formule séparée de CTL_p.*

Preuve La preuve se fait par induction sur la hauteur de passé (le nombre maximum d'opérateurs du passé emboîtés) de $f[\varphi_1^-, \ldots, \varphi_n^-]$. Cette formule est de la forme:

$$g[\psi_1 \,\mathsf{S}\, \psi'_1, \ldots, \mathsf{X}^{-1}\, \psi_i, \ldots, \mathsf{F}^{-1}\psi_j, \ldots]$$

où g est un contexte CTL et où chaque ψ_k et ψ'_k sont des sous-formules pur-passé d'un φ_l^- ayant donc une hauteur de passé inférieure à celui des φ_j^-.

On applique le lemme précédent pour séparer le contexte $g[x_1 \,\mathsf{S}\, y_1, \ldots, \mathsf{X}^{-1}\, x_i, \ldots \mathsf{F}^{-1} x_j, \ldots]$: on obtient alors un contexte séparé équivalent dans lequel les variables $x_i \ldots x_{j-1}$ ne peuvent pas apparaître dans la partie gauche d'un *Until*. Puisque dans CTL_p, les *Since* et les Previous ne peuvent être dans la portée de S ou X^{-1}, alors on peut appliquer l'hypothèse d'induction sur le nouveau contexte dans le lequel on a remplacé les x_k et y_k par ψ_k et ψ'_k. $\square$

Lemme 4.5.7 *Si $f[x_1, \ldots, x_n]$ est un contexte CTL et $\psi_1, \ldots, \psi_n$ sont des formules séparées de CTL_p, alors $f[\psi_1, \ldots, \psi_n]$ est équivalente à une formule séparée de CTL_p.*

Preuve identique au lemme 4.4.9 pour la séparation de $CTL + \mathsf{F}^{-1}$. $\square$

Disposant de ce lemme, on prouve facilement par induction structurelle sur f, que toute formule f de CTL_p est équivalente à une formule séparée. $\square$

On en déduit donc le théorème suivant:

Théorème 4.5.8 $CTL_p \equiv_i CTL$

Remarque La logique CTL_p n'est pas le plus grand fragment de $CTL + \mathsf{X}^{-1} + \mathsf{S}$ séparable. En effet, nous avons considéré les trois remarques (1, 2, 3) concernant les problèmes éventuels sur l'emboîtement des S et la présence des S dans la portée de F^{-1} ou F_s^{-1} comme des conditions nécessaires. Mais il existe des formules de $CTL + \mathsf{S} + \mathsf{X}^{-1}$ qui n'appartiennent pas à CTL_p et qui sont séparables. En voici un exemple:

$$\begin{aligned}
\mathsf{AG}\,(a \,\mathsf{S}\, (b \,\mathsf{S}\, c) \Rightarrow d) &\equiv \neg \mathsf{EF}\,(a \,\mathsf{S}\, (b \,\mathsf{S}\, c) \wedge \neg d) \\
&\equiv \neg \Big[\mathsf{EF}\,(b \,\mathsf{S}\, c \wedge \neg d) \\
&\qquad \vee \mathsf{EF}\,\big(b \,\mathsf{S}\, c \wedge \mathsf{EX}\,(\mathsf{E}\, a \,\mathsf{U}\, (a \wedge \neg d))\big) \\
&\qquad \vee a \,\mathsf{S}\, (b \,\mathsf{S}\, c) \wedge \big(\neg d \vee \mathsf{E}\, a \,\mathsf{U}\, (a \wedge \neg d)\big)\Big]
\end{aligned}$$

La formule obtenue est séparable puisqu'elle est une combinaison booléenne de formules de CTL_p et d'une formule déjà séparée $(a\,\mathsf{S}\,(b\,\mathsf{S}\,c))$.

Le choix de se restreindre à CTL_p est motivé par le souhait de décrire aisément un fragment et de disposer ainsi d'un critère rapide pour décider si une formule donnée en fait, ou non, partie. $\square$

Au chapitre 3, nous avons construit deux spécifications pour un ascenseur. La première, $SPEC_1$, est constituée de formules de $CTL+\mathsf{S}$ [6] et la seconde, $SPEC_2$, de formules de CTL^+. Pour construire $SPEC_2$, nous avons, par exemple, remplacé des formules de $SPEC_1$ de la forme $\mathsf{AG}\,(a \;\Rightarrow\; (a\,\mathsf{S}\,b))$ (formules (3.9), (3.10),...) par des formules de la forme $\mathsf{AG}\,(\neg a \;\Rightarrow\; \mathsf{A}\,\neg a\,\mathsf{W}\;b)$ (formules (3.33), (3.34), ...). Dans l'exemple suivant nous montrons que ces deux formules sont équivalentes initialement en utilisant les règles de séparation pour CTL_p et des simplifications booléennes:

Exemple 4.5.9

$$\begin{aligned}
\mathsf{AG}\,(a \;\Rightarrow\; (a\,\mathsf{S}\,b)) \;&\equiv\; \neg\mathsf{EF}\,\neg(\neg a \;\vee\; (a\,\mathsf{S}\,b))\\
&\equiv\; \neg\Big\{\mathsf{EF}\,(a \wedge \neg a \wedge \neg b) && \textit{Règles } R_2'\\
&\qquad \vee\, \mathsf{EF}\Big(\neg a \wedge \neg b \wedge \mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b)\Big)\\
&\qquad \vee\, \neg(a\,\mathsf{S}\,b) \wedge (a \vee \mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b))\Big\}\\
&\equiv_i\; \neg\Big\{\mathsf{EF}\Big(\neg a \wedge \neg b \wedge \mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b)\Big)\\
&\qquad \vee\, \Big(\neg a \wedge \neg b \wedge (a \vee \mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b))\Big)\Big\}\\
&\equiv_i\; \neg\mathsf{EF}\Big(\neg a \wedge \neg b \wedge \mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b)\Big) && \textit{simpl. bool.}\\
&\equiv_i\; \mathsf{AG}\Big(\neg(\neg a \wedge \neg b) \vee \neg\mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b)\Big)\\
&\equiv_i\; \mathsf{AG}\Big((\neg a \wedge \neg b) \;\Rightarrow\; \neg\mathsf{E}\,(\neg b)\,\mathsf{U}\,(a \wedge \neg b)\Big)\\
&\equiv_i\; \mathsf{AG}\Big((\neg a \wedge \neg b) \;\Rightarrow\; \mathsf{A}\,(\neg a)\,\mathsf{W}\;b\Big)\\
&\equiv_i\; \mathsf{AG}\Big(\neg a \;\Rightarrow\; \mathsf{A}\,(\neg a)\,\mathsf{W}\;b\Big)
\end{aligned}$$

La dernière équivalence est basée sur le fait que $s \models b$ implique $s \models \mathsf{A}\,(\neg a)\,\mathsf{W}\,b$.

[6] En fait, chacune de ces formules fait partie de CTL_p.

4.5.2 La logique CTL_p^+

Nous allons maintenant chercher à étendre la logique $CTL^+ + \mathsf{F}^{-1}$ sur la base des remarques faites pour l'élaboration de CTL_p. Nous rappelons que la logique $CTL^+ + \mathsf{F}^{-1}$ peut être définie à l'aide des deux grammaires suivantes:

$$CTL^+ + \mathsf{F}^{-1} \ni f, g \quad ::= \; a \mid f \wedge g \mid \neg f \mid \mathsf{E}\, f_c \mid \mathsf{F}^{-1} f$$

$$f_c, g_c \quad ::= \; f_c \wedge g_c \mid \neg f_c \mid f\, \mathsf{U}\, g \mid \mathsf{X}\, f \mid \mathsf{F}^{-1} f$$

Nous définissons la logique CTL_p^+ de la manière suivante:

Définition 4.5.10 *La logique* CTL_p^+

$$CTL_p^+ \ni f, g \quad ::= \; a \mid f \wedge g \mid \neg f \mid \mathsf{E}\, f_c \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f$$

$$f_c, g_c \quad ::= \; f_c \wedge g_c \mid f_c \vee g_c \mid \neg \varphi_c \mid \varphi\, \mathsf{U}\, f \mid \mathsf{X}\, f \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f$$

$$CTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1} \ni \varphi, \psi \quad ::= \; a \mid \varphi \wedge \psi \mid \neg \varphi \mid \mathsf{E}\, \varphi_c \mid \mathsf{F}^{-1} \varphi \mid \mathsf{F}_s^{-1} \varphi$$

$$\varphi_c, \psi_c \quad ::= \; \varphi_c \wedge \psi_c \mid \neg \varphi_c \mid \varphi\, \mathsf{U}\, \psi \mid \mathsf{X}\, \varphi \mid \mathsf{F}^{-1} \varphi \mid \mathsf{F}_s^{-1} \varphi$$

La logique CTL_p^+ correspond à la logique CTL^+ munie des opérateurs X^{-1} , S , F^{-1} et F_s^{-1} dans laquelle on autorise librement l'usage de F^{-1} et F_s^{-1} et dans laquelle on interdit la présence d'opérateurs X^{-1} et S dans la partie gauche d'un opérateur *Until* et dans la portée d'un *Since*. Pour cela, on limite les combinaisons booléennes de formules de chemins f_c et g_c aux opérateurs $\wedge$ et $\vee$: en effet, l'usage de la négation pourrait empêcher la séparation du fait des formules du type $\mathsf{E}\,\neg(\varphi\, \mathsf{U}\, f)$ où f contiendrait un opérateur S . Nous allons maintenant montrer que toute formule f de CTL_p^+ est équivalente à une formule de CTL_p:

Lemme 4.5.11 $CTL_p^+ \preceq_g CTL_p$

Preuve Considérons une formule $\mathsf{E}\, f_c$ de CTL_p^+. Alors $\mathsf{E}\, f_c$ est équivalente à une disjonction de formules h ayant la forme suivante:

$$\mathsf{E}\left[\bigwedge_i \varphi_i\, \mathsf{U}\, f_i \wedge \bigwedge_i \mathsf{X}\, f_i^{'} \wedge \bigwedge_i \varphi'\, \mathsf{U}\, \psi' \wedge \bigwedge_i \neg(\varphi_i^{''}\, \mathsf{U}\, \psi_i^{''})\right] \wedge \bigwedge_i \varphi^{'''}\, \mathsf{S}\, \psi^{'''} \wedge \bigwedge_i \mathsf{X}^{-1} f_i^{''} \wedge \bigwedge_i \mathsf{F}_s^{-1} \varphi_i^{''''} \tag{4.54}$$

A chacune de ces formules h on peut associer une formule $\tilde{h}$ de CTL_p équivalente en utilisant des méthodes identiques à celles présentées pour montrer $ECTL^+ \preceq_i$

$\mathcal{F}_{ECTL^+}$. En effet, à chaque étape de cette transformation, aucune formule f_i ou $f_i^{'}$ n'est placée en partie gauche de *Until* ou dans la portée d'un opérateur G. Ceci permet de déduire que si les sous-formules f_i, $f_i^{'}$,...sont des formules de CTL_p et si les sous-formules φ_i, ψ_i, $\varphi_i^{'}$ ...sont des formules de $CTL + \mathsf{F}^{-1}$, alors la formule $\tilde{h}$ est une formule de CTL_p.

Etant donnée une formule f de CTL_p^+, il suffit d'appliquer la transformation précédente aux sous-formules $\mathsf{E}\, f_c$ dont toutes les sous-formules $\mathsf{E}g$ sont dans CTL_p, puis de recommencer jusqu'à obtenir une formule de CTL_p équivalente. □

En conséquence de ce dernier lemme, nous avons:

Lemme 4.5.12 *Toute formule de CTL_p^+ est équivalente à une formule séparée (de CTL_p^+).*

Et:

Théorème 4.5.13 $CTL_p^+ \equiv_i CTL$

4.5.3 La logique $ECTL_p^+$

De la même manière que nous avons construit CTL_p^+, on peut définir la logique $ECTL_p^+$ qui correspond à la logique $ECTL^+ + \mathsf{F}^{-1}$ étendue avec les opérateurs X^{-1} et S munis des mêmes restrictions que dans CTL_p^+:

Définition 4.5.14 $ECTL_p^+$

$$
\begin{aligned}
ECTL_p^+ \ni f, g \quad &::= a \mid f \wedge g \mid \neg f \mid \mathsf{E}\, f_c \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \\
f_c, g_c \quad &::= f_c \wedge g_c \mid f_c \vee g_c \mid \neg \varphi_c \mid \varphi\, \mathsf{U} f \mid \mathsf{X} f \mid \overset{\infty}{\mathsf{F}} \varphi \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \\
ECTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1} \ni \varphi, \psi \quad &::= a \mid \varphi \wedge \psi \mid \neg \varphi \mid \mathsf{E}\, \varphi_c \mid \mathsf{F}^{-1} \varphi \mid \mathsf{F}_s^{-1} \varphi \\
\varphi_c, \psi_c \quad &::= \varphi_c \wedge \psi_c \mid \neg \varphi_c \mid \varphi\, \mathsf{U}\, \psi \mid \mathsf{X}\, \varphi \mid \overset{\infty}{\mathsf{F}} \varphi \mid \mathsf{F}^{-1} \varphi \mid \mathsf{F}_s^{-1} \varphi
\end{aligned}
$$

On n'autorise pas les sous-formules $\psi\, \mathsf{S}\, \varphi$ ou $\mathsf{X}^{-1}\, \varphi$ dans la portée de l'opérateur $\overset{\infty}{\mathsf{F}}$ pour les mêmes raisons qui nous ont conduits à les interdire dans la portée des opérateurs EG.

Nous allons définir le fragment $\mathcal{F}_{ECTL_p^+}$ de $ECTL_p^+$ en suivant une démarche similaire à celle que nous avions utilisée pour définir $\mathcal{F}_{ECTL^+}$ à partir de $ECTL^+$: on va utiliser un opérateur $\mathsf{E}\,(\mathsf{G}\, \varphi \bigwedge_i \overset{\infty}{\mathsf{F}}\, \psi_i)$ qui simplifie la structure des formules (plus de combinaison booléenne de formules de chemin dans la portée du quantificateur E). A la différence de $\mathcal{F}_{ECTL^+}$, le fragment $\mathcal{F}_{ECTL_p^+}$ contient des opérateurs du passé:

Définition 4.5.15 *La logique $\mathcal{F}_{ECTL_p^+}$:*

$$\begin{array}{rl}\mathcal{F}_{ECTL_p^+} \ni f, g & ::= a \mid f \wedge g \mid \neg f \mid \mathsf{E}\varphi\,\mathsf{U}\,g \mid \mathsf{A}\varphi\,\mathsf{U}\,\psi \mid \mathsf{EX}\,f \mid \mathsf{E}(\mathsf{G}\varphi \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\ \psi_i) \\ & \varphi\,\mathsf{S}\,\psi \mid \mathsf{X}^{-1}\,f\end{array}$$

où φ, ψ et les ψ_i sont des formules de $\mathcal{F}_{ECTL^+} + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$.

Nous avons le premier lemme:

Lemme 4.5.16 $ECTL_p^+ \equiv \mathcal{F}_{ECTL_p^+}$

Preuve Etant donnée une formule f de $ECTL_p^+$, on utilise la même transformation que celle utilisée pour montrer $ECTL^+ \preceq_g \mathcal{F}_{ECTL^+}$ et $CTL_p^+ \preceq_i CTL_p$. On obtient facilement $\tilde{f}$, une formule de $\mathcal{F}_{ECTL_p^+}$ équivalente à f: la transformation consiste essentiellement à "déplier" les sous-formules $\mathsf{E}(\bigwedge_i a_i\,\mathsf{U}\,b_i)$, cette transformation ne place aucun b_i en partie gauche de $Until$ dans $\tilde{f}$, ceci garantit $\tilde{f} \in \mathcal{F}_{ECTL_p^+}$. $\square$

Nous allons adapter le lemme de séparation de CTL_p au fragment $\mathcal{F}_{ECTL_p^+}$:

Lemme 4.5.17 *Toute formule de $\mathcal{F}_{ECTL_p^+}$ est équivalente à une formule séparée (de $\mathcal{F}_{ECTL^+}$).*

Preuve Dans les sous-formules $\mathsf{E}(\mathsf{G}\varphi \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\ \psi_i)$, il n'y a pas d'opérateur X^{-1} ni d'opérateur S puisque les ψ_i et φ sont des formules de $\mathcal{F}_{ECTL^+} + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$, on peut donc utiliser la règle de séparation (4.53) présentée dans la preuve du lemme 4.4.32 en plus de celles utilisées pour la séparation de CTL_p. $\square$

On en déduit le lemme de séparation pour $ECTL_p^+$:

Lemme 4.5.18 *Toute formule de $ECTL_p^+$ est équivalente à une formule séparée (de $ECTL_p^+$).*

Preuve Directe à partir des deux derniers lemmes. $\square$

Et enfin, le théorème 4.3.3 nous permet de déduire:

Théorème 4.5.19 $ECTL_p^+ \equiv_i ECTL^+$

$$
\begin{tikzcd}[column sep=small]
ECTL_p^+ & \overset{\succ_g}{\equiv_i} & ECTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1} & \overset{\succ_g}{\equiv_i} \arrow[d] & ECTL^+ + \mathsf{F}^{-1} & \overset{\succ_g}{\equiv_i} & ECTL^+ \\
CTL_p^+ \equiv CTL_p & \overset{\succ_g}{\equiv_i} & CTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1} & \overset{\succ_g}{\equiv_i} & CTL^+ + \mathsf{F}^{-1} & \overset{\succ_g}{\equiv_i} & CTL
\end{tikzcd}
$$

Figure 4.9: Traductions dans CTL et $ECTL^+$

4.6 Conclusion

Les résultats présentés dans ce chapitre montrent qu'il est possible d'ajouter certains opérateurs du passé aux logiques CTL et $ECTL^+$ et de traduire les logiques ainsi étendues dans CTL ou $ECTL^+$. Plus précisément, nous avons le schéma 4.9.

L'existence d'un module de traduction $CTL_p \preceq_i CTL$ en amont d'un outil de model-checking permettrait donc d'utiliser **en plus** des spécifications habituelles en CTL, des spécifications avec du passé comme celle de l'ascenseur du chapitre 3. Il en est de même pour CTL_p^+ et $ECTL_p^+$.

Dans ce cadre (traduction en amont d'un model-checker), le schéma 4.10 présente les différentes solutions possibles en associant à chaque logique L de la hiérarchie classique, l'ensemble des logiques avec passé qui peuvent se traduire initialement dans L.

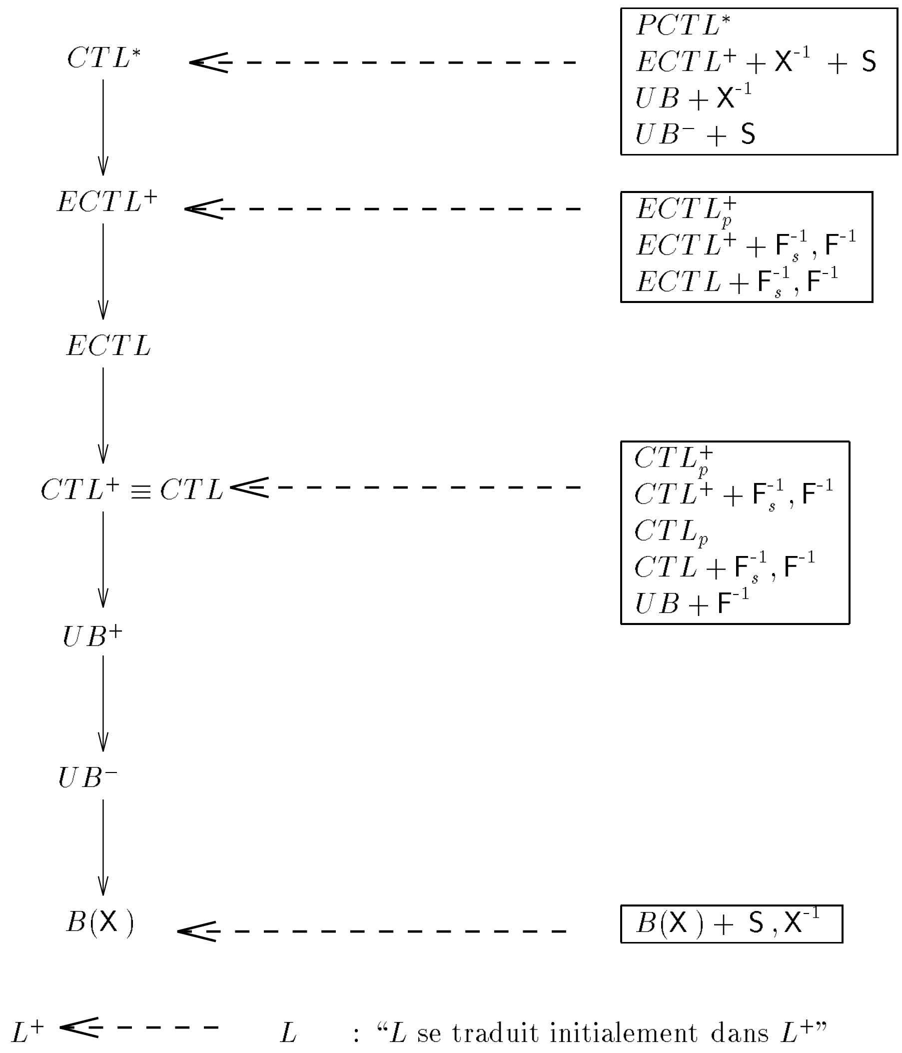


Figure 4.10: Traduction de logiques avec passé dans les fragments de CTL^*

Chapitre 5

Logiques avec *"Now"*

5.1 introduction et motivation

L'utilisation des opérateurs du passé S et X^{-1} permet d'élaborer des spécifications plus simples et plus intuitives que celles obtenues à l'aide des seuls opérateurs du futur. Néanmoins il peut s'avérer utile de définir de nouveaux outils pour bénéficier davantage des possibilités qu'offrent les opérateurs du passé.

Dans l'exemple du distributeur de billet, nous avions considéré la formule de $CTL + \mathsf{F}^{-1}$, $\mathsf{AG}\,(\textit{blocage} \;\Rightarrow\; \mathsf{F}^{-1}\textit{erreur})$ qui énonce que si (un jour) la carte est bloquée, alors il y a eu (précédemment) une erreur dans le code. Pourtant, au cours d'une exécution de ce système, plusieurs utilisateurs différents peuvent se servir du distributeur. Ainsi est-il nécessaire de spécifier que si la carte d'un client donné est bloquée, c'est ce client qui a commis une erreur dans son code. Considérons la formule suivante où la proposition atomique *Nouvelle carte* correspond à l'insertion d'une nouvelle carte:

$$\mathsf{AG}\left(\textit{Nouvelle carte} \;\Rightarrow\; \mathsf{AG}\,(\textit{blocage} \Rightarrow \mathsf{F}^{-1}\textit{erreur})\right) \tag{5.55}$$

Cette formule énonce qu'à partir de chaque état correspondant à l'introduction d'une carte, il est toujours vrai que si (un jour) la carte est bloquée alors il y a eu (avant) une erreur dans le code. Cette formule ne résoud pas le problème posé: la sémantique de l'opérateur F^{-1} fait que l'erreur peut avoir été commise avant l'insertion de cette carte (p.ex. par un autre utilisateur). On souhaiterait donc spécifier que l'état vérifiant *erreur* est situé avant le blocage mais après l'insertion de la carte. Pour cela, nous allons définir l'opérateur *Now* ou *from Now on*, qui permet de restreindre la portée temporelle de ses arguments. On peut alors construire la formule suivante:

$$\mathsf{AG}\left(\mathit{Nouvelle\ carte} \;\Rightarrow\; \mathsf{N}\,\mathsf{AG}\,(\mathit{blocage} \Rightarrow \mathsf{F}^{-1}\mathit{erreur})\right) \tag{5.56}$$

La formule (5.56) énonce qu'il est toujours vrai qu'à chaque fois qu'une nouvelle carte est insérée, alors **à partir de ce moment** tout blocage exige qu'il y ait eu une erreur auparavant.

Au chapitre 3, nous avons décrit une spécification pour un ascenseur. A cette occasion, nous avions eu besoin de dire qu'une lampe ne pouvait être allumée que si le bouton qui lui était associé avait été enfoncé (cf. formule (3.9). L'usage de l'opérateur F^{-1} qui pouvait paraître naturel (avec $\mathsf{AG}\bigwedge_i[\mathit{C\text{-}Light}_i \Rightarrow \mathsf{F}^{-1}\mathit{Call}_i]$) ne convenait pas car une seule pression du bouton suffisait pour autoriser ensuite l'allumage de la lampe à n'importe quel instant. L'opérateur *Now* permet de remédier à ce problème en exprimant que si une lampe passe de l'état "éteinte" à l'état "allumée" alors c'est qu'il y eu *entre temps* pression sur le bouton:

$$\mathsf{AG}\bigwedge_i\left[\neg \mathit{C\text{-}Light}_i \;\Rightarrow\; \mathsf{N}\,\mathsf{AF}\left(\mathit{C\text{-}Light}_i \;\Rightarrow\; \mathsf{F}^{-1}\mathit{Call}_i\right)\right] \tag{5.57}$$

De plus, nous avions eu besoin d'énoncer qu'un étage ne pouvait être desservi que si une requête (non encore satisfaite) existait pour cet étage. Pour cela, on peut utiliser la formule suivante:

$$\mathsf{AG}\bigwedge_i\left[\neg \mathit{request}_i \;\Rightarrow\; \mathsf{N}\,\mathsf{AG}\left(\mathit{servicing}_i \;\Rightarrow\; \mathsf{F}^{-1}\mathit{request}_i\right)\right] \tag{5.58}$$

L'objectif de ce chapitre est l'étude de l'insertion de l'opérateur N dans les différentes logiques présentées au chapitre précédent. Dans un premier temps nous allons présenter la sémantique de cet opérateur, puis nous chercherons à traduire les logiques étendues avec N dans les fragments CTL ou $ECTL^+$.

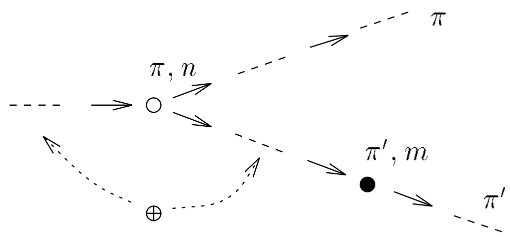
5.2 Définition et propositions

La sémantique de l'opérateur N est définie comme suit:

Définition 5.2.1 $\pi, n \models \mathsf{N} f$ *ssi* $\pi^n, 0 \models f$

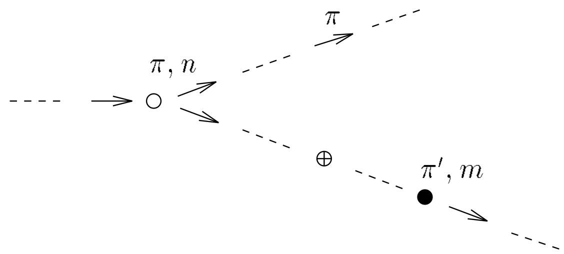
On note $NCTL^*$ la logique $PCTL^*$ étendue avec l'opérateur N .

Exemple 5.2.2 *La formule* $\mathsf{EF}(a \wedge \mathsf{F}^{-1} b)$ *est vraie en* π, n *si il existe dans le futur* π', m *(donc* $\pi_{|n} = \pi'_{|n}$*) vérifiant* a *et si il existe dans le passé de* π', m *un état (le long de* π'*) vérifiant* b *(i.e situé avant* π, n *ou entre* π, n *et* π', m*):*



$\bullet \;\; \models a \qquad \oplus \;\; \models b \qquad \circ \;\; \models \mathsf{EF}(a \wedge \mathsf{F}^{-1} b)$

Mais la formule $\mathsf{N\,EF}(a \wedge \mathsf{F}^{-1} b)$ *est vraie en* π, n *si il existe (dans le futur)* π', m *vérifiant* a *tel qu'il existe un état vérifiant* b *situé entre* π, n *et* π', m*: dans le futur (*N *) de* π, n *et dans le passé (*F^{-1}*) de* π', m*:*



$\bullet \;\; \models a \qquad \oplus \;\; \models b \qquad \circ \;\; \models \mathsf{N\,EF}(a \wedge \mathsf{F}^{-1} b)$

Remarque La logique $PCTL^*_{HT}$ définie dans [HT87] utilise une sémantique différente de la notre pour le quantificateur E . L'opérateur Now nous permet de définir ce quantificateur E_{HT} à partir de notre E :

$$\mathsf{E}_{HT} f \equiv \mathsf{N}\,\mathsf{E}\, f$$

En conséquence, on peut voir $PCTL^*_{HT}$ comme un fragment particulier de $NCTL^*$. $\square$

Nous avons les équivalences suivantes:

Proposition 5.2.3 *Pour toutes formules f et g de $NCTL^*$:*

1. $\mathsf{N}(f \wedge g) \equiv \mathsf{N}f \wedge \mathsf{N}g$
2. $\mathsf{N}(f \vee g) \equiv \mathsf{N}f \vee \mathsf{N}g$
3. $\mathsf{N}(\neg f) \equiv \neg\mathsf{N}f$
4. $\mathsf{N}\,\mathsf{E}f \equiv \mathsf{E}\,\mathsf{N}f$
5. $\mathsf{N}f \equiv_i f$
6. $\mathsf{N}\,\mathsf{N}f \equiv \mathsf{N}f$
7. $\mathsf{N}\,\mathsf{X}^{-1} f \equiv \bot$
8. $\mathsf{N}(f\,\mathsf{S}\,g) \equiv \mathsf{N}g$

Ces propositions sont faciles à montrer. L'opérateur N permet aussi de caractériser syntaxiquement la notion de formule du futur:

Proposition 5.2.4 *Une formule f est une formule du futur ssi $f \equiv \mathsf{N}f$.*

Preuve Soit f une formule du futur. Supposons $\pi, n \models f$. Alors pour tout π' telle que $\pi'^m = \pi^n$, on a $\pi', m \models f$, et donc en particulier: $\pi^n, 0 \models f$, d'où $\pi, n \models \mathsf{N}\,f$. Supposons $\pi, n \models \mathsf{N}\,f$, i.e. $\pi^n, 0 \models f$. Comme f est une formule du futur, on peut ajouter à π^n n'importe quel préfixe et en particulier $\pi_{|n}$: $\pi, n \models f$.

Supposons $f \equiv \mathsf{N}f$ et $\pi, n \models f$. Soit π' et m tels que $\pi'^m = \pi^n$. Comme $f \equiv \mathsf{N}f$, on a $\pi^n, 0 \models f$ et donc $\pi'^m, 0 \models f$. On en déduit $\pi'^m, 0 \models \mathsf{N}\,f$, et donc $\pi', m \models \mathsf{N}\,f$. Comme $f \equiv \mathsf{N}\,f$, on a: $\pi', m \models f$. f est donc une formule du futur. $\square$

En conséquence, il est clair que l'ajout du *Now* à une logique n'a de sens que si cette logique contient des opérateurs du passé. Enfin, l'opérateur N permet d'établir un lien entre l'équivalence initiale et l'équivalence globale:

Proposition 5.2.5 *$f \equiv_i g$ ssi $\mathsf{N}f \equiv \mathsf{N}g$*

Preuve L'équivalence $\mathsf{N}f \equiv_i f$ de la proposition 5.2.3 permet de déduire la preuve facilement. $\square$

Les équivalences 7 et 8 de la proposition 5.2.3 concernant les opérateurs du passé dans la portée du N ont, comme conséquence directe, le lemme suivant:

Lemme 5.2.6 *Soit L une logique de la forme $B(C_1, \ldots, C_n)$, $L(C_1, \ldots, C_n)$ ou $B(L(C_1, \ldots, C_n))$ avec $C_i \in \{ \mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \wedge, \neg\}$.*

Si f est une formule séparée de L, alors $\mathsf{N}\, f$ est équivalente à une formule pur-futur de L.

Preuve On utilise la transformation Φ définie dans la preuve du théorème 4.3.3. Φ associe à toute formule séparée une formule pur-futur équivalente initialement. Etant donnée une formule séparée f, on sait que $f \equiv_i \Phi(f)$, dont on déduit $\mathsf{N}\, f \equiv \mathsf{N}\, \Phi(f)$.

$\Phi(f)$ étant une formule pur-futur, on a: $\mathsf{N}\Phi(f) \equiv \Phi(f)$. D'où $\mathsf{N}f \equiv \Phi(f)$. $\square$

Nous allons maintenant étudier les effets, en terme d'expressivité, de l'introduction de l'opérateur N dans les logiques avec passé présentées au chapitre précédent.

5.3 Comparaison d'expressivité

La première question est de savoir si les résultats positifs obtenus pour $CTL + \mathsf{F}^{-1}$, $CTL^+ + \mathsf{F}^{-1}$, $ECTL^+ + \mathsf{F}^{-1}$, CTL_p ... restent vrais si on ajoute l'opérateur N. Nous avons le théorème suivant:

Théorème 5.3.1 *Soit L une logique de la forme $B(C_1, \ldots, C_n)$, $L(C_1, \ldots, C_n)$ ou $B(L(C_1, \ldots, C_n))$ avec $C_i \in \{ \mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \wedge, \neg\}$.*

Si L est séparable ($L \preceq_g L^{sep}$), alors on a: $L + \mathsf{N} \preceq_g L$.

Preuve Soit f une formule de $L + \mathsf{N}$. On procède par induction sur la hauteur de N dans f. Si f ne contient pas d'opérateur N, alors f est une formule de L. Considérons maintenant le cas où f est de la forme $\mathsf{N}\, g$. On sait par h.i. que g est équivalente à une formule $\tilde{g}$ de L et L étant séparable, $\tilde{g}$ est équivalente à une formule séparée $\tilde{g}^{sep}$. On a donc $f \equiv \mathsf{N}\, \tilde{g}^{sep}$. D'après le lemme 5.2.6, on en déduit que f est équivalente à une formule pur-futur de L. $\square$

Nous avons donc le corollaire suivant:

Corollaire 5.3.2 *Soit L une logique de la forme $B(C_1, \ldots, C_n)$, $L(C_1, \ldots, C_n)$ ou $B(L(C_1, \ldots, C_n))$ avec $C_i \in \{ \mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \wedge, \neg\}$.*

Si la logique L est séparable ($L \preceq_g L^{sep}$), alors $L + \mathsf{N}$ est séparable.

Preuve car: $L + \mathsf{N} \preceq_g L \preceq_g L^{sep}$ $\square$

En conséquence, on peut traduire $NCTL^*$ dans CTL^* et étendre les logiques $CTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$ (et donc $CTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$) et $ECTL^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$ avec l'opérateur N sans perdre la possibilité de les traduire dans respectivement CTL et $ECTL^+$:

Corollaire 5.3.3

- $NCTL^* \equiv PCTL^* \equiv_i CTL^*$
- $CTL + F^{-1}, F_s^{-1}, N \equiv CTL + F^{-1}, F_s^{-1} \equiv_i CTL$
- $CTL^+ + F^{-1}, F_s^{-1}, N \equiv CTL^+ + F^{-1}, F_s^{-1} \equiv_i CTL^+$
- $ECTL^+ + F^{-1}, F_s^{-1}, N \equiv ECTL^+ + F^{-1}, F_s^{-1} \equiv_i ECTL^+$

L'introduction de l'opérateur N aux logiques CTL_p, CTL_p^+ et $ECTL_p^+$ sera traitée à la section suivante car la définition de ces logiques étant différente de celles des fragments classiques, l'ajout de l'opérateur N demande un traitement particulier.

En ce qui concerne les logiques non séparables, l'ajout de l'opérateur N ne peut pas rendre ces logiques séparables puisque $L \subseteq L + \mathsf{N}$. Dans certains cas, l'opérateur N peut augmenter l'expressivité de ces logiques. Nous allons maintenant présenter ces résultats intéressants d'un point de vue théorique. Il y a d'abord le résultat suivant:

Théorème 5.3.4 $UB + F^{-1} + N \equiv_i CTL$

Preuve $UB + \mathsf{F}^{-1}, \mathsf{N} \preceq_i CTL$ est une conséquence de $CTL + \mathsf{F}^{-1}, \mathsf{N} \preceq_i CTL$. D'autre part, étant donnée une formule f de CTL, on peut construire une formule $\tilde{f}$ de $UB + \mathsf{F}^{-1}, \mathsf{N}$ équivalente à f. Le seul opérateur de CTL absent de $UB + \mathsf{F}^{-1}, \mathsf{N}$ est $\mathsf{E\_U}$ (car $\mathsf{A\_U}$ peut être exprimé à l'aide de EG et $\mathsf{E\_U}$). Nous avons:

$$\text{si } f, g \in CTL, \text{ Alors } \mathsf{E}\, f\ \mathsf{U}\ g \equiv \mathsf{N}\,\mathsf{EF}\,(\mathsf{G}^{-1} f \wedge \mathsf{EX}\, g)$$

La condition $f, g \in CTL$ garantit qu'il s'agit bien de formules du futur et que l'on peut donc les placer dans la portée d'un opérateur N sans changer leur valeur de vérité. Avec une stratégie Top-Down, on peut donc transformer toutes les sous-formules $\mathsf{E}\, f\ \mathsf{U}\ g$ de f de manière à éliminer les opérateurs $\mathsf{E\_U}$. Finalement on obtient bien une formule de $UB + \mathsf{F}^{-1} + \mathsf{N}$ équivalente à f. $\square$

Au chapitre précédent, nous avons vu que $B(L(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})) \prec_i ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$, nous allons utiliser une méthode très proche pour montrer que $CTL^* \preceq_g CTL + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$:

Théorème 5.3.5 $CTL^* \preceq_g ECTL + X^{-1}, S, N$

Preuve Soit f une formule de CTL^*. On procède par induction sur la hauteur de quantificateur E dans f. Si f ne contient aucun E, alors f est une combinaison booléenne de propositions atomiques et donc une formule de $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$.

Considérons une formule f de la forme $\mathsf{E}\, g$. Par h.i. on sait que toutes les sous-formules $\mathsf{E}\, h_i$ de g sont équivalentes à des formules ψ_i de $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$. Les ψ_i sont donc des formules d'état et nous allons les remplacer par des variables x_i. On obtient ainsi le contexte $\tilde{g}$:

$$f \equiv \mathsf{E}\, \tilde{g}[\psi_1, \dots, \psi_n] \qquad \text{avec } \forall i,\ \ \psi_i \in ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$$

$\tilde{g}$ est un contexte linéaire (sans quantificateur E). On peut donc appliquer le théorème 2.8.11 de Manna et Pnueli, on sait que $\tilde{g}[x_1, \dots, x_n]$ est initialement équivalent à $\bigwedge_i \Big(\mathsf{GF} p_i[x_1, \dots, x_n] \vee \mathsf{FG} q_i[x_1, \dots, x_n]\Big)$ où $p_i[x_1, \dots, x_n]$ et $q_i[x_1, \dots, x_n]$ sont des contextes pur-passé. On a donc:

$$\tilde{g} \;\equiv\; \mathsf{N} \bigwedge_i \Big(\mathsf{G}\,\mathsf{F}\, p_i[x_1, \dots, x_n] \,\vee\, \mathsf{F}\,\mathsf{G}\, q_i[x_1, \dots, x_n]\Big)$$

$$f \;\equiv\; \mathsf{N}\,\mathsf{E} \bigwedge_i \Big(\mathsf{G}\,\mathsf{F}\, p_i[\psi_1, \dots, \psi_n] \,\vee\, \mathsf{F}\,\mathsf{G}\, q_i[\psi_1, \dots, \psi_n]\Big) \quad \text{car } \ \mathsf{E}\,\mathsf{N}\, f \equiv \mathsf{N}\,\mathsf{E}\, f$$

Les ψ_i étant des formules de $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$ et comme $\mathsf{G}\,\mathsf{F} = \overset{\infty}{\mathsf{F}}$ et $\mathsf{F}\,\mathsf{G} = \overset{\infty}{\mathsf{G}}$, on en déduit que f est équivalente à une formule de $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$. □

On en déduit donc:

Théorème 5.3.6 $ECTL + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N} \equiv_i CTL^*$

Preuve Puisque $CTL^* \preceq_g ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N}$ et $ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}, \mathsf{N} \subseteq NCTL^* \equiv_i CTL^*$. □

Au chapitre 4, nous avons montré que la logique $ECTL + \mathsf{F}^{-1}$ est plus expressive que $ECTL$ et qu'elle se traduit initialement dans une logique L^+ moins expressive que $ECTL^+$. L'ajout de l'opérateur N à $ECTL + \mathsf{F}^{-1}$ ne permet pas d'atteindre l'expressivité de $ECTL^+$, car nous avons:

Lemme 5.3.7 $ECTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N} \;\equiv_i\; L^+ \;\prec_i\; ECTL^+$

Preuve Montrons que $ECTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N} \preceq_i L^+$: comme $ECTL \subseteq L^+$, il est clair que $ECTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N} \preceq_g L^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N}$. Or $L^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}$ étant séparable, on a bien: $L^+ + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N} \preceq_i L^+$.

Montrons que $L^+ \preceq_i ECTL + \mathsf{F}^{-1}, \mathsf{F}_s^{-1}, \mathsf{N}$: étant donnée une formule f de L^+, nous allons construire une formule de $ECTL + \mathsf{F}^{-1}, \mathsf{N}$ équivalente à f. Les seules sous-formules de f à transformer sont celles de la forme: $\mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ g \wedge \mathsf{G}\,h)$, et on a:

$$\text{si } f, g \in L^+,\ \ \mathsf{E}\,(\overset{\infty}{\mathsf{F}}\ g \wedge \mathsf{G}\,h) \equiv \mathsf{N}\,\mathsf{E}\,\overset{\infty}{\mathsf{F}}\ (g \wedge \mathsf{G}^{-1} h) \tag{5.59}$$

L'équivalence (5.59) est valide non seulement si f et g dont des formules de L^+, mais plus généralement pour toutes formules f et g du futur. Avec une stratégie Top-Down, on peut construire une formule $\tilde{f}$ de $ECTL + \mathsf{F}^{-1} + \mathsf{N}$ équivalente à f en utilisant la transformation (5.59). $\square$

On peut noter que si la conjecture 4.4.28 du chapitre précédent ($ECTL + \mathsf{F}_s^{-1} \prec_i L^+$) était prouvée, cela montrerait que l'introduction du N dans $ECTL + \mathsf{F}_s^{-1}$ augmente l'expressivité.

5.4 L'ajout de l'opérateur N à CTL_p, CTL_p^+ et $ECTL_p^+$

A partir des logiques CTL_p, CTL_p^+ et $ECTL_p^+$, l'ajout de l'opérateur Now va nous conduire à définir trois nouvelles logiques ($\boldsymbol{P}CTL$, $\boldsymbol{P}CTL^+$ et $\boldsymbol{P}ECTL^+$).

5.4.1 La logique $\boldsymbol{P}CTL$

La logique CTL_p étant séparable, on peut, d'après le théorème 5.3.1, ajouter l'opérateur N (i.e. étendre la grammaire de CTL_p de la définition 4.5.1 avec $\mathsf{N}\,f$ et considérer des formules φ et ψ de $CTL + \mathsf{F}_s^{-1}, \mathsf{N}$) sans accroître l'expressivité de CTL_p (on peut toujours traduire les formules ainsi obtenue dans CTL_p). Mais les raisons qui nous ont conduit à définir CTL_p, incitent à introduire plus finement l'opérateur N : CTL_p correspond à la logique $CTL + \mathsf{F}_s^{-1}$ dans laquelle on autorise l'emploi de *Since* hors de la portée d'un X^{-1} ou d'un S, et de *Previous* dans les parties droites de E_U. Comme pour toute formule f de CTL_p, $\mathsf{N}\,f$ est équivalente à une formule CTL, on peut donc autoriser $\mathsf{N}\,f$ dans la portée de tout opérateur de $CTL + \mathsf{F}_s^{-1}$. On définit ainsi la logique $\boldsymbol{P}CTL$:

Définition 5.4.1 *La logique* $\boldsymbol{P}CTL$

$$\boldsymbol{P}CTL \ni f, g \quad ::= a \mid f \wedge g \mid \neg f \mid \mathsf{EX}\,f \mid \mathsf{E}\,\varphi\,\mathsf{U}\,f \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\psi \mid \varphi\,\mathsf{S}\,\psi \mid \mathsf{X}^{-1} f \mid \mathsf{N}\,f$$

$$\textit{avec } \varphi, \psi \quad ::= a \mid \varphi \wedge \psi \mid \neg\varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{E}\,\varphi\,\mathsf{U}\,\psi \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\psi \mid \mathsf{F}_s^{-1}\varphi \mid \mathsf{N}\,f$$

La différence entre $\boldsymbol{P}CTL$ et la logique dénotée par $CTL_p + \mathsf{N}$ réside donc dans la dernière règle de la seconde grammaire: $\mathsf{N}\,f$ et non pas $\mathsf{N}\,\varphi$. Clairement on a: $CTL_p + \mathsf{N} \subset \boldsymbol{P}CTL$.

Nous avons le lemme suivant:

Lemme 5.4.2 $\boldsymbol{P}CTL \equiv CTL_p$.

Preuve On utilise le fait que pour tout f de CTL_p, il existe une formule $\tilde{f}$ de CTL équivalente initialement à f. On a donc $\mathsf{N}\, f \equiv \tilde{f}$. Avec une stratégie bottom-up, on peut donc transformer toute formule de $\boldsymbol{P}CTL$ en une formule équivalente de CTL_p. □

On en déduit directement un lemme de séparation pour $\boldsymbol{P}CTL$:

Lemme 5.4.3 *Toute formule de* $\boldsymbol{P}CTL$ *est équivalente à une formule séparée (de* $\boldsymbol{P}CTL$*).*

En conséquence, nous avons le résultat suivant:

Théorème 5.4.4 $\boldsymbol{P}CTL \preceq_i CTL$

Preuve Utilisation du théorème 4.3.3. □

5.4.2 La logique $\boldsymbol{P}CTL^+$

Nous allons procéder de manière identique pour l'insertion de l'opérateur N dans CTL_p^+. Nous définissons la logique $\boldsymbol{P}CTL^+$ de la manière suivante:

Définition 5.4.5 *La logique* $\boldsymbol{P}CTL^+$

$$\begin{aligned}
\boldsymbol{P}CTL^+ \ni f, g \quad &::= a \mid f \wedge g \mid \neg f \mid \mathsf{E}\, f_c \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \mid \mathsf{N} f \\
f_c, g_c \quad &::= f_c \wedge g_c \mid f_c \vee g_c \mid \neg\varphi_c \mid \varphi\, \mathsf{U} f \mid \mathsf{X} f \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \mid \mathsf{N} f_c \\
\varphi, \psi \quad &::= a \mid \varphi \wedge \psi \mid \neg\varphi \mid \mathsf{E}\, \varphi_c \mid \mathsf{F}^{-1}\varphi \mid \mathsf{F}_s^{-1}\varphi \mid \mathsf{N} f \\
\varphi_c, \psi_c \quad &::= \varphi_c \wedge \psi_c \mid \neg\varphi_c \mid \varphi\, \mathsf{U}\, \psi \mid \mathsf{X}\varphi \mid \mathsf{F}^{-1}\varphi \mid \mathsf{F}_s^{-1}\varphi \mid \mathsf{N} f_c
\end{aligned}$$

On a clairement: $CTL_p^+ + \mathsf{N} \subset \boldsymbol{P}CTL^+$.

De manière identique à $\boldsymbol{P}CTL$, nous avons le lemme suivant qui s'appuie sur le fait que $\mathsf{N}\, f$ (avec $f \in CTL_p^+$) est équivalent à une formule de CTL:

Lemme 5.4.6 $\boldsymbol{P}CTL^+ \equiv CTL_p^+$

Puisque CTL_p^+ est séparable, on a:

Lemme 5.4.7 *Toute formule de $\boldsymbol{PCTL^+}$ est équivalente à une formule séparée (de $\boldsymbol{PCTL^+}$).*

On en déduit donc:

Théorème 5.4.8 $\boldsymbol{PCTL^+} \preceq_i CTL$

5.4.3 La logique $\boldsymbol{PECTL^+}$

L'opérateur N permet aussi d'étendre la logique $ECTL_p^+$. Nous définissons la logique $\boldsymbol{PECTL^+}$ de la manière suivante:

Définition 5.4.9 *La logique $\boldsymbol{PECTL^+}$*

$$\boldsymbol{PECTL^+} \ni f, g \quad ::= \ a \mid f \wedge g \mid \neg f \mid \mathsf{E} f_c \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \mid \mathsf{N} f$$

$$f_c, g_c \quad ::= \ f_c \wedge g_c \mid f_c \vee g_c \mid \neg\varphi_c \mid \varphi\, \mathsf{U} f \mid \mathsf{X} f \mid \overset{\infty}{\mathsf{F}} \varphi \mid \varphi\, \mathsf{S}\, \psi \mid \mathsf{X}^{-1} f \mid \mathsf{N} f_c$$

$$\varphi, \psi \quad ::= \ a \mid \varphi \wedge \psi \mid \neg\varphi \mid \mathsf{E}\varphi_c \mid \mathsf{F}^{-1}\varphi \mid \mathsf{F}_s^{-1}\varphi \mid \mathsf{N} f$$

$$\varphi_c, \psi_c \quad ::= \ \varphi_c \wedge \psi_c \mid \neg\varphi_c \mid \varphi\, \mathsf{U}\, \psi \mid \mathsf{X}\varphi \mid \overset{\infty}{\mathsf{F}} \varphi \mid \mathsf{F}^{-1}\varphi \mid \mathsf{F}_s^{-1}\varphi \mid \mathsf{N} f_c$$

On a clairement: $ECTL_p^+ + \mathsf{N} \subset \boldsymbol{PECTL^+}$. De plus, comme pour $\boldsymbol{PCTL}$ et $\boldsymbol{PCTL^+}$, on a le lemme suivant:

Lemme 5.4.10 $\boldsymbol{PECTL^+} \equiv ECTL_p^+$

$ECTL_p^+$ étant séparable, on a:

Lemme 5.4.11 *Toute formule de $\boldsymbol{PECTL^+}$ est équivalente à une formule séparée (de $\boldsymbol{PECTL^+}$).*

D'où on déduit:

Théorème 5.4.12 $\boldsymbol{PECTL^+} \preceq_i ECTL^+$

$$CTL^* \equiv_i NCTL^* \equiv ECTL^+ + \mathsf{S}, \mathsf{X}^{-1}, \mathsf{N}$$
$$\downarrow$$
$$ECTL^+ + \mathsf{S}, \mathsf{X}^{-1}$$
$$\downarrow$$
$$\boldsymbol{P}ECTL^+ \equiv_i ECTL^+$$
$$\downarrow$$
$$ECTL + \mathsf{F}_s^{-1}, \mathsf{N}$$
$$\downarrow$$
$$ECTL + \mathsf{F}_s^{-1}$$
$$\downarrow$$
$$ECTL$$
$$\downarrow$$
$$\boldsymbol{P}CTL^+ \equiv_i \boldsymbol{P}CTL \equiv_i CTL \equiv_i UB + \mathsf{F}_s^{-1}, \mathsf{N}$$

Figure 5.1: Une hiérarchie des logiques avec *Now*.

5.5 Conclusion

L'ensemble des résultats concernant les logiques avec *Now* sont résumés par la figure 5.1.

Si les conjectures 4.3.13 ($CTL^* \not\preceq_i ECTL^+ + \mathsf{X}^{-1}, \mathsf{S}$) et 4.4.28 ($L^+ \not\preceq_i ECTL + \mathsf{F}_s^{-1}$) sont vraies, le schéma 5.1 est complet (les seules flèches pouvant être ajoutées sont obtenues par transitivité).

En conclusion, l'opérateur N peut donc être ajouté aux logiques séparables que nous avons étudiées au chapitre précédent en gardant la possibilité de traduire les formules ainsi obtenues dans les fragments pur-futur CTL ou $ECTL^+$. Dans l'introduction, nous avons expliqué l'intérêt de cet opérateur pour la spécification: la possibilité de créer des contextes temporels où le passé est oublié permet d'énoncer aisément certaines propriétés avec des opérateurs du passé. De plus, l'opérateur N associé au quantificateur E permet d'exprimer le quantificateur E proposé dans [HT87] (où le passé n'est pas cumulatif), la logique $NCTL^*$ englobe donc à la fois la logique $PCTL^*$ décrite au chapitre 2 et la logique $PCTL^*$ de [HT87]. Enfin, l'opérateur N permet d'établir un nouveau lien entre l'équivalence initiale et l'équivalence globale.

On peut compléter le schéma 4.10 représentant pour chaque logique du futur L de la hiérarchie de [Eme90], un ensemble de logiques avec passé pouvant se traduire initialement dans L (schéma 5.2).

Le fragment $\boldsymbol{P}CTL^+$ est donc le plus grand fragment, présenté dans cette étude, contenant du passé et qui peut être traduit initialement dans la logique CTL. D'autre part, $\boldsymbol{P}ECTL^+$ représente le plus grand fragment avec passé (présenté dans ce travail) qui peut être traduit initialement dans $ECTL^+$.

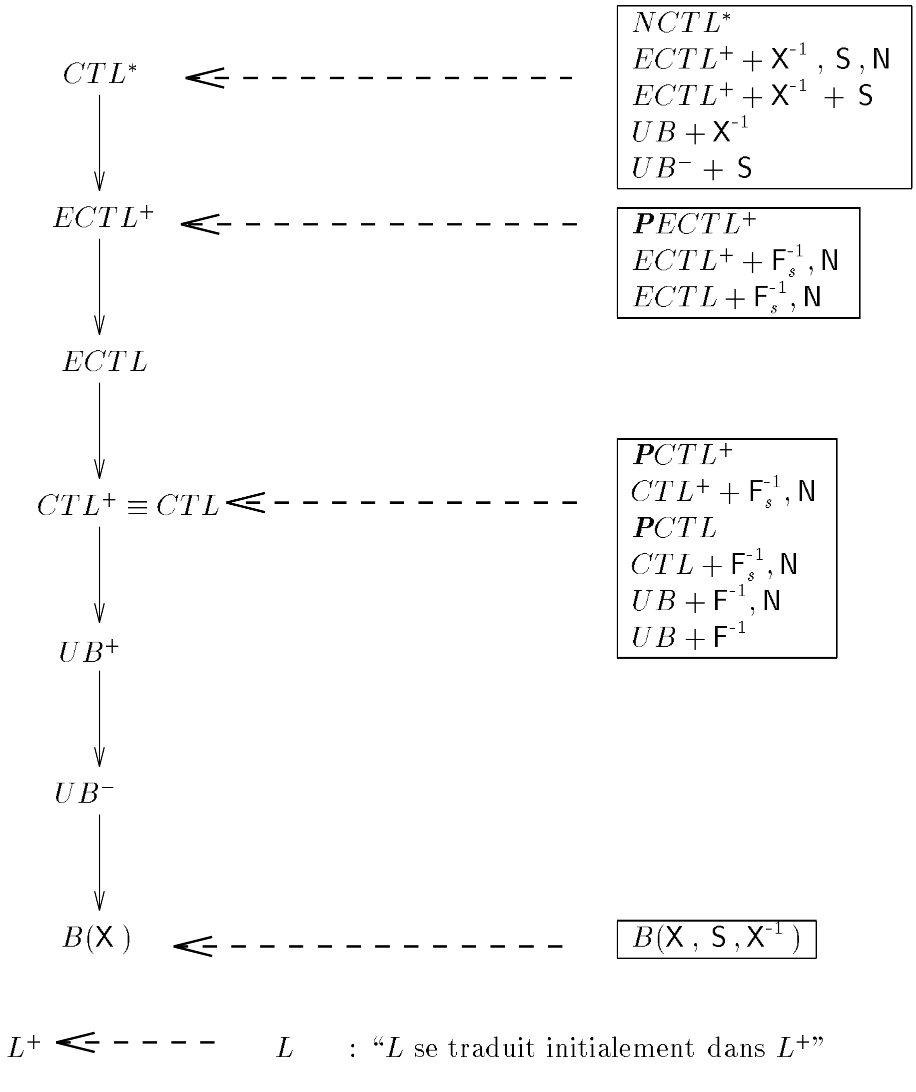


Figure 5.2: Traduction de logiques avec S, X^{-1} et/ou N dans les fragments de CTL^*

Chapitre 6

Logiques sans *Next*

6.1 Introduction et motivation

L'opérateur *Next* a été introduit en tant qu'opérateur de base dans [MP79] mais il est souvent défini à partir de l'opérateur *Until* strict. Toutes les logiques que nous avons considérées jusqu'ici, comportaient cet opérateur. Pourtant l'opérateur *Next* pose des problème lorsque l'on s'intéresse à la spécification des systèmes réactifs. En effet, Lamport [Lam83] a expliqué que la logique temporelle devait être un formalisme pour raisonner à un certain niveau d'abstraction et ne pas s'intéresser de trop près aux modèles sous-jacents, or l'opérateur *Next* permet de distinguer des modèles qui pourraient être considérés comme équivalents. La notion d'état suivant dépend directement du modèle choisi: dans l'exemple de l'ascenseur nous avons choisi de faire une spécification très abstraite qui ne suppose, par exemple, aucun choix particulier d'implémentation, et nous n'avons pas eu besoin de l'opérateur *Next*.

Nous avons donc choisi de présenter dans ce chapitre un ensemble de résultats concernant les logiques dans lesquelles l'opérateur *Next* n'est pas autorisé. Nous verrons qu'une part importante de ceux-ci est obtenue de manière similaire aux résultats des chapitres 4 et 5, nous ne détaillerons pas ces preuves et nous nous contenterons de renvoyer aux preuves des chapitres précédents en mentionnant les éventuelles petites modifications.

Nous allons maintenant considérer l'ajout du passé dans les logiques $B(\mathsf{F})$, $B(\mathsf{U})$, $B(\mathsf{U},\wedge,\neg)$, $B(\mathsf{U},\overset{\infty}{\mathsf{F}})$ et $B(\mathsf{U},\overset{\infty}{\mathsf{F}},\wedge,\neg)$ (i.e. respectivement $UB\backslash\mathsf{X}$, $CTL\backslash\mathsf{X}$, $CTL^{+}\backslash\mathsf{X}$, $ECTL\backslash\mathsf{X}$ et $ECTL^{+}\backslash\mathsf{X}$). Toutes ces logiques sont des fragments de $CTL^{*}\backslash\mathsf{X}$.

L'objectif de ce chapitre est donc la recherche de fragment avec passé pouvant se traduire dans les logiques $B(\mathsf{U})$ ou $B(\mathsf{U},\overset{\infty}{\mathsf{F}},\wedge,\neg)$.

Nous allons suivre une démarche similaire à celle du chapitre 4, néanmoins nous ne considérerons pas le cas du temps linéaire puisque les résultats obtenus pour les logiques LTL, $PLTL$ et PTL peuvent être trivialement étendus de manière à obtenir: $L(\mathsf{F}) \prec_i L(\mathsf{F},\mathsf{F}^{-1}) \prec_i L(\mathsf{U})$.

6.2 Logiques du temps arborescent avec l'opérateur $Since$

Nous avons vu au chapitre 4 que l'ajout de l'opérateur $Since$ à la logique UB^{-} permet d'exprimer la propriété $\mathsf{E}\,(s \vee a\,\mathsf{U}\,b)\,\mathsf{U}\,r$ qui n'est pas exprimable avec $ECTL^{+}$. En fait nous allons voir, qu'il est possible d'exprimer cette propriété sans l'opérateur X :

Lemme 6.2.1

$$
\begin{aligned}
\mathsf{E}(s \vee a\,\mathsf{U}\,b)\,\mathsf{U}\,r \quad \equiv_i \\
& r \vee \mathsf{EF}\Big[b \wedge b\,\mathsf{S}\Big(a\,\mathsf{S}\big[r \wedge a \wedge \neg\mathsf{F}^{-1}\big(\neg a \wedge \neg b \wedge (\neg b)\,\mathsf{S}\,(\neg b \wedge \neg s)\big)\big]\Big)\Big] \\
& \vee \mathsf{EF}\Big[r \wedge b \wedge (r \wedge b)\,\mathsf{S}\big[\neg\mathsf{F}^{-1}\big(\neg a \wedge \neg b \wedge (\neg b)\,\mathsf{S}\,(\neg b \wedge \neg s)\big)\big]\Big] \\
& \vee \mathsf{EF}\Big[r \wedge (r)\,\mathsf{S}\big[\neg\mathsf{F}^{-1}\big(\neg a \wedge \neg b \wedge (\neg b)\,\mathsf{S}\,(\neg b \wedge \neg s)\big) \wedge \neg(\neg b)\,\mathsf{S}\,(\neg b \wedge \neg s)\big]\Big]
\end{aligned}
$$

Preuve Il est plus difficile de s'assurer intuitivement de la validité de cette équivalence que de celle du lemme 4.3.7. En fait, nous avons remplacé les sous-formules de la forme $\mathsf{EF}\,(f \wedge \mathsf{EX}\,g)$ par $\mathsf{EF}\,(g \wedge (g\,\mathsf{S}\,f))$. Ces deux sous-formules ne sont pas équivalentes : en fait on a $\models \mathsf{EF}\,(f \wedge \mathsf{EX}\,g) \;\Rightarrow\; \mathsf{EF}\,(g \wedge (g\,\mathsf{S}\,f))$. En effet, si il existe dans le futur un état vérifiant $g \wedge f$, alors l'état courant vérifie $\mathsf{EF}\,(g \wedge (g\,\mathsf{S}\,f))$ et non $\mathsf{EF}\,(f \wedge \mathsf{EX}\,g)$, cependant un autre terme de la disjonction couvre alors ce cas. $\square$

Ce résultat montre qu'il n'est pas possible d'ajouter l'opérateur S aux logiques $B(\mathsf{F})$, $B(\mathsf{U})$, $B(\mathsf{U},\overset{\infty}{\mathsf{F}})$ et $B(\mathsf{U},\overset{\infty}{\mathsf{F}},\wedge,\neg)$ et de traduire les logiques enrichies dans $B(\mathsf{U},\overset{\infty}{\mathsf{F}},\wedge,\neg)$.

Concernant l'opérateur X^{-1}, on peut penser que celui-ci ne correspond pas au choix qui nous a conduit à étudier les logiques sans next puisque les objections que l'on peut formuler à l'égard de X sont tout aussi valables pour X^{-1}. De plus, nous allons voir que l'ajout de cet opérateur ou de F_s^{-1} à la logique $B(\mathsf{F})$ permet d'exprimer initialement la propriété $\mathsf{EX}\,a$ qui est inexprimable avec $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$:

Lemme 6.2.2

$$\begin{aligned} \mathsf{EF}\left(a \wedge \mathsf{X}^{-1}\top \wedge \neg\mathsf{X}^{-1}\mathsf{X}^{-1}\top\right) &\equiv_i \mathsf{EX}\,a \\ \mathsf{EF}\left(a \wedge \mathsf{F}_s^{-1}\top \wedge \neg\mathsf{F}_s^{-1}\mathsf{F}_s^{-1}\top\right) &\equiv_i \mathsf{EX}\,a \end{aligned}$$

Mais la formule $\mathsf{EX}\,a$ est inexprimable avec $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$:

Lemme 6.2.3 *Il n'existe aucune formule de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ équivalente à $\mathsf{EX}\,a$.*

Preuve En fait tout ce qu'on exprime avec $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ est compatible avec la stuttering bisimulation [BCG88], ce qui n'est pas le cas de $\mathsf{EX}\,a$. $\square$

Des lemmes 6.2.2 et 6.2.3 on peut déduire qu'il n'est pas possible d'ajouter les opérateurs X^{-1} et F_s^{-1} à la logique $B(\mathsf{F})$ si l'on veut obtenir une logique qui puisse se traduire initialement dans $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$.

Avant de considérer, l'ajout de l'opérateur F^{-1}, nous pouvons considérer le lemme suivant concernant $PCTL^*\backslash\mathsf{X}$:

Lemme 6.2.4 *Toute formule de $PCTL^*\backslash\mathsf{X}$ est équivalente à une formule séparée de $PCTL^*\backslash\mathsf{X}$.*

Preuve Il suffit de reprendre la preuve utilisée pour le lemme 4.3.4 énonçant que $PCTL^*$ est séparable, en considérant le théorème 2.8.9 établi dans [MMKR93] qui énonce que $L(\mathsf{U}, \mathsf{S})$ est séparable, à la place du théorème de séparation de Gabbay. $\square$

On en déduit donc:

Théorème 6.2.5 $PCTL^*\backslash\mathsf{X} \equiv_i CTL^*\backslash\mathsf{X}$

Nous allons donc nous intéresser d'une part à l'ajout de l'opérateur F^{-1} dans les différentes logiques du temps arborescent, et d'autre part à une insertion restreinte de l'opérateur S comme nous nous l'avons fait pour la définition CTL_p ou $ECTL_p^+$.

6.3 Logiques du temps arborescent avec l'opérateur F^{-1}

Comme pour le temps linéaire, l'ajout de l'opérateur F^{-1} augmente l'expressivité de $B(\mathsf{F})$ mais reste en deça de l'expressivité de $B(\mathsf{U})$:

Théorème 6.3.1 $B(\mathsf{F}) \prec_i B(\mathsf{F}, \mathsf{F}^{-1}) \prec_i B(\mathsf{U})$

Preuve Clairement, on a: $B(\mathsf{F}) \preceq_i B(\mathsf{F}, \mathsf{F}^{-1})$.

Nous savons que la formule $\mathsf{E}\, a\, \mathsf{U}\, b$ est inexprimable avec $B(\mathsf{F})$, or nous avons: $\mathsf{E}\, a\, \mathsf{U}\, b \equiv_i \mathsf{EF}\, (b \wedge \mathsf{G}^{-1}(a \vee b))$. On en déduit $B(\mathsf{F}) \prec_i B(\mathsf{F}, \mathsf{F}^{-1})$.

Comme la logique $UB + \mathsf{F}^{-1}$ ne peut pas exprimer la formule $\mathsf{EF}\, (a \wedge \mathsf{E}\, b\, \mathsf{U}\, c)$ qui appartient à $B(\mathsf{U})$, il en découle que $B(\mathsf{F}, \mathsf{F}^{-1}) \not\succeq_i B(\mathsf{U})$. De plus $B(\mathsf{F}, \mathsf{F}^{-1}) \preceq_i B(\mathsf{U})$ est une conséquence immédiate du théorème 6.3.3 qui énonce $B(\mathsf{U}, \mathsf{F}^{-1}) \preceq_i B(\mathsf{U})$. □

Concernant l'ajout de l'opérateur F^{-1} à la logique $B(\mathsf{U})$, nous avons le résultat suivant:

Lemme 6.3.2 *Toute formule de* $B(\mathsf{U}, \mathsf{F}^{-1})$ *est équivalente à une formule séparée de* $B(\mathsf{U}, \mathsf{F}^{-1})$.

Preuve La preuve se fait de manière identique à celle du lemme de séparation de CTL. En effet, les équivalences (4.47) et (4.49) que l'on utilise comme règles de séparation pour extraire les opérateurs F^{-1} de la portée des opérateurs $\mathsf{E\_U}$ et EG n'utilisent aucun opérateur EX. On peut donc utiliser la même preuve pour séparer les formules de $B(\mathsf{U}, \mathsf{F}^{-1})$. □

La logique $B(\mathsf{U}, \mathsf{F}^{-1})$ se traduit donc initialement dans $B(\mathsf{U})$:

Théorème 6.3.3 $B(\mathsf{U}, \mathsf{F}^{-1}) \equiv_i B(\mathsf{U})$

Preuve Immédiate à partir du lemme 6.3.2. □

Le résultat précédent peut être étendu sans difficulté à la logique $B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg)$:

Lemme 6.3.4 *Toute formule de* $B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg)$ *est équivalente à une formule séparée de* $B(\mathsf{U}, \mathsf{F}^{-1})$.

Preuve Soit f une formule de $B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg)$. On considère la transformation utilisée pour montrer que toute formule de $CTL^{+} + \mathsf{F}^{-1}$ est équivalente à une formule de $CTL+\mathsf{F}^{-1}$ (cf. théorème 4.4.14): si on applique cette transformation à la formule f (qui ne contient pas d'opérateur X), on obtient une formule $\tilde{f}$ de $B(\mathsf{U}, \mathsf{F}^{-1})$ (puisque cette transformation n'insère aucun EX) équivalente à f.

Il suffit alors d'utiliser le lemme 6.3.2 pour obtenir une formule séparée de $B(\mathsf{U}, \mathsf{F}^{-1})$ équivalente à f. $\square$

Comme pour $ECTL$, l'ajout de l'opérateur F^{-1} augmente l'expressivité de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}})$, sans toutefois atteindre celle de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$, et on a:

Théorème 6.3.5 $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}) \prec_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \prec_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$

Preuve Clairement, on a: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}) \preceq_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1})$.

$B(\mathsf{U}, \overset{\infty}{\mathsf{F}}) \prec_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1})$ puisque la formule $\mathsf{E}\overset{\infty}{\mathsf{F}}(a \wedge \mathsf{G}^{-1}b)$ appartient à $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1})$ et est inexprimable avec $ECTL$ et à fortiori avec $B(\mathsf{U}, \overset{\infty}{\mathsf{F}})$.

On a: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \preceq_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ car la transformation utilisée pour montrer $ECTL+\mathsf{F}^{-1} \preceq_i L^+$ (cf. théorème 4.4.22) peut être appliquée aux formules de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1})$ et on obtient alors: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \preceq_i L^+\backslash\mathsf{X}$ puisque cette transformation n'insère aucun opérateur EX. On en déduit donc: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \preceq_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$.

$B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \prec_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ puisque la formule $\mathsf{E}(\overset{\infty}{\mathsf{F}}\, a \wedge \overset{\infty}{\mathsf{F}}\, b)$ de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ est inexprimable avec $ECTL+\mathsf{F}^{-1}$ et à fortiori avec $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1})$. $\square$

Enfin, nous avons, le théorème concernant l'ajout de l'opérateur F^{-1} à la logique $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$:

Théorème 6.3.6 *Toute formule de* $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg)$ *est équivalente à une formule séparée de* $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg)$.

Preuve On peut facilement montrer que toute formule f de $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg)$ est équivalente à une formule du fragment $\mathcal{F}_{ECTL^+\backslash X} + \mathsf{F}^{-1}$ où $\mathcal{F}_{ECTL^+\backslash X}$ est défini par:

$$\mathcal{F}_{ECTL^+\backslash X} \ni f, g ::= a \mid f \wedge g \mid \neg f \mid \mathsf{E}\, f\, \mathsf{U}\, g \mid \mathsf{A}\, f\, \mathsf{U}\, g \mid \mathsf{E}\,(\mathsf{G}\, f \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\, g_i)$$

D'autre part, on peut utiliser l'équivalence (4.53) comme règle de séparation pour extraire les opérateurs F^{-1} de la portée de l'opérateur $\mathsf{E}\,(\mathsf{G}\, f \wedge \bigwedge_i \overset{\infty}{\mathsf{F}}\, g_i)$. On peut alors séparer les formules de $\mathcal{F}_{ECTL^+\backslash X} + \mathsf{F}^{-1}$. $\square$

6.4 L'insertion restreinte de l'opérateur S

La définition de la logique CTL_p était motivée par le constat qu'il était possible dans certains cas, d'extraire un opérateur S de la portée d'un opérateur $\mathsf{E\_U}$. Nous allons constater qu'il en est de même lorsque nous considérons la logique $B(\mathsf{U}, \mathsf{S})$. Nous définissons donc la logique $CTL_p\backslash\mathsf{X}$:

Définition 6.4.1 *La logique* $CTL_p\backslash\mathsf{X}$

$$CTL_p\backslash\mathsf{X} \ni f,g ::= a \mid f \wedge g \mid \neg f \mid \mathsf{E}\varphi\,\mathsf{U}\,f \mid \mathsf{A}\varphi\,\mathsf{U}\,\psi \mid \varphi\,\mathsf{S}\,\psi$$

où $\varphi,\psi \in B(\mathsf{U}, \mathsf{F}^{-1})$

La logique $CTL_p\backslash\mathsf{X}$ correspond donc au fragment de CTL_p où l'on n'autorise pas les opérateurs EX, X^{-1} et F_s^{-1}. Clairement on a: $B(\mathsf{U}, \mathsf{F}^{-1}) \subset CTL_p\backslash\mathsf{X}$.

La logique $CTL_p\backslash\mathsf{X}$ est séparable:

Lemme 6.4.2 *Toute formule de* $CTL_p\backslash\mathsf{X}$ *est équivalente à une formule séparée (de* $CTL_p\backslash\mathsf{X}$*).*

Preuve Pour établir cette preuve nous ne pouvons pas utiliser les règles de séparation considérées dans la séparation de CTL_p car la règle $R'1$ utilise l'opérateur EX pour extraire l'opérateur S de la portée de l'opérateur $\mathsf{E\_U}$. Nous allons donc construire une nouvelle règle à partir de l'équivalence suivante:

$\mathsf{E}\left(\varphi\,\mathsf{U}\,(\alpha \wedge x\,\mathsf{S}\,y)\right) \equiv$ $\quad \mathsf{E}\,\varphi\,\mathsf{U}\,(\alpha \wedge y)$ $\quad \vee\,\mathsf{E}\,\varphi\,\mathsf{U}\left[\varphi \wedge y \wedge \mathsf{E}\,(\varphi \wedge y)\,\mathsf{U}\left(\mathsf{E}\,(\varphi \wedge x)\,\mathsf{U}\,(\alpha \wedge x)\right)\right]$ $\quad \vee\, x\,\mathsf{S}\,y \wedge \left(\alpha \vee \mathsf{E}\,(\varphi \wedge x)\,\mathsf{U}\,(\alpha \wedge x)\right)$	$R''1$

Pour construire $R''1$ à partir de $R'1$, on a remplacé la sous-formule $a \wedge \mathsf{EX}\,(\mathsf{E}c\,\mathsf{U}\,d)$ par $a \wedge \mathsf{E}\,a\,\mathsf{U}\,(\mathsf{E}\,c\,\mathsf{U}\,d)$. En général, cette substitution n'est pas correcte (au sens où ces deux formules ne sont pas équivalentes) car si l'état courant vérifie $a \wedge d$ alors la seconde formule est vraie à la différence de la première. Mais dans la formule de droite de $R''1$, le cas d'un état vérifiant $a \wedge d$ est pris en compte par le premier terme de la disjonction $(\mathsf{E}\,\varphi\,\mathsf{U}\,(\alpha \wedge y))$. L'équivalence $R''1$ est donc valide.

On peut alors prouver la séparation de $CTL_p\backslash\mathsf{X}$ de manière identique à celle de CTL_p en considérant les règles $R''1$ et $R'2$ ainsi que les règles utilisées pour $B(\mathsf{U}, \mathsf{F}^{-1})$. $\square$

On en déduit le théorème suivant:

Théorème 6.4.3 $CTL_p \backslash \mathsf{X} \ \equiv_i \ B(\mathsf{U})$

Nous pouvons aussi définir la logique $CTL_p^+ \backslash \mathsf{X}$ sur le même modèle:

Définition 6.4.4 *La logique* $CTL_p^+ \backslash \mathsf{X}$

$$
\begin{aligned}
CTL_p^+ \backslash \mathsf{X} \ni f, g \quad &::= a \mid f \wedge g \mid \neg f \mid \mathsf{E} f_c \mid \varphi \, \mathsf{S} \, \psi \\
f_c, g_c \quad &::= f_c \wedge g_c \mid f_c \vee g_c \mid \neg \varphi_c \mid \varphi \, \mathsf{U} f \mid \varphi \, \mathsf{S} \, \psi \\
B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg) \ni \varphi, \psi \quad &::= a \mid \varphi \wedge \psi \mid \neg \varphi \mid \mathsf{E} \varphi_c \mid \mathsf{F}^{-1} \varphi \\
\varphi_c, \psi_c \quad &::= \varphi_c \wedge \psi_c \mid \neg \varphi_c \mid \varphi \, \mathsf{U} \, \psi \mid \mathsf{F}^{-1} \varphi
\end{aligned}
$$

On a: $B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg) \subset CTL_p^+ \backslash \mathsf{X}$. De plus, on peut établir le lemme suivant:

Lemme 6.4.5 *Toute formule de* $CTL_p^+ \backslash \mathsf{X}$ *est équivalente à une formule séparée (de* $CTL_p \backslash \mathsf{X}$*).*

Preuve On utilise la même méthode que celle considérée pour montrer $CTL_p^+ \preceq_g CTL_p$. Etant donnée une formule f de $CTL_p^+ \backslash \mathsf{X}$, on obtient alors une formule $\tilde{f}$ de $CTL_p \backslash \mathsf{X}$ (car la transformation n'insère pas d'opérateur EX) équivalente à f. On utilise ensuite le lemme de séparation pour $CTL_p \backslash \mathsf{X}$. □

On en déduit donc:

Théorème 6.4.6 $CTL_p^+ \backslash \mathsf{X} \ \equiv_i \ B(\mathsf{U})$

Sur le même principe, on peut définir la logique $ECTL_p^+ \backslash \mathsf{X}$:

Définition 6.4.7 $ECTL_p^+ \backslash \mathsf{X}$

$$
\begin{aligned}
ECTL_p^+ \backslash \mathsf{X} \ni f, g \quad &::= a \mid f \wedge g \mid \neg f \mid \mathsf{E} f_c \mid \varphi \, \mathsf{S} \, \psi \\
f_c, g_c \quad &::= f_c \wedge g_c \mid f_c \vee g_c \mid \neg \varphi_c \mid \varphi \, \mathsf{U} f \mid \overset{\infty}{\mathsf{F}} \varphi \mid \varphi \, \mathsf{S} \, \psi \\
B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg) \ni \varphi, \psi \quad &::= a \mid \varphi \wedge \psi \mid \neg \varphi \mid \mathsf{E} \varphi_c \mid \mathsf{F}^{-1} \varphi \\
\varphi_c, \psi_c \quad &::= \varphi_c \wedge \psi_c \mid \neg \varphi_c \mid \varphi \, \mathsf{U} \, \psi \mid \overset{\infty}{\mathsf{F}} \varphi \mid \mathsf{F}^{-1} \varphi
\end{aligned}
$$

On a: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg) \subset ECTL_p^+\backslash\mathsf{X}$. De plus, on dispose d'un lemme de séparation pour $ECTL_p^+\backslash\mathsf{X}$:

Lemme 6.4.8 *Toute formule de $ECTL_p^+\backslash\mathsf{X}$ est équivalente à une formule séparée (de $ECTL_p^+\backslash\mathsf{X}$).*

Preuve Pour établir cette preuve, il suffit de montrer que $ECTL_p^+\backslash\mathsf{X} \preceq_g \mathcal{F}_{ECTL_p^+\backslash X}$ où $\mathcal{F}_{ECTL_p^+\backslash X}$ est le fragment de $ECTL_p^+\backslash\mathsf{X}$ où l'on a remplacé les combinaisons booléennes de formules de chemin dans la portée des E par l'opérateur $\mathsf{E}(\mathsf{G}\varphi\wedge\bigwedge_i \psi_i)$.

On peut alors utiliser la même méthode de séparation que pour $CTL_p\backslash\mathsf{X}$ en ajoutant la règle définie par l'équivalence (4.53) pour extraire les F^{-1} de la portée des $\mathsf{E}\ (\mathsf{G}\,\varphi \wedge \bigwedge_i \psi_i)$. □

Nous allons maintenant présenter des résultats concernant l'insertion de l'opérateur Now.

6.5 L'insertion de l'opérateur N

Nous allons étendre les résultats obtenus au chapitre 5 aux logiques sans $Next$. Tout d'abord, pour toutes les logiques séparables considérées ici, on peut ajouter l'opérateur Now sans perdre la propriété de séparation:

Lemme 6.5.1

- *$NCTL^*\backslash X$ est séparable.*
- *$B(U, F^{-1}, N)$ est séparable.*
- *$B(U, F^{-1}, N, \wedge, \neg)$ est séparable.*
- *$B(U, \overset{\infty}{F}, F^{-1}, N, \wedge, \neg)$ est séparable.*

Preuve La preuve est basée sur le lemme 5.2.6. □

On en déduit immédiatement:

Théorème 6.5.2

- $NCTL^*\backslash X \equiv PCTL^*\backslash X \equiv_i CTL^*\backslash X$
- $B(U, F^{-1}, N) \equiv B(U, F^{-1}) \equiv_i B(U)$
- $B(U, F^{-1}, N, \wedge, \neg) \equiv B(U, F^{-1}, \wedge, \neg) \equiv_i B(U)$
- $B(U, \overset{\infty}{F}, F^{-1}, N, \wedge, \neg) \equiv B(U, \overset{\infty}{F}, F^{-1}, N, \wedge, \neg) \equiv_i B(U, \overset{\infty}{F}, \wedge, \neg)$

Dans certains cas, l'opérateur Now peut augmenter l'expressivité des logiques. Il en est ainsi pour $B(\mathsf{F}, \mathsf{F}^{-1})$:

Théorème 6.5.3 $B(\mathsf{F}, \mathsf{F}^{-1}) \prec_i B(\mathsf{F}, \mathsf{F}^{-1}, \mathsf{N}) \equiv_i B(\mathsf{U})$

Preuve Ce lemme est basé sur l'équivalence suivante:

$$\text{Pour toutes formules } f \text{ et } g \text{ du futur,} \quad \mathsf{E}\, f\, \mathsf{U}\, g \;\equiv\; \mathsf{N}\,\mathsf{EF}\Big(g \wedge \mathsf{G}^{-1}(f \vee g)\Big)$$

Cette équivalence permet de transformer toute formule de $B(\mathsf{U})$ en une formule équivalente de $B(\mathsf{F}, \mathsf{F}^{-1}, \mathsf{N})$: une stratégie top-down garantit que toute les sous-formules f et g appartiennent à $B(\mathsf{U})$, mais on peut même appliquer cette transformation sans stratégie particulière: dans ce cas, soit les formules f et g appartiennent à $B(\mathsf{U})$, soit elles contiennent des sous-formules de la forme $\mathsf{N}\,\mathsf{EF}(\ldots)$ mais celles-ci ne remettent pas en cause le fait que f et g soient du futur. ☐

Lorsque nous avions considéré l'ajout de l'opérateur Now aux logiques CTL_p, CTL_p^+ et $ECTL_p^+$, nous avions choisi de définir de nouvelles logiques de manière à mieux tirer profit de cet opérateur puisqu'il autorisait notamment des opérateurs $Since$ en partie gauche de $Until$ ou dans la portée de $Since$ dans certaines conditions. Nous allons définir les logiques $\boldsymbol{P}CTL_{\backslash X}$, $\boldsymbol{P}CTL^+_{\backslash X}$ et $\boldsymbol{PE}CTL^+_{\backslash X}$:

Définition 6.5.4 *La logique* $\boldsymbol{P}CTL_{\backslash X}$

$$\boldsymbol{P}CTL_{\backslash X} \ni f, g \quad ::= \; a \mid f \wedge g \mid \neg f \mid E\,\varphi\, U f \mid A\,\varphi\, U\,\psi \mid \varphi\, S\,\psi \mid N f$$

$$\text{avec } \varphi, \psi \quad ::= \; a \mid \varphi \wedge \psi \mid \neg\varphi \mid E\,\varphi\, U\,\psi \mid A\,\varphi\, U\,\psi \mid F^{-1}\varphi \mid N f$$

On a donc: $B(\mathsf{U}, \mathsf{F}^{-1}, \mathsf{N}) \subseteq CTL_p \backslash \mathsf{X} + \mathsf{N} \subseteq \boldsymbol{P}CTL_{\backslash X}$.

Exemple 6.5.5 *La formule* $E\Big(N\,EF\,(a \wedge b\,S\,c)\Big)\,U\,d$ *est une formule de* $\boldsymbol{P}CTL_{\backslash X}$ *mais n'appartient pas à* $CTL_p \backslash X + N$ *puisque l'opérateur* S *apparait en partie gauche de Until.*

Définition 6.5.6 *La logique* $\boldsymbol{P}CTL^+_{\backslash X}$

$$\boldsymbol{P}CTL^+_{\backslash X} \ni f, g \quad ::= \; a \mid f \wedge g \mid \neg f \mid E\, f_c \mid \varphi\, S\,\psi \mid N f$$

$$f_c, g_c \quad ::= \; f_c \wedge g_c \mid f_c \vee g_c \mid \neg\varphi_c \mid \varphi\, U f \mid \varphi\, S\,\psi \mid N f_c$$

$$\varphi, \psi \quad ::= \; a \mid \varphi \wedge \psi \mid \neg\varphi \mid E\,\varphi_c \mid F^{-1}\varphi \mid N f$$

$$\varphi_c, \psi_c \quad ::= \; \varphi_c \wedge \psi_c \mid \neg\varphi_c \mid \varphi\, U\,\psi \mid F^{-1}\varphi \mid N f_c$$

On a donc: $B(\mathsf{U}, \mathsf{F}^{-1}, \mathsf{N}, \wedge, \neg) \subset CTL_p^+ \backslash \mathsf{X} + \mathsf{N} \subset \boldsymbol{P}CTL_{\backslash X}^+$.

Exemple 6.5.7 *La formule* $E\Big[\Big([N\,EF(a \wedge b\,S\,c)]\,U\,d\Big) \wedge \Big(a\,U\,(b \wedge c\,S\,d)\Big)\Big]$ *est une formule de* $\boldsymbol{P}CTL_{\backslash X}^+$ *mais n'appartient pas à* $CTL_p^+ \backslash X + N$ *puisque l'opérateur* S *apparait en partie gauche de* $Until$.

Et enfin, on a:

Définition 6.5.8 *La logique* $\boldsymbol{P}ECTL_{\backslash X}^+$

$$\begin{aligned}
\boldsymbol{P}ECTL_{\backslash X}^+ \ni f, g \quad &::= a \mid f \wedge g \mid \neg f \mid E\, f_c \mid \varphi\, S\, \psi \mid N\, f \\
f_c, g_c \quad &::= f_c \wedge g_c \mid f_c \vee g_c \mid \neg\varphi_c \mid \varphi\, U\, f \mid \overset{\infty}{F}\varphi \mid \varphi\, S\, \psi \mid N\, f_c \\
\varphi, \psi \quad &::= a \mid \varphi \wedge \psi \mid \neg\varphi \mid E\,\varphi_c \mid F^{-1}\varphi \mid N\, f \\
\varphi_c, \psi_c \quad &::= \varphi_c \wedge \psi_c \mid \neg\varphi_c \mid \varphi\, U\, \psi \mid \overset{\infty}{F}\varphi \mid F^{-1}\varphi \mid N\, f_c
\end{aligned}$$

On a donc: $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \mathsf{N}, \wedge, \neg) \subset ECTL_p^+ \backslash \mathsf{X} + \mathsf{N} \subset \boldsymbol{P}ECTL_{\backslash X}^+$.

Exemple 6.5.9 *La formule* $E\Big[\Big([N\,EF(a \wedge b\,S\,c)]\,U\,d\Big) \wedge \overset{\infty}{F}\Big(b \wedge N\,EF(a \wedge b\,S\,c)\Big)\Big]$ *est une formule de* $\boldsymbol{P}ECTL_{\backslash X}^+$ *mais n'appartient pas à* $ECTL_p^+ \backslash X + N$ *puisque un opérateur* S *apparait en partie gauche de* $Until$ *et un autre en dans la portée d'un* $\overset{\infty}{F}$.

Ces trois logiques sont séparables:

Lemme 6.5.10

- $\boldsymbol{P}CTL_{\backslash X}$ *est séparable.*
- $\boldsymbol{P}CTL_{\backslash X}^+$ *est séparable.*
- $\boldsymbol{P}ECTL_{\backslash X}^+$ *est séparable.*

Preuve La preuve est similaire à celle des lemmes 5.4.3, 5.4.7 et 5.4.11. □

On en déduit donc:

Théorème 6.5.11

- $\boldsymbol{P}CTL_{\backslash X} \equiv_i B(U)$
- $\boldsymbol{P}CTL_{\backslash X}^+ \equiv_i B(U)$
- $\boldsymbol{P}ECTL_{\backslash X}^+ \equiv_i B(U, \overset{\infty}{F}, \wedge, \neg)$

Preuve conséquence du lemme 6.5.10. □

$$
\begin{array}{c}
NCTL^*\backslash \mathsf{X} \equiv_i PCTL^*\backslash \mathsf{X} \equiv CTL^*\backslash \mathsf{X} \\
\downarrow \\
\boldsymbol{P}ECTL^+_{\backslash X} \equiv_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg \equiv_i B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg) \\
\downarrow \\
B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}) \\
\downarrow \\
B(\mathsf{U}, \overset{\infty}{\mathsf{F}}) \\
\downarrow \\
\boldsymbol{P}CTL^+_{\backslash X} \equiv_i \boldsymbol{P}CTL_{\backslash X} \equiv_i B(\mathsf{U}, \mathsf{F}^{-1}) \equiv_i B(\mathsf{U}) \equiv_i B(\mathsf{F}, \mathsf{F}^{-1}, \mathsf{N})
\end{array}
$$

Figure 6.1: Expressivité des logiques sans $Next$.

6.6 Conclusion

En conclusion de ce chapitre sur les logiques sans $Next$, on peut souligner à la fois plusieurs points communs ainsi que des différences avec les logiques étudiées aux chapitres 4 et 5.

A la différence des fragments CTL, $ECTL^+$, ...il n'est pas possible d'ajouter les opérateurs X^{-1} et F^{-1}_s aux logiques $B(\mathsf{F})$, $B(\mathsf{U})$, $B(\mathsf{U}, \overset{\infty}{\mathsf{F}})$ ou $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ si l'on veut les traduire dans $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$. Cette impossibilité est dûe à la nature de ses logiques où l'on a choisi de prélever l'opérateur EX pour ne pas distinguer des modèles qui ne diffèrent que par la duplication d'états contigus vérifiant les mêmes propositions atomiques. Les deux opérateurs X^{-1} et F^{-1}_s ne respectent pas cette restriction puisqu'il permettent d'exprimer initialement la propriété $\mathsf{EX}\, a$. Ajouter ces opérateurs reviendraient à remettre en cause le choix initial qui a motivé l'usage de ces logiques.

Comme pour les logiques CTL, CTL^+ et $ECTL^+$, nous avons vu qu'il est possible d'ajouter l'opérateur F^{-1} aux logiques $B(\mathsf{U})$, $B(\mathsf{U}, \wedge, \neg)$ et $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ et de traduire les logiques ainsi définies dans leur fragment pur-futur: on peut donc utiliser $B(\mathsf{U}, \mathsf{F}^{-1})$ ou $B(\mathsf{U}, \mathsf{F}^{-1}, \wedge, \neg)$ (resp. $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \mathsf{F}^{-1}, \wedge, \neg)$) pour la spécification et se ramener (après traduction) à des formules de $B(\mathsf{U})$ (resp. $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$) pour la vérification.

De plus, il est possible d'insérer partiellement (avec les mêmes contraintes qu'au chapitre 4) l'opérateur S aux logiques $B(\mathsf{U})$, $B(\mathsf{U}, \wedge, \neg)$ et $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ et de traduire les logiques ainsi étendues dans $B(\mathsf{U})$ ou $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$.

Enfin, l'opérateur Now nous a conduit à définir les fragments $\boldsymbol{P}CTL_{\setminus X}$, $\boldsymbol{P}CTL^+_{\setminus X}$ et $\boldsymbol{P}ECTL^+_{\setminus X}$ sur le même modèle que nous avions défini $\boldsymbol{P}CTL$, $\boldsymbol{P}CTL^+$ et $\boldsymbol{P}ECTL^+$. Le fragment $\boldsymbol{P}CTL^+_{\setminus X}$ (resp. $\boldsymbol{P}ECTL^+_{\setminus X}$) est le plus gros fragment de $PCTL^*\backslash\mathsf{X}$ traduisible dans $B(\mathsf{U})$ (resp. $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$) présenté dans cette étude. Ces résultats sont décrits par la figure 6.1. Ces fragments permettent, dans la pratique, une expression beaucoup plus naturelle que $B(\mathsf{U})$ ou $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$. Par exemple, la spécification avec passé de l'ascenseur présentée au chapitre 3 est composée de formules de $\boldsymbol{P}CTL_{\setminus X}$.

Chapitre 7

Conclusion

7.1 Bilan

Dans cette étude, nous avons présenté plusieurs traductions de logiques avec passé dans des logiques pur-futur. Les principaux résultats concernent les logiques CTL et $ECTL^+$ pour lesquelles il existe des algorithmes de model-checking efficaces.

Tout d'abord nous avons montré comment traduire les logiques $CTL + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$ ou $CTL^+ + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$ dans CTL ainsi que la logique $ECTL^+ + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$ dans $ECTL^+$. De plus, nous avons montré que l'opérateur Now qui permet, associé aux opérateurs F^{-1} et F_s^{-1} d'exprimer facilement de nombreuses propriétés, pouvait être ajouté à $CTL + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$, $CTL^+ + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$ et $ECTL^+ + \mathsf{F}^{-1},\mathsf{F}_s^{-1}$ sans empêcher la traduction des logiques ainsi définies dans CTL ou $ECTL^+$.

Enfin, nous avons montré que seule une insertion partielle des opérateurs X^{-1} et S aux logiques CTL ou $ECTL^+$ permettait des traductions dans leur fragment pur-futur. Cela nous a conduit à la définition des logiques $\boldsymbol{P}CTL$, $\boldsymbol{P}CTL^+$ et $\boldsymbol{P}ECTL^+$ qui autorisent l'emploi des opérateurs S et X^{-1} avec certaines restrictions mais laissent toute liberté pour l'usage des opérateurs F^{-1}, F_s^{-1} et N. Parmi les logiques considérées dans cette thèse, $\boldsymbol{P}CTL^+$ (resp. $\boldsymbol{P}ECTL^+$) est le plus grand fragment avec passé de $PCTL^*$ pouvant se traduire dans CTL (resp. $ECTL^+$).

Au chapitre 6, nous avons montré que les résultats obtenus pour CTL, CTL^+ et $ECTL^+$ pouvaient être adaptés aux logiques $B(\mathsf{U})$, $B(\mathsf{U}, \wedge, \neg)$ et $B(\mathsf{U}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$ dans lesquelles il est possible d'ajouter les opérateurs F^{-1} et N ainsi que l'opérateur S sous certaines conditions.

Les résultats obtenus permettent donc de construire un module de traduction en amont d'un outil de model-checking classique admettant CTL ou $ECTL^+$ comme langage d'entrée, et d'écrire des spécifications plus naturelles avec la logique $\boldsymbol{P}CTL^+$ ou $\boldsymbol{P}ECTL^+$. L'exemple de l'ascenseur présenté au chapitre 3 montre comment la spécification avec $\boldsymbol{P}CTL$ est plus naturelle (plus proche de la spécification informelle, plus simple et donc plus facile à concevoir et à modifier) que la spécification du même système avec CTL ou CTL^+.

7.2 Perspectives

Nous pouvons dégager plusieurs perspectives possibles pour continuer ce travail. Tout d'abord, il serait intéressant de définir des axiomatisations de nos différentes logiques avec passé qui prendraient en compte l'opérateur Now . D'autre part, une étude de la complexité théorique et pratique des traductions, dans le prolongement des remarques de la section 4.4.6, serait nécessaire.

Enfin, il faut souligner que les méthodes de traduction présentées dans ce travail peuvent être utilisées dans d'autres domaines.

7.3 Applicabilité des méthodes de séparation

7.3.1 Logiques modales de processus

Les logiques modales de processus sont des formalismes assez proches des logiques temporelles, elles contiennent des modalités permettant de faire référence aux actions effectuées par des processus, définis par exemple par une algèbre de processus, et dont la sémantique est un système de transitions étiquetées. Elles constituent un outil important pour l'analyse des systèmes réactifs. Ces logiques, comme par exemple HML [HM80], ont été notamment utilisées en tant que langage caractérisant des équivalences sémantiques. Elles donnent alors un autre point de vue sur ces équivalences et constituent un langage pour expliquer ces équivalences sémantiques.

En collaboration avec S. Pinchinat et Ph. Schnoebelen, nous nous sommes intéressé à trois logiques modales (dont deux munies d'opérateurs du passé) qui ont été définies dans la littérature [DV90, Gla90] et qui caractérisent la *branching bisimulation*. Nous avons établi trois théorèmes de traduction pour ces logiques basés sur des lemmes de séparation de même nature que ceux présentés ici pour les fragments de $PCTL^*$[LPS95].

7.3.2 Logiques modales pour les modèles de vrai parallélisme

Le choix du modèle sémantique sur lequel on interprète la spécification, dépend des caractéristiques que l'on souhaite modéliser et du type de spécification que l'on veut établir. Ainsi lorsqu'on utilise des graphes (structure de Kripke), on représente le comportement de deux processus concurrents par l'entrelacement de leur actions atomiques (on parle de modèle d'entrelacement): la représentation du programme $a \parallel b$ est alors identique à celle du programme $ab + ba$ (choix non-déterministe de l'exécution de a suivi de b ou de b suivi de a). Par ce choix de représentation, on perd donc une certaine information sur les processus.

Les limites des modèles interleaving ont été soulignées depuis longtemps et d'autres modèles, dits de "vrai parallélisme" ont été étudiés. L'idée sous-jacente est d'intégrer dans les modèles la notion de concurrence ou de causalité. Les logiques temporelles ont, elles aussi, été adaptées à ces modèles beaucoup plus riches. Par exemple, plusieurs logiques temporelles ont été proposées pour les réseaux de Petri [Rei89, Rei92].

En collaboration avec F. Cherief, S. Pinchinat et Ph. Schnoebelen, nous avons présenté deux logiques modales pour les structures d'événements [Win89] (munies d'opérateurs du passé et incorporant des observations de pomsets) caractérisant la bisimulation *history-preserving* [GV89] et montré comment elles pouvaient se traduire l'une dans l'autre [CLP92, PLS94], là encore, à la base on retrouve un résultat de séparation avec un ensemble de règles de réécriture et une stratégie d'application.

Ces résultats soulignent l'intérêt de ces méthodes de séparation. Il seraient intéressant d'étudier des problèmes similaires dans le cadre d'autres logiques temporelles, comme les logiques d'intervalles, les logiques munies de l'opérateur *Chop* ou les logiques pour le temps-réel. Chacun de ces cas pose des problèmes particuliers en plus de ceux qui apparaîssent pour les logiques temporelles classiques.

Annexe A

Annexe

Au chapitre 4, nous avons utilisé une approche basée sur la théorie des jeux pour montrer que la formule de PTL $\mathsf{F}\,(a \ \wedge \ b\,\mathsf{U}\,c)$ était inexprimable avec la logique $PLTL$ (i.e. $L(\mathsf{F}\,, \mathsf{X}\,, \mathsf{F}^{-1}, \mathsf{X}^{-1}\,)$). Dans cette annexe, se trouvent rassemblés plusieurs lemmes utiles pour la preuve du théorème 4.2.8.

Le premier lemme dont nous avons besoin, concerne un jeu où les deux modèles considérés sont identiques (M'_n) et seule la position initiale des jetons diffère. Ce lemme énonce que le jeu où un jeton bleu et un jeton rouge sont placés respectivement en k et l sur la séquence S_0^- de M'_n et d'autre part en k et l sur la séquence S_0^+, est gagnant pour le joueur II si le nombre de coups est inférieur à n:

Lemme A.0.1

$$\forall k, l \ | \ 0 \leq k\ , l \ \leq 2n+2, \ \ \forall r \leq n, \quad On\ a : G_r(M'_n, \alpha_k^{-0}, \alpha_l^{-0}; M'_n, \alpha_k^0, \alpha_l^0) \in II$$

Preuve Pour gagner, le joueur I doit utiliser le fait qu'il y a n séquences S dans le premier modèle à gauche des jetons, tandis qu'il y en a $n+1$ dans le second modèle (ou de manière symétrique: $n+1$ séquences S à droite des jetons pour le premier modèle et n pour le second). Il est clair que mettre en évidence ce décalage nécessiterait pour le joueur I au moins $n+1$ coups. Donc $G_r(M'_n, \alpha_k^{-0}, \alpha_l^{-0}, M'_n, \alpha_k^0, \alpha_l^0) \in II$. La figure A illustre cette situation.

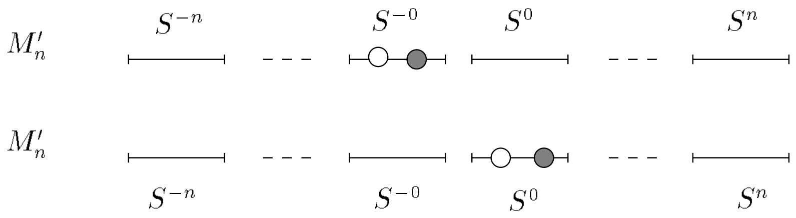


Figure A.1: Le jeu $G_r(M'_n, \alpha_k^{-0}, \alpha_l^{-0}; M'_n, \alpha_k^{-0}, \alpha_l^{-0})$

Pour montrer ce lemme, on peut facilement prouver simultanément par induction sur r les propositions suivantes où $\varepsilon \in \{+,-\}$:

$r \leq n$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-0}, \alpha_l^{-0}; M'_n, \alpha_k^0, \alpha_l^0) \in II$	1
$r \leq n-i, \ \ i \leq n$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-i}, \alpha_l^{-i}; M'_n, \alpha_k^{-i+1}, \alpha_l^{-i+1}) \in II$	2
$r \leq n-i-1, \ \ i \leq n-1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{i}, \alpha_l^{i}; M'_n, \alpha_k^{i+1}, \alpha_l^{i+1}) \in II$	3
$r \leq n-1,$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-1}, \alpha_l^{-0}; M'_n, \alpha_k^{-0}, \alpha_l^0) \in II$	4
$r \leq n,$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-0}, \alpha_l^{0}; M'_n, \alpha_k^0, \alpha_l^1) \in II$	5
$r \leq n-i, \ \ i \leq n$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-i}, \alpha_l^{-i+1}; M'_n, \alpha_k^{-i+1}, \alpha_l^{-i+2}) \in II$	6
$r \leq n-i-2, \ \ i \leq n-2$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{i}, \alpha_l^{i+1}; M'_n, \alpha_k^{i+1}, \alpha_l^{i+2}) \in II$	7
$r \leq n \ \ 0 < j \leq n$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-0}, \alpha_l^{\varepsilon j}; M'_n, \alpha_k^0, \alpha_l^{\varepsilon j}) \in II$	8
$r \leq n \ \ \mid k-l \mid > 1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-0}, \alpha_l^{\varepsilon 0}; M'_n, \alpha_k^0, \alpha_l^{\varepsilon 0}) \in II$	8b
$\left.\begin{array}{l} r \leq n-i, \ i \leq n, \ 0 \leq j \leq n \\ (\varepsilon j \neq -i \text{ et } \varepsilon j \neq = -i+1 \end{array}\right\}$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-i}, \alpha_l^{\varepsilon j}; M'_n, \alpha_k^{-i+1}, \alpha_l^{\varepsilon j}) \in II$	9
$r \leq n-i, \ i \leq n, \ \mid k-l \mid > 1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-i}, \alpha_l^{-i}; M'_n, \alpha_k^{-i+1}, \alpha_l^{-i}) \in II$	9b
$r \leq n-i, \ i \leq n, \ \mid k-l \mid > 1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{-i}, \alpha_l^{-i+1}; M'_n, \alpha_k^{-i+1}, \alpha_l^{-i+1}) \in II$	9c
$\left.\begin{array}{l} r \leq n-i-1, \ 0 \leq j \leq n \\ i \leq n-1 \ , \varepsilon j \neq i \text{ et } \varepsilon j \neq i+1 \end{array}\right\}$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{i}, \alpha_l^{\varepsilon j}; M'_n, \alpha_k^{i+1}, \alpha_l^{\varepsilon j}) \in II$	10
$r \leq n-i-1, \ i \leq n-1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{i}, \alpha_l^{i}; M'_n, \alpha_k^{i+1}, \alpha_l^{i}) \in II$	10b
$r \leq n-i-1, \ i \leq n-1$	$\Rightarrow$	$G_r(M'_n, \alpha_k^{i}, \alpha_l^{i+1}; M'_n, \alpha_k^{i+1}, \alpha_l^{i+1}) \in II$	10c

Ces propositions correspondent aux différents cas possibles pour “imiter” les coups du joueur I: à partir de la position initiale, si le joueur I déplace un jeton du premier modèle en dehors des séquences S_1^-, S_0^- et S_0^+, alors le joueur II exécute le même coup sur le second modèle, si le joueur I déplace un jeton du premier modèle sur la séquence S_1^-, alors le joueur II déplace le jeton correspondant sur la séquence S_0^-, etc ... $\square$

En conséquence, nous avons le lemme suivant qui correspond à la proposition $8b$ établit dans la preuve précédente:

Lemme A.0.2 $\forall\ k < k' - 1\ ,\ \ \forall\ r \leq n$

$$G_r(M'_n, \alpha_k^0, \alpha_{k'}^0; M'_n, \alpha_k^{-0}, \alpha_{k'}^0) \in II$$

Nous allons maintenant définir une famille de fonction $f_{i,k}^{\varepsilon}$ qui vont nous permettre de représenter les positions des jetons correspondant aux jeux gagnant pour le joueur II.

Définition A.0.3 *La famille de fonctions $f_{i,k}^{\varepsilon}$ de M_n dans M'_n avec $\varepsilon \in \{+,-\}$, $0 \leq i \leq n$, $k \in \{0,1,2\}$ et $l \leq 2n+2$:*

$$f_{i,k}^{+} = \begin{cases} \alpha_l^{-j} \to \alpha_l^{-j} & \forall j \leq n \\ \alpha_l^{+j} \to \alpha_l^{+(j+1)} & \forall j \leq i-1 \\ \alpha_l^{+j} \to \alpha_l^{+j} & \forall j \geq i \end{cases} \qquad f_{i,0}^{+} = \begin{cases} \beta_l \to \alpha_l^{+0} & \forall l \leq \frac{n}{2} \\ \beta_l \to \alpha_{l+1+3\frac{n}{2}}^{+0} & \forall l \geq 1+\frac{n}{2} \end{cases}$$

$$f_{i,1}^{+} = \{\ \beta_l \to \alpha_l^{+0}$$

$$f_{i,2}^{+} = \{\ \beta_l \to \alpha_{l+n+1}^{+0}$$

De manière symétrique, on a:

$$f_{i,k}^{-} = \begin{cases} \alpha_l^{-j} \to \alpha_l^{-j} & \forall j \geq i \\ \alpha_l^{-j} \to \alpha_l^{-(j+1)} & \forall j \leq i-1 \\ \alpha_l^{+j} \to \alpha_l^{+j} & \forall j \leq n \end{cases} \qquad f_{i,0}^{-} = \begin{cases} \beta_l \to \alpha_l^{-0} & \forall l \leq \frac{n}{2} \\ \beta_l \to \alpha_{l+1+3\frac{n}{2}}^{-0} & \forall l \geq 1+\frac{n}{2} \end{cases}$$

$$f_{i,1}^{-} = \{\ \beta_l \to \alpha_l^{-0}$$

$$f_{i,2}^{-} \{\ \beta_l \to \alpha_{l+n+1}^{-0}$$

La différence entre $f_{i,0}^{+}$, $f_{i,1}^{+}$ et $f_{i,2}^{+}$ (resp. $f_{i,0}^{-}$, $f_{i,1}^{-}$ et $f_{i,2}^{-}$) réside uniquement dans la définition de l'image des états de T dans S_0^+ (resp. S_0^-) (voir figure A.2).

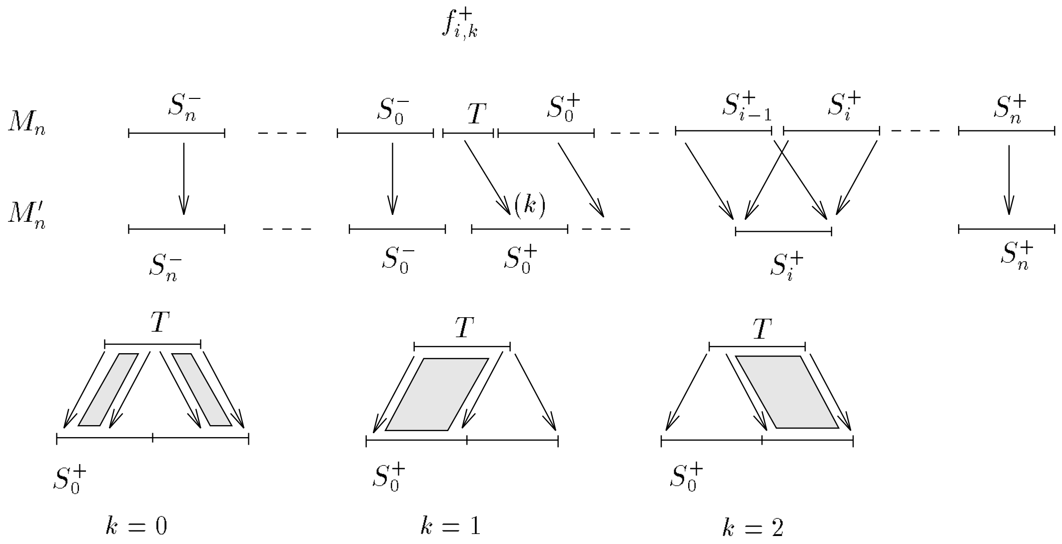


Figure A.2: Fonctions $f^+_{i,k}$

Le lien entre l'existence de positions gagnantes pour le joueur II et les fonctions $f^\varepsilon_{i,k}$ est établi par le lemme suivant dont découle directement le théorème 4.2.8:

Lemme A.0.4 *Pour* $r \leq \frac{n}{2}$*, et* $\varepsilon = \pm$ *, on a :*

$$(1) \quad G_r(M_n, x, y; M'_n, f^\varepsilon_{i,0}(x), f^\varepsilon_{i,0}(y)) \in II \quad si \begin{cases} r + i \leq n \\ x, y \sim f^\varepsilon_{i,0}(x), f^\varepsilon_{i,0}(y) \end{cases}$$

$$(2) \quad G_r(M_n, x, y; M'_n, f^\varepsilon_{i,2}(x), f^\varepsilon_{i,2}(y)) \in II \quad si \begin{cases} r + i \leq n \\ x, y \sim f^\varepsilon_{i,2}(x), f^\varepsilon_{i,2}(y) \\ r \leq Min(x, y) - 1 \end{cases}$$

$$(3) \quad G_r(M_n, x, y; M'_n, f^\varepsilon_{i,1}(x), f^\varepsilon_{i,1}(y)) \in II \quad si \begin{cases} r + i \leq n \\ x, y \sim f^\varepsilon_{i,1}(x), f^\varepsilon_{i,1}(y) \\ r \leq n - Max(x, y) \end{cases}$$

$$(4) \quad G_r(M_n, x, y; M'_n, f^\varepsilon_{i,2}(x), f^\varepsilon_{i,0}(y)) \in II \quad si \begin{cases} r + i \leq n \\ x, y \sim f^\varepsilon_{i,2}(x), f^\varepsilon_{i,0}(y) \\ r \leq x - 1 \end{cases}$$

$$(5) \quad G_r(M_n, x, y; M'_n, f^\varepsilon_{i,1}(x), f^\varepsilon_{i,0}(y)) \in II \quad si \begin{cases} r + i \leq n \\ x, y \sim f^\varepsilon_{i,1}(x), f^\varepsilon_{i,0}(y) \\ r \leq n - x - 1 \end{cases}$$

Preuve Preuve par induction sur r.

$\star$ $r = 0$: on a toujours comme hypothèse que la position des jetons est compatible, la situation est donc toujours gagnante pour le joueur II.

$\star$ $r = s+1$:Nous allons montrer successivement (1), (2), (3), (4) et (5). Supposons que $\varepsilon = +$ (l'autre cas est similaire).

1. Montrons (1): $G_r(M_n, x, y; M'_n, f^+_{i,0}(x), f^+_{i,0}(y)) \in II$ si $\begin{cases} r+i \leq n \\ x, y \sim f^+_{i,0}(x), f^+_{i,0}(y) \end{cases}$.

   Pour établir ce résultat, nous allons montrer comment le joueur II peut obtenir un jeu gagnant pour s tours quel que soit le premier coup de I.

   Supposons que I joue : $x \rightsquigarrow_b x'$. On distingue plusieurs cas possibles:

   - $x', y \sim f^+_{i,0}(x'), f^+_{i,0}(y)$: Par hypothèse d'induction, on a donc $G_s(x', y, f^+_{i,0}(x'), f^+_{i,0}(y)) \in II$.
   - $x', y \not\sim f^+_{i,0}(x'), f^+_{i,0}(y)$: Cette situation correspond à cinq possibilités :

$$\begin{array}{llcl} (a) & y < x' & \& & f^+_{i,0}(x') \leq f^+_{i,0}(y) \\ (b) & x' < y & \& & f^+_{i,0}(y) \leq f^+_{i,0}(x') \\ (c) & \mid x' - y \mid > 1 & \& & \mid f^+_{i,0}(x') - f^+_{i,0}(y) \mid = 1 \\ (d) & \mid x' - y \mid = 1 & \& & \mid f^+_{i,0}(x') - f^+_{i,0}(y) \mid > 1 \end{array}$$

   Pour chacun de ces cas, nous allons indiquer le coup que le joueur II peut faire de manière à obtenir une situation dans laquelle il dispose d'un jeu gagnant pour s tours:

   (a) Le premier cas correspond à une de ces deux situations:

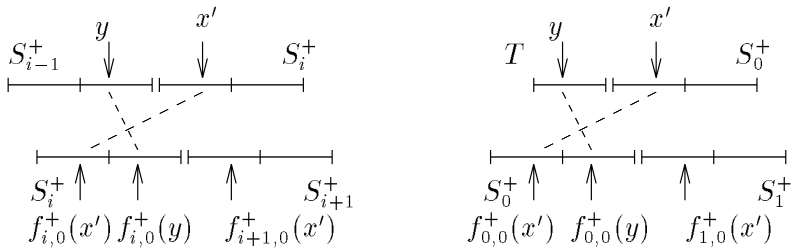


Figure A.3: $y < x'$ et $f^+_{i,0}(x') < f^+_{i,0}(y)$

   Dans ce cas, $f^+_{i,0}(y) = f^+_{i+1,0}(y)$, et donc II peut jouer: $f^+_{i,0}(x) \rightsquigarrow_b f^+_{i+1,0}(x')$. On a bien: $x', y \sim f^+_{i+1,0}(x'), f^+_{i+1,0}(y)$ et $s+1+i \leq n$, on peut utiliser l'hypothèse d'induction et déduire que :

$$G_s(M_n, x', y; M'_n, f^+_{i+1,0}(x'), f^+_{i,0}(y)) \in II$$

   (b) Le second cas correspond à une de ces deux situations:

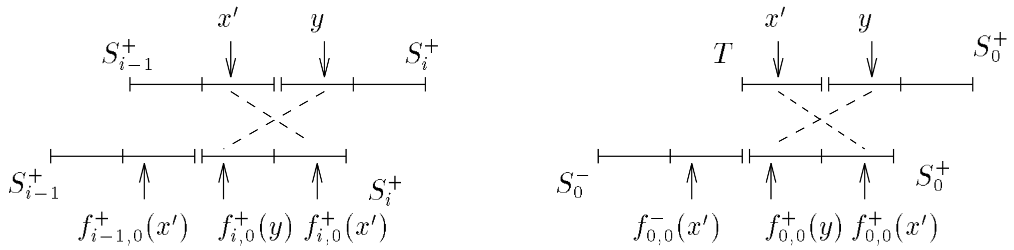


Figure A.4: $x' < y$ et $f_{i,0}^+(y) < f_{i,0}^+(x')$

Si $i > 0$, II peut jouer $f_{i,0}^+(x) \rightsquigarrow_b f_{i-1,0}^+(x')$. Comme $f_{i,0}^+(y) = f_{i-1,0}^+(y)$, et $x', y \sim f_{i-1,0}^+(x'), f_{i-1,0}^+(y)$, on peut utiliser l'hypothèse d'induction et déduire que :

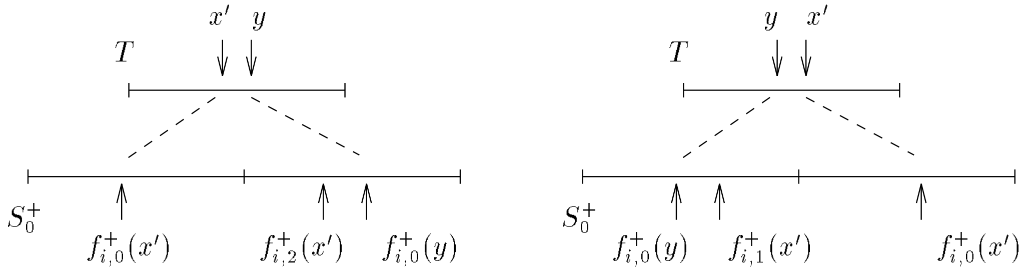
$$G_s(M_n, x', y; M_n', f_{i-1,0}^+(x'), f_{i,0}^+(y)) \in II$$

Si $i = 0$ (figure de droite ci-dessus), on a $f_{0,0}^+(y) = f_{0,0}^-(y)$ et $x', y \sim f_{0,0}^-(x'), f_{0,0}^-(y)$ dont on déduit que :

$$G_r(M_n, x', y; M_n', f_{0,0}^-(x'), f_{0,0}^+(y)) \in II$$

(c) Le cas où $\mid x' - y \mid > 1$ et $\mid f_{i,0}^+(x') - f_{i,0}^+(y) \mid = 1$ correspond à des situations similaires aux deux premiers cas: on utilise $f_{i+1,0}^+(x')$, $f_{i-1,0}^+(x')$ ou $f_{0,0}^-(x')$ selon que respectivement $y < x'$, $x' > y$ ou $x' > y$ avec $i = 0$.

(d) $\mid x' - y \mid = 1$ et $\mid f_{i,0}^+(x') - f_{i,0}^+(y) \mid > 1$.
Le cas où $x' = \alpha_{2n+2}^{i-1}$ et $y = \alpha_0^i$, et celui où $x' = \beta_{n+1}$ et $y = \alpha_0^0$ et $i = 0$, ainsi que celui où $y = \alpha_{2n+2}^{i-1}$ et $x' = \alpha_0^i$ et enfin celui où $y = \beta_{n+1}$ et $x' = \alpha_0^0$ et $i = 0$ peuvent être traités en utilisant les mêmes méthodes que celles utilisées pour les cas 1 et 2.
Il reste à traiter les deux cas suivants: $x' = \beta_{\frac{n}{2}}$ et $y = \beta_{\frac{n}{2}+1}$ ou $y = \beta_{\frac{n}{2}}$ et $x' = \beta_{\frac{n}{2}+1}$, c'est à dire :



(a) (b)

Dans le cas (a), on II peut jouer :$f_{i,0}^+(x) \rightsquigarrow_b f_{i,2}^+(x')$. Comme $s + i \leq n$, $\mid f_{i,2}^+(x') - f_{i,0}^+(y) \mid = 1$, et $s \leq \frac{n}{2} - 1$ (car $s + 1 \leq \frac{n}{2}$), on obtient par hypothèse d'induction que :

$$G_s(M_n, x', y; M'_n, f^+_{i,2}(x'), f^+_{i,0}(y)) \in II$$

Dans le cas (b), on II peut jouer :$f^+_{i,0}(x) \rightsquigarrow_b f^+_{i,1}(x')$. Comme $s+i \leq n$, $\mid f^+_{i,1}(x') - f^+_{i,0}(y) \mid = 1$, et $s \leq n - \frac{n}{2} - 1$ (car $s+1 \leq \frac{n}{2}$), on obtient par hypothèse d'induction que :

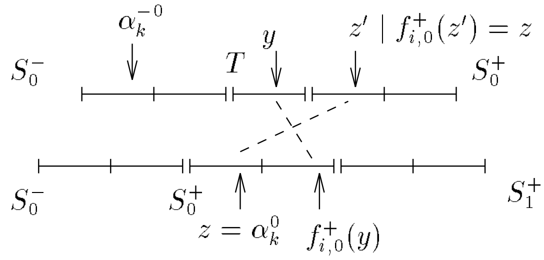
$$G_s(M_n, x', y; M'_n, f^+_{i,1}(x'), f^+_{i,0}(y)) \in II$$

Pour chaque situation où $x', y \not\sim f^+_{i,0}(x'), f^+_{i,0}(y)$, nous avons montré qu'il existait un coup $f^+_{i,0}(x) \rightsquigarrow_b z$ pour II de manière à ce que :

$$G_s(M_n, x', y; M'_n, z, f^+_{i,0}(y)) \in II$$

Pour tout coup de I : $x \rightsquigarrow_b x'$, nous savons donc qu'il existe une manière pour II d'avoir un jeu gagnant pour s coups. On a de même si I joue $y \rightsquigarrow_r y'$.

Si I déplace un des jetons de M'_n comme $f^+_{i,0}(x) \rightsquigarrow_b z$, le joueur II peut imiter ce déplacement sur M_n en jouant $x \rightsquigarrow_b z'$ tq $f^+_{i,0}(z') = z$. (il peut être nécessaire de faire appel à des arguments similaires à ceux présentés ci-dessus, mais les cas considérés sont plus simples). La seule difficulté se produit lorsque $y = \beta_{k'}$ avec $k' > \frac{n}{2} + 1$ et $z = \alpha^0_k$ avec $\frac{n}{2} + 1 \leq k \leq 3\frac{n}{2}$:



Dans ce cas, II peut jouer $x \rightsquigarrow_b \alpha_k^{-0}$:

$$\left.\begin{array}{l} \text{Par h.i., on a} \\ G_s(M_n, \alpha_k^{-0}, \beta_{k'}; M'_n, \alpha_k^{-0}, f^+_{i,0}(\beta_{k'})) \in II \\ \text{D'après le lemme A.0.2, on a:} \\ G_s(M'_n, \alpha_k^{-0}, f^+_{i,0}(\beta_{k'}); M'_n, \alpha_k^0, f^+_{i,0}(\beta_{k'})) \in II \end{array}\right\} \Rightarrow G_s(M_n, \alpha_k^{-0}, \beta_{k'}; M'_n, \alpha_k^0, f^+_{i,0}(\beta_{k'})) \in II$$

Nous avons donc :

$$G_r(M_n, x, y; M'_n, f^+_{i,0}(x), f^+_{i,0}(y)) \in II$$

2. Montrons (2): $G_r(M_n, x, y; M'_n, f^+_{i,2}(x), f^+_{i,2}(y)) \in II$ si $\begin{cases} r+i \leq n \\ x, y \sim f^+_{i,2}(x), f^+_{i,2}(y) \\ r \leq Min(x,y)-1 \end{cases}$

   Nous pouvons supposer que $x, y \in T$ (sinon on se ramène aux cas (1) ou (4) puisque $\forall z \notin T, f^+_{i,2}(z) = f^+_{i,0}(z)$).

   Nous distinguons plusieurs cas:

$$\begin{array}{ll} (a) & I : x \rightsquigarrow_b x' \text{ et } x' = y-1 \\ (b) & I : x \rightsquigarrow_b x' \text{ et } x' = y+1 \\ (c) & I : x \rightsquigarrow_b x' \text{ et } \mid x'-y \mid > 1 \text{ et } x' \in T \\ (d) & I : x \rightsquigarrow_b x' \text{ et } \mid x'-y \mid > 1 \text{ et } x' \notin T \\ (e) & I : f^+_{i,2}(x) \rightsquigarrow_b z \text{ et } z \in S^0 \\ (f) & I : f^+_{i,2}(x) \rightsquigarrow_b z \text{ et } z \notin S^0 \end{array}$$

   (a) $I : x \rightsquigarrow_b y-1$. II peut alors jouer $f^+_{i,2}(x) \rightsquigarrow_b f^+_{i,2}(y)-1$. Comme $y-1 \geq r \geq 1$, on a $y-1, y \sim f^+_{i,2}(y)-1, f^+_{i,2}(y)$ et de plus: $f^+_{i,2}(y)-1 = f^+_{i,2}(y-1)$. D'autre part, $s \leq Min(y, y-1)-1$, on peut appliquer l'hypothèse d'induction et obtenir:

$$G_s(M_n, y-1, y; M'_n, f^+_{i,2}(y-1), f^+_{i,2}(y)) \in II$$

   (b) $I : x \rightsquigarrow_b y+1$. II peut jouer $f^+_{i,2}(x) \rightsquigarrow_b f^+_{i,2}(y)+1$. On a $y+1, y \sim f^+_{i,2}(y)+1, f^+_{i,2}(y)$ et de plus: $f^+_{i,2}(y)+1 = f^+_{i,2}(y+1)$. D'autre part, $s+1 \leq y-1$, on peut donc appliquer l'hypothèse d'induction et obtenir:

$$G_s(M_n, y+1, y; M'_n, f^+_{i,2}(y+1), f^+_{i,2}(y)) \in II$$

   (c) $I : x \rightsquigarrow_b x'$ et $\mid x'-y \mid > 1$ et $x' \in T$. On distingue deux cas:

   - $x', y \sim f^+_{i,0}(x'), f^+_{i,2}(y)$: Dans ce cas, on peut utiliser l'hypothèse d'induction (car $s \leq x-1$ et $s+i \leq n$) pour obtenir:

$$G_s(M_n, x', y; M'_n, f^+_{i,0}(x'), f^+_{i,2}(y)) \in II$$

   - $x', y \not\sim f^+_{i,0}(x'), f^+_{i,2}(y)$: On alors une situation où $y < x'$ et $f^+_{i,2}(y) > f^+_{i,0}(x')$, c'est à dire:

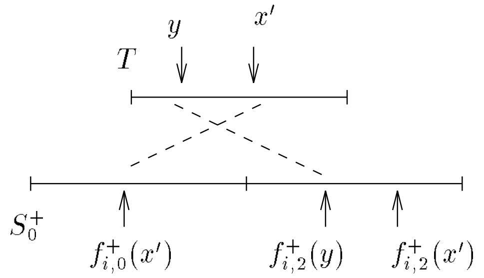


Figure A.5: $x', y \not\sim f^+_{i,0}(x'), f^+_{i,2}(y)$

II peut alors jouer $f^+_{i,2}(x) \rightsquigarrow_b f^+_{i,2}(x')$. La position ainsi atteinte est compatible et comme $s+1 \leq x-1$ et $s+i \leq n$, on peut utiliser l'hypothèse d'induction et obtenir:

$$G_s(M_n, x', y; M'_n, f^+_{i,2}(x'), f^+_{i,2}(y)) \in II$$

(d) $I : x \rightsquigarrow_b x'$ et $\mid x' - y \mid > 1$ et $x' \notin T$. On distingue deux cas:

- $x', y \sim f^+_{i,0}(x'), f^+_{i,2}(y)$: Dans ce cas, on peut utiliser l'hypothèse d'induction (car $s \leq x-1$ et $s+i \leq n$) pour obtenir:

$$G_s(M_n, x', y; M'_n, f^+_{i,0}(x'), f^+_{i,2}(y)) \in II$$

- $x', y \not\sim f^+_{i,0}(x'), f^+_{i,2}(y)$: On alors une situation proche de $1-a$ où $i=0$ et $x' \in S^0$ et $f^+_{i,2}(y) > f^+_{i,0}(x')$ (ou $\mid f^+_{i,2}(y) - f^+_{i,0}(x') \mid \leq 1$).
II peut alors jouer $f^+_{0,2}(x) \rightsquigarrow_b f^+_{1,0}(x')$; sachant que $f^+_{0,2}(y) = f^+_{1,2}(y)$, on applique l'h.i., et on obtient:

$$G_s(M_n, x', y; M'_n, f^+_{1,0}(x'), f^+_{1,2}(y)) \in II$$

(e) $I : f^+_{i,2}(x) \rightsquigarrow_b z$ et $z \notin S^0$. Dans ce cas, II peut jouer $x \rightsquigarrow_b z'$ tel que $f^+_{i,0}(z') = z'$ (il peut exister plusieurs z' vérifiant cette propriété: II a alors plusieurs coups possibles).

(f) $I : f^+_{i,2}(x) \rightsquigarrow_b z$ et $z \in S^0$. Soit $z' \in S^0$. Nous distinguons deux cas:

- $z', y \sim z, f^+_{i,2}(y)$: II peut alors jouer $x \rightsquigarrow_b z'$.
- $z', y \not\sim z, f^+_{i,2}(y)$: Si $\mid z - f^+_{i,2}(y) \mid \leq 1$, voir (a) et (b).
On se trouve alors dans une situation similaire à:

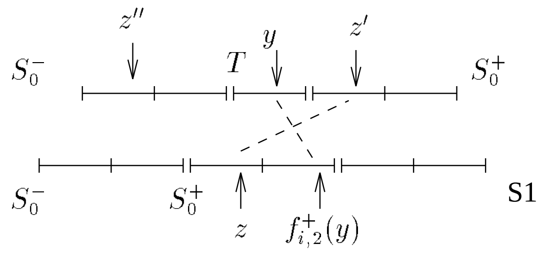


Figure A.6: $z', y \not\sim z, f^+_{i,2}(y)$

On a $z = \alpha_k^0$, on choisit $z'' = \alpha_k^{-0}$, et on obtient $G_s(M_n, z'', y; M'_n, z, f^+_{i,2}(y)) \in II$ de la manière qu'au point (1) (utilisation du lemme A.0.1).

3. Montrons (3): $G_r(M_n, x, y; M'_n, f^\varepsilon_{i,1}(x), f^\varepsilon_{i,1}(y)) \in II$ si $\begin{cases} r+i \leq n \\ x, y \sim f^\varepsilon_{i,1}(x), f^\varepsilon_{i,1}(y). \\ r \leq n - Max(x,y) \end{cases}$

idem que le cas (2).

4. Montrons (4): $G_r(M_n, x, y; M'_n, f^{\varepsilon}_{i,2}(x), f^{\varepsilon}_{i,0}(y)) \in II$ si $\begin{cases} r+i \leq n \\ x, y \sim f^{\varepsilon}_{i,2}(x), f^{\varepsilon}_{i,0}(y) \\ r \leq x-1 \end{cases}$

   On suppose que $x \in T$ (en cas contraire on se ramène à (1) car $f^{+}_{i,2}(x) = f^{+}_{i,0}(x)$ si $x \in S^{\varepsilon j}$). Nous distinguons plusieurs cas, et pour chacun, nous pouvons nous ramener à des cas étudiés précédemment:

$$\begin{array}{lll} (a) & I: x \rightsquigarrow_b x' & \text{cf. } (1) \\ (b) & I: y \rightsquigarrow_b y' & \text{cf. } (2) \\ (c) & I: f^{+}_{i,2}(x) \rightsquigarrow_b z & \text{cf. } (1) \\ (d) & I: f^{+}_{i,0}(y) \rightsquigarrow_b z & \text{cf. } (2) \end{array}$$

5. Montrons (5): $G_r(M_n, x, y; M'_n, f^{\varepsilon}_{i,1}(x), f^{\varepsilon}_{i,0}(y)) \in II$ si $\begin{cases} r+i \leq n \\ x, y \sim f^{\varepsilon}_{i,1}(x), f^{\varepsilon}_{i,0}(y) \\ r \leq n-x-1 \end{cases}$

On procède de même pour $\varepsilon = -$. $\square$

Le lemme A.0.4 permet de déduire le théorème 4.2.8 qui énonce que le jeu $G_r(M_n, \alpha_0^{-n}, \alpha_0^{-n}; M'_n, \alpha_0^{-n}, \alpha_0^{-n})$ avec $r \leq \frac{n}{2}$ est gagnant pour le joueur II.

Remarque Dans cette étude, nous avons utilisé une approche basée sur la théorie des jeux pour montrer que la formule $\mathsf{F}\,(a \wedge b\,\mathsf{U}\,c)$ est inexprimable avec $PLTL$. Nous nous sommes donc limités à la définition d'un jeu adéquat pour la logique $L(\mathsf{F}_{\gg}, \mathsf{X}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}^{-1})$ [1], mais il suffit de changer la relation $\sim$ pour obtenir d'autres jeux qui sont adéquats pour d'autres logiques. $\square$

[1] $L(\mathsf{F}_{\gg}, \mathsf{X}, \mathsf{F}^{-1}_{\ll}, \mathsf{X}^{-1}) \equiv PLTL$

Bibliographie

[AEF89] E. Audureau, P. Enjalbert, and L. Fariñas del Cerro. *Logique temporelle. Sémantique et validation de programmes parallèles.* Masson, Paris, 1989.

[Bar85] H. Barringer. Up and down the temporal way. Tech. Report UMCS-85-9-3, Dept. of Computer Science, Univ. of Manchester, September 1985.

[BCG87] M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing Kripke structures in temporal logic. In *Proc. CAAP'87, Pisa, LNCS 249*, pages 256–270. Springer-Verlag, March 1987.

[BCG88] M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, PA*, pages 428–439, June 1990.

[BCMD90] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill. Sequential circuit verification using symbolic model checking. In *Proc. 27th ACM/IEEE Design Automation Conf., Orlando, Florida*, June 1990.

[BdG92] A. Burrieza and I. P. de Guzmán. A new algebraic semantic approach and some adequate connectives for computation with temporal logic over discrete systems. *J. Applied Non-Classical Logics*, 2(2):181–201, 1992.

[BPM83] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[CD89] E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 428–437. Springer-Verlag, 1989.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Logics of Programs Workshop, Yorktown Heights, LNCS 131*, pages 52–71. Springer-Verlag, May 1981.

[CES83] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. In *Proc. 10th ACM Symp. Principles of Programming Languages, Austin, Texas*, pages 117–126, January 1983.

[CG87] E. M. Clarke and O. Grümberg. Research on automatic verification of finite-state concurrent systems. *Ann. Rev. Comput. Sci.*, 2:269–290, 1987.

[CGL92] E. M. Clarke, O. Grümberg, and D. E. Long. Model checking and abstraction. In *Proc. 19th ACM Symp. Principles of Programming Languages, Albuquerque, New Mexico*, pages 343–354, January 1992.

[CLP92] F. Cherief, F. Laroussinie, and S. Pinchinat. Modal logics with past for true concurrency. Internal Report, May 1992.

[DV90] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation (extended abstract). In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, PA*, pages 118–129, June 1990.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th ICALP, Noordwijkerhout, LNCS 85*, pages 169–181. Springer-Verlag, July 1980.

[EC82] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[EH85] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[EH86] E. A. Emerson and J. Y. Halpern. "Sometimes" and "Not Never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[EL87] E. A. Emerson and C. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8:275–306, 1987.

[Eme83] E. A. Emerson. Alternative semantics for temporal logics. *Theoretical Computer Science*, 26:121–130, 1983.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. Elsevier Science Publishers, 1990.

[ES89] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 123–172. Springer-Verlag, 1989.

[Gab87] D. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Proc. Temporal Logic in Specification, Altrincham, UK, LNCS 398*, pages 409–448. Springer-Verlag, April 1987.

[Gab92] D. M. Gabbay. Temporal logic: Mathematical foundations. Tech. Report MPI-I-92-213, Max-Planck-Institut für Informatik, Saarbrücken, March 1992.

[GHR92] D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds. Temporal logic: Mathematical foundations - part 2. Tech. Report MPI-I-92-242, Max-Planck-Institut für Informatik, Saarbrücken, September 1992.

[Gla90] R. J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Free Univ. of Amsterdam, May 1990.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. Principles of Programming Languages, Las Vegas, Nevada*, pages 163–173, January 1980.

[Gra83] S. Graf. On Lamport's comparison between linear and branching time logic. Research Report 348, IMAG, Grenoble, January 1983.

[GV89] R. J. van Glabbeek and F. Vaandrager. Modular specifications in process algebra. In *Proc. 9th IFIP WG 6.1 Int. Symp. Protocol Specification, Testing and Verification, Enschede, NL*, June 1989.

[Hal87] R. Hale. Using temporal logic for prototyping: the design of a lift controller. In *Proc. Temporal Logic in Specification, Altrincham, UK, LNCS 398*, pages 375–408. Springer-Verlag, April 1987.

[HKP82] D. Harel, D. Kozen, and R. Parikh. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25:144–170, 1982.

[HM80] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *Proc. 7th ICALP, Noordwijkerhout, LNCS 85*, pages 299–309. Springer-Verlag, July 1980.

[HT87] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proc. 14th ICALP, Karlsruhe, LNCS 267*, pages 269–279. Springer-Verlag, July 1987.

[IK87] N. Immerman and D. Kozen. Definability with bounded number of bound variables. In *Proc. 2nd IEEE Symp. Logic in Computer Science, Ithaca, NY*, pages 236–244, 1987.

[Kam68] J. A. W. Kamp. *Tense Logic and the theory of linear order.* PhD thesis, UCLA, Los Angeles, 1968.

[Kri63] S. Kripke. Semantical analysis of modal logic I. Normal modal propositional calculi. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 9:67–96, 1963.

[KT90] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 14, pages 789–840. Elsevier Science Publishers, 1990.

[KVd83] R. Koymans, J. Vytopil, and W. P. de Roever. Real-time programming and asynchronous message passing. In *Proc. 2nd ACM Symp. Principles of Distributed Computing, Montreal, Canada*, pages 187–197, August 1983.

[Lam80] L. Lamport. "Sometimes" is sometimes "Not Never". In *Proc. 7th ACM Symp. Principles of Programming Languages, Las Vegas, Nevada*, pages 174–185, January 1980.

[Lam83] L. Lamport. What good is temporal logic ? In *Proc. Information Processing (IFIP) 83*, pages 657–668. North-Holland, September 1983.

[Lar91] F. Laroussinie. *L'expressivité du passé dans la logique temporelle.* Rapport de DEA, Univ. Grenoble, September 1991.

[Lar94] F. Laroussinie. About the expressive power of CTL combinators. To appear in Inf. Proc. Letters, November 1994.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symp. Principles of Programming Languages, New Orleans*, January 1985.

[LPS95] F. Laroussinie, S. Pinchinat, and Ph. Schnoebelen. Translations between modal logics of reactive systems. *Theoretical Computer Science*, 140(1):53–71, 1995.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. Logics of Programs Workshop, Brooklyn, LNCS 193*, pages 196–218. Springer-Verlag, June 1985.

[LS92] B. Littlewood and L. Strigini. The risks of software. *Sci. Am.*, 276(5):38–43, November 1992.

[LS94] F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. In *Proc. STACS'94, Caen, France, LNCS 775*, pages 47–58. Springer-Verlag, February 1994.

[MMKR93] L. E. Moser, P. M. Melliar-Smith, G. Kutty, and Y. S. Ramakrishna. Completeness and soundness of axiomatizations for temporal logics without next. *Fundamenta Informaticae*, 1993. To appear.

[MP79] Z. Manna and A. Pnueli. The modal logic of programs. In *Proc. 6th ICALP, Graz, LNCS 71*, pages 385–409. Springer-Verlag, July 1979.

[MP81] Z. Manna and A. Pnueli. Verification of concurrent programs: the temporal framework. In R. S. Boyer and J. S. Moore, editors, *The Correctness Problem in Computer Science*, chapter 5, pages 215–273. Academic Press, 1981.

[MP83] Z. Manna and A. Pnueli. How to cook a temporal proof system for your pet language. In *Proc. 10th ACM Symp. Principles of Programming Languages, Austin, Texas*, January 1983.

[MP89] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 201–284. Springer-Verlag, 1989.

[MP90] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. 9th ACM Symp. Principles of Distributed Computing, Quebec City, Canada*, pages 377–408, August 1990.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*, volume I: Specification. Springer-Verlag, 1992.

[MT27] J. M. E. Mc Taggart. *The Nature of Existence*, volume I. Cambridge, 1927.

[MW84] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6(1), January 1984.

[OL82] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455–495, July 1982.

[Pin92] S. Pinchinat. Ordinal processes in comparative concurrency semantics. Tech. Report 81, LIFIA-IMAG, Grenoble, April 1992.

[PLS94] S. Pinchinat, F. Laroussinie, and Ph. Schnoebelen. Logical characterizations of truly concurrent bisimulation. Tech. Report 114, LIFIA-IMAG, Grenoble, March 1994.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Foundations of Computer Science, Providence*, pages 46–57, November 1977.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[Pnu86a] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In *Current Trends in Concurrency, LNCS 224*, pages 510–584. Springer-Verlag, 1986.

[Pnu86b] A. Pnueli. Specification and development of reactive systems. In *Proc. IFIP 86*, pages 845–858. North-Holland, 1986.

[Pri67] A. Prior. *Past, Present, and Future.* Clarendon Press, Oxford, 1967.

[QS82] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Int. Symp. on Programming, Turin, LNCS 137*, pages 337–351. Springer-Verlag, April 1982.

[Rei89] W. Reisig. Towards a temporal logic for causality and choice in distributed systems. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 603–627. Springer-Verlag, 1989.

[Rei92] W. Reisig. Elements of a temporal logic coping with concurrency. SFB-Bericht 342/23/92 A, Institut für Informatik, TUM, Munich, November 1992.

[RP86] R. Rosner and A. Pnueli. A choppy logic. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, pages 306–313, June 1986.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, July 1985.

[Sch90a] B.-H. Schlingloff. *Zur temporalen Logik von Baümen.* PhD thesis, Munich Technical Univ., March 1990.

[Sch90b] Ph. Schnoebelen. *Sémantique du parallélisme et logique temporelle. Application au langage FP2.* Thèse de Doctorat, I.N.P. de Grenoble, France, June 1990.

[Sma64] J. Smart, editor. *Problems of Space and Time.* Mc Millan, 1964.

[Sti92] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Oxford Univ. Press, 1992.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 4, pages 133–191. Elsevier Science Publishers, 1990.

[vB89] J. van Benthem. Time, logic and computation. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 1–49. Springer-Verlag, 1989.

[Win89] G. Winskel. An introduction to event structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 364–397. Springer-Verlag, 1989.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.

[ZC93] A. Zanardo and J. Carmo. Ockhamist computational logic: Past-sensitive necessitation in CTL*. *J. Logic Computat.*, 3(3):249–268, 1993.

[Zuc86] L. Zuck. *Past Temporal Logic.* PhD thesis, Weizmann Institute, Rehovot, Israel, August 1986.