

UNIVERSITÉ PARIS DIDEROT - PARIS 7
Laboratoire d'Informatique Algorithmique : Fondements et Applications

THÈSE
pour l'obtention du diplôme de
Docteur de l'Université Paris Diderot,
spécialité Informatique

à l'École Doctorale de Sciences Mathématiques de Paris Centre

JEUX ET AUTOMATES

SUR LES ORDRES

présentée et soutenue publiquement par

Julien CRISTAU

le 13 décembre 2010

Directeur de thèse : Olivier CARTON

Après avis de : Alexander RABINOVICH
Géraud SÉNIZERGUES

Jury composé de : Olivier CARTON Directeur
François LAROUSSINIE
Dominique PERRIN
Philippe SCHNOEBELEN
Géraud SÉNIZERGUES Rapporteur
Olivier SERRE

Chapitre 1

Introduction

Cette thèse se place dans le cadre de la théorie des automates et langages formels. Un des objectifs de ce domaine est d'étudier les propriétés d'ensembles de mots (suites de symboles provenant d'un alphabet en général fini). Ces ensembles de mots, ou langages, peuvent être décrits par des formules logiques, ou encore par des automates finis, c'est-à-dire des systèmes à nombre fini d'états qui lisent le mot lettre par lettre. On s'intéresse alors à comparer l'expressivité de différents formalismes, les classes de langages correspondantes, et les problèmes de complexité liées à la manipulation de ces langages (test du vide, intersection, etc.).

L'une des motivations de cette théorie, qui reste très importante de nos jours, est la spécification et la vérification de systèmes, pour laquelle elle fournit des outils fondamentaux. Dans ce contexte, une lettre est souvent une abstraction d'un état du système considéré et un mot représente une suite d'états ou d'actions. On peut alors exprimer sous la forme d'un langage l'ensemble des exécutions « correctes », *i.e.* la spécification d'une propriété voulue du système. Cette spécification est souvent exprimée sous la forme d'une formule logique. De façon similaire, l'ensemble des comportements possibles du système constitue un langage ; on peut dans certains cas représenter le système lui-même sous la forme d'une machine à états qui génère ce langage.

On a donc besoin de bien comprendre les différents formalismes impliqués (classes de langages, logiques, automates) et leurs relations pour pouvoir les manipuler efficacement. Des recherches ont également porté sur différentes extensions, fournissant des modèles d'automates et des logiques plus expressifs. Cela permet de décrire des exécutions de systèmes plus complexes, de

supplémenter les modèles existants par des abstractions plus fines. Cela permet également d'étudier de nouvelles familles de spécifications.

Ces applications considèrent le plus souvent des mots infinis ou dans certains cas des mots temporisés. On a aussi vu se développer de nouveaux formalismes comme les logiques temporelles qui permettent d'exprimer plus facilement des propriétés sur le comportement dans le temps du système.

Parallèlement, le développement de la théorie des jeux a fourni des outils pour analyser des systèmes où interagissent plusieurs entités : acteurs économiques, biologiques, etc. Elle est utilisée en informatique, et en particulier en vérification, où l'utilisation de jeux est fertile pour les problèmes de synthèse de contrôleurs pour les systèmes ouverts. Dans ce cadre, on considère le système et son environnement comme deux joueurs opposés et on cherche à réaliser une certaine spécification. Cette spécification est alors représentée par la condition de victoire du jeu.

Les deux théories se rejoignent dans le domaine des jeux sur les graphes, où les objets étudiés sont très proches de ceux de la théorie des automates : graphes finis, ou graphes d'exécution de divers types d'automates (à pile, par exemple). Une partie dans le jeu correspond à un chemin dans le graphe, et donc à un mot. De la même façon l'exécution d'un automate est un parcours du graphe sous-jacent, qui constitue un mot (suite des états ou des étiquettes). Pour modéliser des systèmes dont le temps d'exécution est indéterminé on s'intéresse pour les automates comme pour les jeux à des mots (ou parties) finis ou infinis (*i.e.* de longueur ω). Cela permet par exemple de modéliser l'interaction finie ou infinie entre deux agents, ou le comportement à la limite d'un système à horloge discrète. On peut cependant raffiner le modèle de temps utilisé, par exemple pour modéliser de façon plus fine des systèmes comportant différentes horloges avec des échelles de temps très différentes, ou bien pour s'affranchir du temps discret.

Le point clé des travaux présentés dans cette thèse est l'extension du modèle de temps pour certaines questions liées aux automates et aux jeux : d'une part les liens entre automates et logique temporelle, d'autre part des jeux de longueur ordinaire. Par extension du modèle de temps, on entend l'utilisation d'ordres plus grands comme support pour les mots ou les parties : ordinaux dans le cas des jeux, ordres linéaires quelconques dans le cas des automates.

Automates La théorie des automates est née dès les années 1950 avec entre autres Kleene [Kle56], Rabin et Scott [RS59]. Il s'agissait au départ de mots finis, puis par la suite de mots infinis (indexés par ω) avec Büchi [Bü62], Muller [Mul63] ou McNaughton [McN66]. De nombreuses recherches ont permis de mettre en évidence des présentations des langages réguliers et ω -réguliers, de MSO aux semi-groupes en passant par les expressions rationnelles.

Les automates permettent de modéliser des systèmes simples (le modèle est plus primitif que celui des machines de Turing, par exemple). L'un des attraits est que cette simplicité permet de rendre la plupart des problèmes relatifs aux automates décidables, tout en restant un modèle relativement expressif.

Jeux La théorie des jeux est utilisée dans des domaines très variés, de la biologie [Smi82] à la philosophie [Kav86], en passant par l'économie [Cou38]. En informatique, on la retrouve dans l'intelligence artificielle [GMW87], la logique [Bla92], les langages de programmation [Chr03], et bien d'autres. Nous considérons ici un modèle de jeux à deux joueurs sur des graphes finis. Ces jeux sont très liés aux automates finis. Ils ont permis la première solution au problème de synthèse de Church par Büchi et Landweber [BL69]. Les jeux sont aussi des ingrédients majeurs de toutes les preuves modernes du théorème de Rabin [Rab69] depuis les travaux de Gurevich et Harrington [GH82].

Contributions Dans le Chapitre 2 on donne des définitions générales et propriétés importantes des automates, logique et ordres.

On s'intéresse ensuite dans le Chapitre 3 au lien entre automates sur les ordres et logique temporelle. On montrera en particulier l'importance de l'utilisation des transducteurs pour transformer une formule LTL en un automate. Tester le vide du langage reconnu par cet automate fournit ainsi une réponse à la satisfaisabilité de la formule.

Dans le Chapitre 4 on s'intéresse à des jeux à deux joueurs sur des graphes finis, dans lesquels les parties ont une longueur ordinaire. On montre que ces jeux sont déterminés et on donne un algorithme permettant de déterminer le vainqueur, et un algorithme qui calcule une stratégie gagnante.

Chapitre 2

Généralités

Les automates permettent d'étudier des langages de mots. Ils fournissent un modèle simple alternatif à la logique. Ils sont par exemple utilisés en vérification pour modéliser des systèmes et des spécifications. Pour étudier des systèmes réactifs on utilise des jeux à deux joueurs. Les automates comme les jeux sont en général considérés avec un temps discret (où la longueur des mots ou des parties est ω), on s'intéressera dans cette thèse à des modèles de temps étendus : ordinaux et ordres linéaires.

Dans ce chapitre, on donne quelques définitions, notations et résultats de base, qui seront utiles dans la suite de la thèse.

2.1 Automates et logique

Les automates sont depuis longtemps un modèle central en informatique théorique. Ils fournissent un modèle simple et naturel de machines à états, et ont donné lieu à énormément d'algorithmes, extensions, etc.

L'association des automates, de formalismes logiques et de jeux a été (et est toujours) extrêmement utile pour l'étude de systèmes réactifs, aussi bien logiciels que matériels, et la complémentarité de ces approches est établie.

Ces systèmes ont la particularité d'interagir perpétuellement avec leur environnement, d'où l'utilisation d'une part d'objets infinis, d'autre part de jeux à deux joueurs. Ces objets proposent des problèmes théoriques majeurs, mais sont aussi très directement liés aux algorithmes de model-checking permettant le développement et la vérification automatique de systèmes logiciels et matériels complexes.

2.1.1 ω -automates

Les automates finis sont utilisés depuis longtemps en théorie des langages, ils fournissent un formalisme simple et naturel pour décrire des ensembles de mots, pendant à la logique, mais plus faciles à manipuler dans certains cas. Ils ont donc été largement étudiés, aussi bien sur les mots finis que sur les mots infinis. De plus, les langages de mots infinis apparaissent par exemple comme traces d'exécution de systèmes réactifs, et les automates sont un moyen simple de spécifier des propriétés sur ces traces.

On donne ici quelques définitions et propriétés de base, le lecteur pouvant consulter [Tho97] pour une étude détaillée.

Définition 1 (ω -automate). *Un automate est donné par un tuple $\mathcal{A} = (Q, \Sigma, \delta, I, \Omega)$ où Q est un ensemble fini d'états de contrôle, Σ est un alphabet fini, I est l'ensemble des états initiaux, $\delta \subseteq Q \times \Sigma \times Q$ est la relation de transition, et Ω est la condition d'acceptation.*

Une *exécution* de l' ω -automate \mathcal{A} sur le mot d'entrée $x = (x_i)_{i < \omega}$ est une séquence $\rho = (\rho_i)_{i < \omega}$ telle que pour tout i , $(\rho_i, x_i, \rho_{i+1}) \in \delta$. Dans le cas des langages de mots finis, une exécution est acceptante si elle commence dans un état initial et se termine dans un état final. Dans le cas des mots infinis, il n'y a pas un unique état final, donc l'acceptation dépend des états visités infiniment souvent.

Étant donnée une exécution ρ , on note $\lim \rho$ l'ensemble des états visités infiniment souvent : $\lim \rho = \{q \in Q \mid \forall i \exists j > i \rho_j = q\}$. La condition d'acceptation Ω a plusieurs représentations classiques, parmi lesquelles :

- automates de Büchi [Bü62] : on donne $F \subseteq Q$, et ρ est acceptée si $F \cap \lim \rho \neq \emptyset$;
- automates de Muller [Mul63] : on donne $\mathcal{F} \subseteq \mathcal{P}(Q)$, et ρ est acceptée si $\lim \rho \in \mathcal{F}$;
- automates de parité : on donne une fonction de coloriage $\chi : Q \rightarrow [0, \dots, k]$ et ρ est accepté si la plus petite couleur visitée infiniment souvent est paire, *i.e.* $\min \chi(\lim \rho)$ est pair.

Tous ces modèles (ainsi que les variantes déterministes pour Muller et parité) permettent de reconnaître les mêmes langages ; les détails des équivalences sont regroupés dans [Far01]. En revanche, la variante déterministe des automates de Büchi est strictement moins puissante, et ne peut pas reconnaître des langages tels que l'ensemble des mots ayant un nombre fini de b (sur l'alphabet $\{a, b\}$).

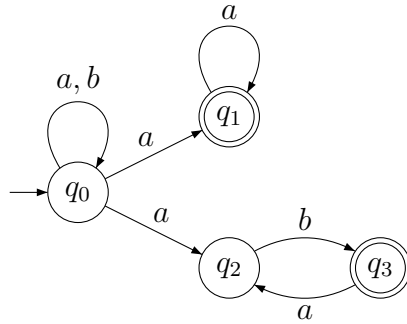


FIGURE 2.1 – Un automate de Büchi acceptant le langage défini par l’expression rationnelle $(a + b)a^\omega + (a + b)(ab)^\omega$

Un automate de Büchi peut être représenté graphiquement comme Figure 2.1, les flèches représentant les transitions possibles, et les états finaux étant doublement cerclés.

Les automates finis fournissent un modèle à la fois puissant et algorithmiquement plaisant : les questions de vide, d’universalité, d’inclusion, etc, sont décidables avec une complexité raisonnable. Par exemple, tester le vide revient à trouver un état final accessible qui appartienne à un cycle.

Théorème 2. *Le vide d’un automate de Büchi est décidable en temps linéaire.*

Une autre façon simple de décrire un langage est d’utiliser une expression rationnelle. Grâce au théorème de Kleene, on sait que cette description fournit la même expressivité que les automates :

Théorème 3 (Kleene). *Les langages reconnaissables par (ω -)automate fini sont exactement les langages définissables par une expression (ω -)rationnelle.*

Une expression rationnelle est construite avec les opérateurs de concaténation (\cdot), union ($+$), étoile ($*$) et itération infinie ($^\omega$) à partir des lettres de l’alphabet. Ce lien fort permet une interaction entre théorie des automates et étude algébrique des monoïdes et semi-groupes.

Un langage peut aussi être défini comme l’ensemble des modèles d’une formule logique, que ce soit en logique du premier ordre (FO), en logique monadique du second ordre (MSO) ou avec une logique temporelle comme LTL.

Une des motivations de ces études est la vérification et le model-checking de systèmes logiciels. On peut représenter certains systèmes comme des machines à nombre fini d'états, dont on modélise l'évolution par un automate. On peut spécifier les comportements voulus ou interdits pour le système par des formules logiques, et une fois encore les liens forts entre automates et logique sont très utiles.

De ce lien avec la logique découlent des propriétés importantes comme par exemple la clôture par complémentation, qui dans le cas des automates non déterministes est loin d'être évidente. La classe des langages ω -rationnels est également close par union et intersection.

Inversement, les automates fournissent des réponses simples à certains problèmes qui du point de vue purement logique sont difficiles ou algorithmiquement complexes, comme le test du vide.

2.1.2 Logique

On suppose fixé un alphabet Σ .

Définition 4. On note $\text{FO}(<)$, ou simplement FO , la logique propositionnelle du premier ordre utilisant le prédicat binaire d'ordre.

Les formules de FO sont définies inductivement, et contiennent :

- les formules atomiques $a(x)$ et $x < y$ où a est une lettre de l'alphabet Σ , x et y sont des variables du premier ordre
- $\neg\phi$ et $\phi \wedge \psi$ pour toutes formules ϕ et ψ de FO
- $\exists x\phi$ pour toute variable x et toute formule ϕ

On dit qu'une formule est *close* si elle ne contient pas de variable libre.

La logique monadique du second ordre, notée $\text{MSO}(<)$ ou MSO , contient :

- les formules atomiques $a(x)$, $x < y$, $x \in X$ où a est une lettre de l'alphabet Σ , x et y sont des variables du premier ordre, X est une variable du second ordre
- $\neg\phi$ et $\phi \wedge \psi$ pour toutes formules ϕ et ψ de MSO
- $\exists x\phi$ et $\exists X\phi$ pour toutes variables x et X du premier et du second ordre, respectivement, et pour toute formule ψ de MSO .

Théorème 5 (Büchi). Les langages reconnaissables par (ω -)automate fini sont exactement les langages définissables par formule de MSO .

Ce théorème est fondamental puisqu'il pose l'équivalence entre automates et logique, qui complète les liens avec semi-groupes et expressions rationnelles.

Dans le cadre de la vérification de systèmes, et du model-checking en particulier, on préfère souvent utiliser des logiques modales. On mentionnera en particulier ici la logique temporelle linéaire, ou LTL, définie comme suit.

Définition 6. *La logique LTL est définie inductivement par :*

- a pour $a \in \Sigma$
- $X\phi$ pour toute formule ϕ de LTL
- $\phi\mathcal{U}\psi$ et $\phi\mathcal{S}\psi$ pour toutes formules ϕ et ψ de LTL

LTL possède certains atouts : elle permet d'exprimer facilement des propriétés de sûreté ou d'accessibilité, qui sont classiques en model checking, et la complexité du problème de satisfaisabilité est PSPACE pour LTL, et non élémentaire pour FO.

Théorème 7 (Kamp). *Sur les mots infinis LTL et FO ont la même expressivité.*

Les langages définis par des formules du premier ordre (et donc LTL) sont dits apériodiques. Ce nom vient de leur caractérisation algébrique comme langages reconnus par un semi-groupe du même nom. On peut également les définir comme les langages définis par les automates non-comptants.

Ce lien entre automates et LTL permet de réduire des questions de model-checking ou de satisfaisabilité à des questions de test du vide du langage reconnu par un automate, et a donc été très utilisé en vérification.

2.2 Jeux

Les automates ne permettent cependant pas de modéliser tous les systèmes de façon satisfaisante. Les jeux infinis à deux joueurs apparaissent naturellement comme modélisation de l'interaction entre un système (pour lequel on souhaite par exemple synthétiser un contrôleur) et son environnement (a priori hostile). On peut souhaiter en revanche représenter le contrôleur à synthétiser sous la forme d'un automate fini.

Les jeux apportent également un élément d'alternation qui est absent des automates, et permet de résoudre certains problèmes de complémentation. Par exemple, les jeux infinis sont utilisés pour prouver la correction

d'une méthode de complémentation des automates de Büchi dans [Kla01], et la détermination des jeux de parité est nécessaire à la complémentation d'automates d'arbres dans [Nie01].

On représente le système comme un graphe fini avec des états contrôlés par le système (Ève) ou l'environnement (Adam), et un jeton se déplace sur le graphe en suivant les choix des joueurs. Une condition de victoire détermine (en général après une partie infinie) lequel des deux joueurs est gagnant.

Formellement une *arène* est un graphe (V, E) où l'ensemble V de sommets est partitionné en V_E (sommets contrôlés par Ève) et V_A (sommets contrôlés par Adam). Lors d'une partie, le jeton est placé sur un sommet initial q_0 , et à chaque tour le joueur auquel appartient le sommet courant décide de déplacer le jeton le long d'une arête. Les deux joueurs construisent ainsi une *partie* $\rho \in V^\omega$. Une *condition de victoire* $\Omega \subseteq V^\omega$ détermine si la partie est gagnée par Ève ($\rho \in \Omega$) ou Adam ($\rho \notin \Omega$).

La condition de victoire peut être définie de différentes manières. Celles qui nous intéressent particulièrement sont les conditions régulières (Ω est un langage rationnel sur l'alphabet V) qui sont un cas particulier des conditions boréliennes. Les conditions régulières sont celles où l'ensemble de parties gagnantes peut être décrit par un ω -automate (Section 2.1.1).

Parmi les conditions régulières, les *conditions de Muller* et de *parité* présentent un intérêt particulier. Les premières parce que n'importe quelle condition régulière peut s'y réduire. Les secondes parce qu'elles fournissent des jeux où le gagnant peut se contenter de stratégies sans mémoire, comme on le verra ci-dessous.

On peut alors se demander s'il est possible à l'un des joueurs de gagner quelles que soient les actions de son adversaire. Si c'est le cas, la question se pose de savoir si sa « stratégie », la recette qui lui indique quoi jouer en fonction du passé, est effectivement calculable, et quelle mémoire elle nécessite.

Théorème 8 (Martin [Mar75]). *Les jeux boréliens sont déterminés.*

Définition 9. *Une stratégie (pure) pour Ève est une fonction $\sigma : V^*V_E \rightarrow E$. Pour chaque partie finie ρ se terminant dans un sommet d'Ève, σ donne le prochain coup à jouer.*

Une partie (finie ou infinie) ρ est *cohérente* avec une stratégie σ si pour tout $i < |\rho|$, $\rho_{i-1} \in V_E$ implique $(\rho_{i-1}, \rho_i) = \sigma(\rho_0 \dots \rho_{i-1})$.

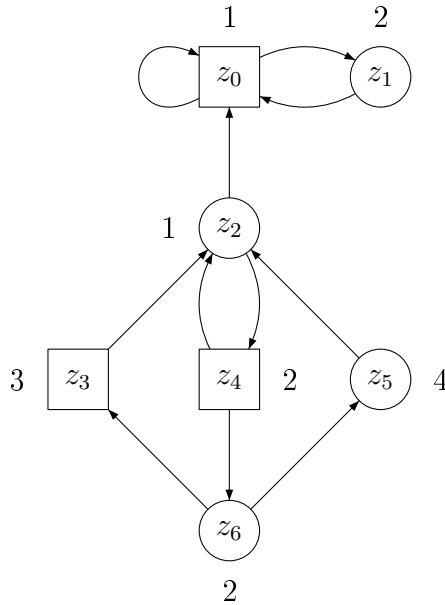


FIGURE 2.2 – Arène munie de priorités

Afin de représenter les stratégies de manière finie, on s'intéresse aux stratégies avec mémoire, définies comme suit : une *stratégie avec mémoire* M est donnée par deux fonctions $\sigma^n : M \times V_E \rightarrow E$ et $\sigma^u : M \times V \rightarrow M$, et un état de mémoire initial $m_0 \in M$. Les fonctions σ^n et σ^u donnent respectivement le prochain coup à jouer (lorsque le sommet courant est contrôlé par Ève), et le prochain état de mémoire. On peut trivialement représenter toute stratégie comme une stratégie avec mémoire V^* . Inversement, une stratégie avec mémoire $(\sigma^n, \sigma^u, m_0)$ définit bien bien une stratégie σ par :

- $\sigma(q) = \sigma^n(m_0, q)$
- $\sigma(q_0 \dots q_n) = \sigma^n(\sigma^u(\sigma^u(\dots \sigma^u(m_0, q_0), \dots), q_{n-1}), q_n)$

On peut définir de même les stratégies pour Adam, et les stratégies avec mémoire pour Adam.

Les stratégies à mémoire finie présentent un intérêt particulier. Ce sont en effet celles qui représentent un contrôleur implémentable comme un transducteur fini. Enfin les stratégies les plus simples, celles où le coup à jouer ne dépend que du sommet courant, sont appelées *positionnelles*, ou sans mémoire.

Théorème 10 ([EJ91, Mos91, Zie98]). *Les jeux de parité sont positionnels.*

Si P est un sous-ensemble de sommets, on appelle *attracteur* pour Ève vers P la région depuis laquelle Ève peut s'assurer de visiter P après un nombre fini de coups. On note cette région $\text{Attr}_E(P)$, et on définit de même l'attracteur pour Adam vers P , $\text{Attr}_A(P)$.

On appelle *piège* pour Ève une région dont Ève ne peut pas s'échapper. On notera en particulier que le complémentaire de $\text{Attr}_E(P)$ est toujours un piège pour Ève.

L'idée de la preuve du Théorème 10 est d'une part que dans $\text{Attr}_i(P) \setminus P$, le joueur i peut attirer son adversaire vers P sans mémoire, et d'autre part que gagner un jeu de parité est possible en combinant (spatialement) des stratégies d'attracteurs.

Posons $X = \text{Attr}_E^G(\chi^{-1}(0))$. L'ensemble $Q \setminus X$ est un piège pour Ève et le jeu restreint à ces sommets forme une sous-arène avec une couleur de moins que G . Il peut donc être résolu par induction. La région gagnante pour Adam dans le jeu restreint à $Q \setminus X$ l'est aussi dans G , de même que son attracteur pour Adam. Si cette région est vide, alors G est tout entier gagnant pour Ève. Sinon, on peut recommencer la procédure sur une arène strictement plus petite.

Dans sa région gagnante de $Q \setminus X$, Adam peut jouer positionnellement par induction. Dans l'attracteur il peut jouer positionnellement par la propriété mentionnée ci-dessus. Si Ève gagne partout dans $Q \setminus X$, alors elle peut jouer une stratégie attracteur vers $\chi^{-1}(0)$ dans X , et une stratégie (positionnelle) gagnante dans $Q \setminus X$ pour gagner dans G .

Corollaire 11. *Trouver le vainqueur d'un jeu de parité est dans $\text{NP} \cap \text{CO-NP}$.*

Ce résultat est immédiat puisqu'il existe un nombre polynomial de stratégies positionnelles. Il suffit donc de deviner une stratégie gagnante, et de tester le vide de l'automate (ou jeu à un seul joueur) obtenu en fixant cette stratégie.

Théorème 12. *Les jeux de Muller sont PSPACE.*

On peut obtenir ce résultat par réduction aux jeux de parité, en utilisant le LAR, ou « latest appearance record », de Gurevich et Harrington [GH82]. Ce résultat dépend évidemment de la façon dont les ensembles gagnants sont représentés. En particulier s'ils sont représentés explicitement le problème devient PTIME [Hor08].

2.3 Ordinaux et ordres

Le modèle de temps le plus classique aussi bien pour les automates que pour les jeux est ω . Cependant il est possible d'aller au delà. Büchi avait déjà défini des automates acceptants des mots indexés par des ordinaux [Bü65]. De même Wojciechowski [Woj85] et Choueka [Cho78] ont proposé des modèles d'automates reconnaissant des séquences transfinites.

2.3.1 Ordres linéaires

On dit qu'une relation d'ordre $<$ sur un ensemble J est totale si pour tous $i, j \in J$ on a $i < j$ ou $i = j$ ou $j < i$, et dans ce cas on dit que $(J, <)$ est un *ordre linéaire*. Lorsque le contexte le permettra sans ambiguïté, on omettra de préciser la relation d'ordre, et on écrira J pour $(J, <)$.

La relation $<$ est un bon ordre sur J si pour tout sous ensemble non-vide $I \subseteq J$, I a un plus petit élément. Un *ordinal* est une classe de bons ordres isomorphes.

Étant donnés deux ordres linéaires $(J_1, <_1)$ et $(J_2, <_2)$, que l'on peut supposer disjoints, on définit leur *somme* $(J, <)$, notée $(J_1, <_1) + (J_2, <_2)$, ainsi : J est $J_1 \cup J_2$, et $<$ est $<_1 \cup <_2 \cup \{(j_1, j_2) \mid j_1 \in J_1 \text{ et } j_2 \in J_2\}$. Autrement dit $J_1 + J_2$ est l'ordre lexicographique sur $(1, J_1) \cup (2, J_2)$. On peut encore définir $J = \sum_{i \in I} J_i$ pour un ordre linéaire I et une famille d'ordres linéaires $(J_i)_{i \in I}$.

Le *produit* des ordres J_1 et J_2 , noté $J_1 \times J_2$, est alors $\sum_{i \in J_2} J_1$. On pourra également utiliser la notation exponentielle $J_1^{J_2}$, avec J_2 un ordinal, pour $J_1 \times J_1 \cdots \times J_1$ répété J_2 fois.

L'ensemble des *coupures* de J , noté \hat{J} , est l'ensemble des partitions $c = (K, L)$ de J telles que $k < \ell$ pour tous $k \in K, \ell \in L$. Cet ensemble est muni d'un ordre total défini par $(K, L) < (K', L')$ si $K \subsetneq K'$. On peut également ordonner de façon naturelle l'ensemble $J \cup \hat{J}$ en posant $k < (K, L)$ si et seulement si $k \in K$. On note $c_{\min} = (\emptyset, J)$ et $c_{\max} = (J, \emptyset)$, et \hat{J}^* l'ensemble \hat{J} privé de ses deux extrémités.

Pour tout élément j d'un ordre J , il existe deux coupures c_j^- et c_j^+ qui précède (resp. suit) immédiatement j . Formellement $c_j^- = (\{k \in J \mid k < j\}, \{k \in J \mid j \leq k\})$ et $c_j^+ = (\{k \in J \mid k \leq j\}, \{k \in J \mid j < k\})$. En revanche une coupure n'a pas forcément de prédécesseur ou de successeur. Un ordre est dit *complet*, ou Dedekind-complet, si les coupures peuvent toutes s'écrire

comme c_{\min} , c_{\max} , c_{j^-} ou c_{j^+} . Une coupure qui ne s'écrit pas sous cette forme est appelé un *trou* dans l'ordre J .

On considère des mots indexés par des ordres linéaires (ou des ordinaux). Étant donné un mot $x = (x_k)_{k \in J}$ on appelle l'ordre J la *longueur* de x .

On définit $\lim_{c^-} x = \{a \mid \forall j < c \exists j < k < c x_k = a\}$ et $\lim_{c^+} x = \{a \mid \forall c < j \exists c < k < j x_k = a\}$. L'ensemble $\lim_{c^-} x$ est appelé ensemble cofinal à gauche de c , et $\lim_{c^+} x$ est appelé ensemble cofinal à droite de c : ce sont les lettres qui apparaissent arbitrairement proches de c dans le mot x .

De même que pour les mots finis, on pose l'opération fondamentale de concaténation de deux mots x et y , comme le mot $x \cdot y$ indexé par $|x| + |y|$, avec $(x \cdot y)_i = x_i$ si $i \in |x|$ et $(x \cdot y)_i = y_j$ si $i = |x| + j$.

Chapitre 3

Automates et logique temporelle sur les ordres linéaires

Les automates se sont imposés comme une abstraction particulièrement utile pour modéliser des systèmes et les étudier. L'évolution du système dans le temps est vue comme un mot sur un certain alphabet, représentant la suite des états de contrôle. Les liens forts entre automates et logique permettent de raisonner sur ces séquences et de décrire des propriétés importantes des systèmes, par exemple : toutes les exécutions évitent-elles les états d'erreur ? toutes les exécutions atteignent-elles un but donné ?

Les mots infinis fournissent un modèle très utile pour suivre l'exécution d'un système qui doit fonctionner sans interruption. Mais ω ne permet pas de décrire tous les modèles de temps possibles, et il y a des situations qui ne peuvent pas se produire dans un mot de longueur ω : soit qu'elles dépendent de la densité du modèle, ou de l'existence de trous, ou simplement du fait que tout instant n'ait pas forcément de successeur.

On reprend ici certains objets bien connus dans le cas des mots infinis, en les étendant au cas de mots indexés par des ordres linéaires. Qu'il s'agisse de formalismes logiques ou d'automates finis, on peut définir ces objets de manière similaire au cas infini. Ceci est fait dans la Section 3.1. En revanche certains résultats ne s'appliquent plus : en particulier la satisfaisabilité de MSO sur les mots indexés par des ordres quelconques est indécidable, et MSO n'a pas la même expressivité que les automates. La Section 3.2 rappelle certains résultats obtenus sur les langages de mots indexés par des ordinaux ou des ordres, du point de vue de la logique (propositionnelle ou temporelle) comme du point de vue des automates. Dans la Section 3.3 on montre, en

utilisant des automates, que la logique LTL reste satisfaisable sur des ordres linéaires arbitraires.

3.1 Définitions

3.1.1 Automates

Déjà Büchi proposait un modèle d'automates acceptant des mots indexés par des ordinaux [Bü65]. Un modèle similaire est introduit plus tard par Choueka [Cho78]. L'idée générale est d'ajouter des transitions « limites » permettant à l'exécution de l'automate de dépasser un ordinal limite. Initialement restreints à des « petits » ordinaux (inférieurs à ω^ω), on peut ensuite s'intéresser à des automates qui ne restreignent pas la longueur des mots reconnus.

Choueka a donc introduit des automates reconnaissant des mots indexés par des ordinaux de rang fini, c'est-à-dire inférieurs à ω^ω . Les opérations rationnelles qui correspondent à ces automates sont l'union, le produit fini et l' ω -produit.

Définition 13 (Automates ordinaux de Choueka [Cho78]). *Un automate de Choueka de rang n est structuré en $n + 1$ niveaux. Étant donné un ensemble d'états Q on note $[Q]^0 = Q$, et $[Q]^{n+1} = \mathcal{P}([Q]^n)$. L'ensemble des transitions de l'automate est un sous-ensemble de $\cup_{i=0}^n [Q]^i \times \Sigma \times Q$. Un automate ordinal de Choueka de rang n est donné par $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$ où*

- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $I \subseteq Q$ est l'ensemble des états initiaux
- $F \subseteq \cup_{i=0}^n [Q]^i$ est l'ensemble d'états finaux
- $\Delta \subseteq \cup_{i=0}^n [Q]^i \times \Sigma \times Q$ est la relation de transition

Ces automates contiennent n niveaux d'états limites, le niveau k étant atteint après ω^k actions. Une *exécution* d'un tel automate sur un mot w de longueur $\alpha \leq \omega^n$ est donnée par un étiquetage ρ de $\alpha + 1$ par $\cup_{i=0}^n [Q]^i$, tel que

- $\rho_0 \in I$ et $\rho_\alpha \in F$
- pour tout $i + 1 < \alpha$, $(\rho_i, w_i, \rho_{i+1}) \in \Delta$
- pour tout $\beta + \omega^k < \alpha$, $\rho_{\beta + \omega^k}$ est l'ensemble des états vus infiniment souvent parmi $\{\rho_{\beta + \omega^{k-1}.i} \mid i < \omega\}$.

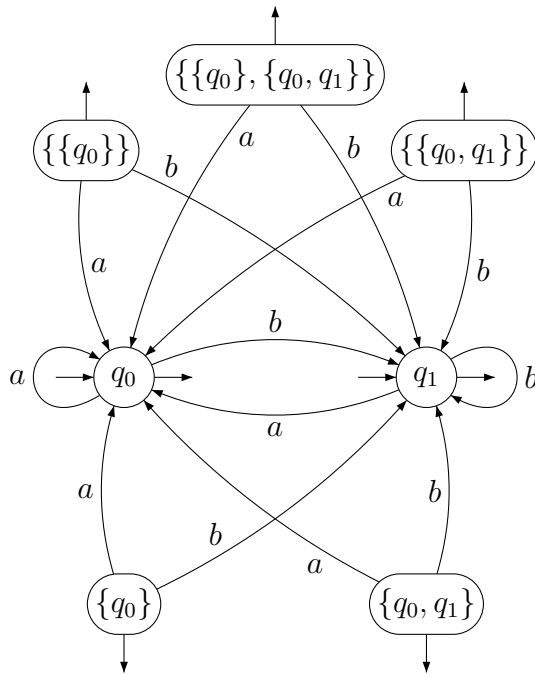


FIGURE 3.1 – Exemple d’automate de Choueka

L’automate décrit en Figure 3.1 reconnaît les mots de longueur inférieure à ω^3 sur l’alphabet $\{a, b\}$ qui ne contenant pas comme facteur infini sans a .

Choueka a montré un théorème « à la Kleene » (équivalence entre langages reconnus par automate et définissables par expression rationnelle) pour les langages rationnels de mots indexés par des ordinaux de rang finis. Cependant cette définition d’automates ne s’étend pas bien aux ordinaux plus grands ou aux ordres linéaires en général, on s’intéressera donc plutôt au modèle suivant, défini par Büchi pour les mots indexés par des ordinaux dénombrables. Pour plus de détails sur les automates de Choueka on pourra se référer à la thèse de Nicolas Bedon [Bed98].

Définition 14 (D-automates). *Un D-automate \mathcal{A} est un tuple $(\Sigma, Q, I, F, \Delta)$ où*

- Σ est un alphabet fini

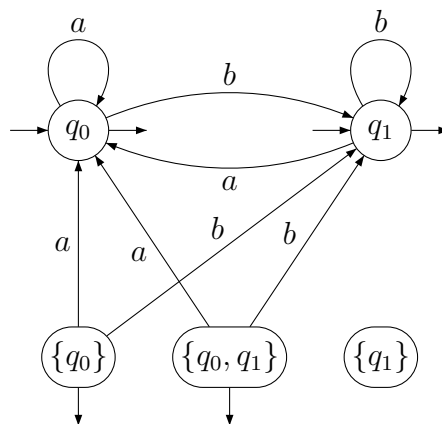


FIGURE 3.2 – D-automate

- Q est un ensemble fini d'états
- $I \subseteq Q$ et $F \subseteq Q \cup \mathcal{P}(Q)$ sont les états initiaux et finaux
- $\Delta \subseteq (Q \cup \mathcal{P}(Q)) \times \Sigma \times Q$ est la relation de transition

Une *exécution* sur un mot w de longueur α est un étiquetage ρ de $\alpha + 1$ tel que :

- pour tout $i \in \alpha$, $(\rho_i, w_i, \rho_{i+1}) \in \Delta$
- pour tout ordinal limite $\beta < \alpha$, $\rho_\beta \subseteq Q$ est l'ensemble des q tels qu'il existe une suite $(\gamma_n)_{n < \omega}$ cofinale à β telle que $\rho_{\gamma_n} = q$

Elle est acceptante si $\rho_0 \in I$ et $\rho_\alpha \in F$.

La Figure 3.2 montre un D-automate acceptant les mots sur un ordinal dénombrable qui ne contiennent pas de suite infinie de b .

Les automates de Choueka de rang n ne permettent d'accepter que des mots indexés par des ordinaux inférieurs à ω^n , ce qui n'est pas le cas des D-automates, qui peuvent reconnaître des mots indexés par ordinaux dénombrables. Les deux modèles reconnaissent les mêmes langages si l'on se restreint aux ordinaux de rang fini. Büchi a montré que les langages de mots indexés par des ordinaux dénombrables et reconnus par D-automate l'étaient aussi par D-automate complet et déterministe. Un théorème de Kleene donne également une correspondance entre D-automates et expressions rationnelles

pour cette classe de langages [Woj85].

On peut éliminer la distinction entre états de niveau 0 et états de niveau 1 dans le modèle de Büchi en introduisant explicitement des transitions limites, et ainsi abandonner la limitation aux ordinaux dénombrables, ce qui donne la définition suivante.

Définition 15 (Automates sur les ordinaux). *Un automate sur les ordinaux (ou B-automate) est donné par $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$ où*

- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $I \subseteq Q$ et $F \subseteq Q$ sont les ensembles d'états initiaux et finaux
- $\Delta \subseteq (Q \times \Sigma \times Q) \cup (\mathcal{P}(Q) \times Q)$ est la relation de transition

Pour les mots indexés par des ordinaux dénombrables, les D-automates et B-automates sont équivalents.

Bruyère et Carton ont étendu cette approche aux ordres linéaires arbitraires [BC07]. Pour ce faire, on a besoin, en plus des transitions limites à gauche ($P \rightarrow q$) des automates sur les ordinaux, de transitions limites à droite ($q \rightarrow P$).

Définition 16 ([BC07]). *Un automate sur les ordres linéaires est un tuple $(Q, \Sigma, \delta, I, F)$ où δ comprend des transitions successeur, de la forme $p \xrightarrow{a} q$, des transitions limites à gauche $P \rightarrow q$ et des transitions limites à droite $q \rightarrow P$.*

Une exécution d'un automate sur un mot x indicé par l'ordre J est un étiquetage ρ de \hat{J} par Q tel que :

- $\rho_{c_j^-} \xrightarrow{x_j} \rho_{c_j^+} \in \delta$ pour tout $j \in J$
- $\lim_{c^-} \rho \rightarrow \rho_c \in \delta$ pour toute coupure $c \in \hat{J}$ sans prédécesseur
- $\rho_{>c} \rightarrow \lim_{c^+} \rho \in \delta$ pour toute coupure $c \in \hat{J}$ sans successeur

Une exécution ρ est *acceptante* si $\rho_{c_{\min}} \in I$ et $\rho_{c_{\max}} \in F$. Un mot x est *reconnu* par l'automate \mathcal{A} s'il existe une exécution acceptante de \mathcal{A} sur x . Le *langage* défini par \mathcal{A} est l'ensemble des mots reconnus par l'automate.

Cette définition généralise à la fois les automates finis (qui n'ont pas de transitions limites), les automates de Muller (que l'on peut représenter comme des automates avec un seul état final f et des transitions $P \rightarrow f$ pour les ensembles P gagnants), et les automates sur les ordinaux (sans

transitions $q \rightarrow P$). L'étiquetage des coupures correspond également au modèle classique, où l'on a un état initial $\rho_{c_{\min}}$, des transitions $\rho_{c_j^-} \xrightarrow{x_j} \rho_{c_j^+}$, et un état final $\rho_{c_{\max}}$.

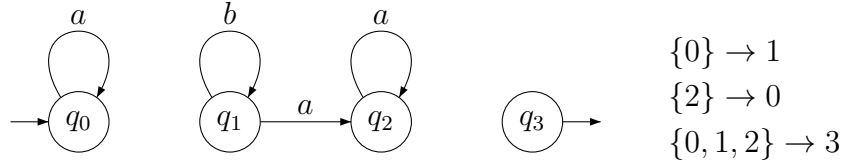
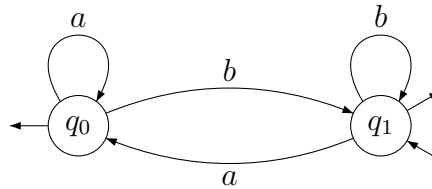


FIGURE 3.3 – Automate sur les ordinaux

La Figure 3.3 représente un automate reconnaissant le langage décrit par l'expression $(a^\omega b^* a^\omega)^\omega$. L'état initial est q_0 , depuis lequel la seule transition possible est une boucle lisant un a . Après ω transitions, la transition limite permet d'atteindre q_1 . On peut alors lire un nombre arbitraire (mais fini !) de b , puis un nombre arbitraire de a , et prendre la transition limite $\{q_2\} \rightarrow q_0$ pour revenir au point de départ. En répétant ce processus ω fois, on arrive à une position limite telle que q_0 , q_1 et q_2 sont répétés cofinalement. On peut donc prendre la transition $\{q_0, q_1, q_2\} \rightarrow q_3$, et s'arrêter.



$$q \rightarrow \{q_0\} \cup P \text{ pour tous } q \in Q, P \subseteq Q$$

$$\{q_0\} \cup P \rightarrow q_1 \text{ pour tout } P \subseteq Q$$

FIGURE 3.4 – Automate sur les ordres

L'automate représenté Figure 3.4 accepte les mots, indexés par des ordres quelconques, qui n'ont aucun sous-mot infini sans a . L'absence de transition

limite depuis et vers $\{q_1\}$ impose en effet que tout sous-mot infini contienne au moins une fois la lettre a .

Ces automates préservent certaines des propriétés des automates finis, en particulier des variantes du théorème de Kleene (pour les ordinaux [Woj85], les ordres linéaires dispersés [BC07] ou le cas général [BC06]). La clôture par complémentation est préservée pour les ordres linéaires dénombrables et dispersés [Ris04, RC05], mais pas pour les ordinaux non dénombrables. De la même façon, l'équivalence avec MSO (Théorème 5) n'est pas préservée dans le cas non-dénombrable (MSO est indécidable sur les réels), comme dans le cas non-dispersé dénombrable.

De même que pour les mots finis ou infinis, il est possible de définir des expressions rationnelles qui décrivent des langages de mots sur les ordres. Un théorème de Kleene existe également pour ces langages [BC06].

Les opérations rationnelles considérées sont :

$$\begin{aligned}
X + Y &= X \cup Y \\
X \cdot Y &= \{x \cdot y \mid x \in X, y \in Y\} \\
X^* &= \left\{ \prod_{j \in \{1, \dots, n\}} x_j \mid n \in \mathbb{N}, x_j \in X \right\} \\
X^\omega &= \left\{ \prod_{j \in \omega} x_j \mid x_j \in X \right\} \\
X^{-\omega} &= \left\{ \prod_{j \in -\omega} x_j \mid x_j \in X \right\} \\
X^\# &= \left\{ \prod_{j \in \alpha} x_j \mid \alpha \in \mathcal{O}, x_j \in X \right\} \\
X^{-\#} &= \left\{ \prod_{j \in -\alpha} x_j \mid \alpha \in \mathcal{O}, x_j \in X \right\} \\
X \diamond Y &= \left\{ \prod_{j \in J \cup \hat{J}^*} z_j \mid J \in \mathcal{L}, z_j \in X \text{ si } j \in J \text{ et } z_j \in Y \text{ si } j \in \hat{J}^* \right\}
\end{aligned}$$

$sh(X_1, \dots, X_n) = \textit{shuffle}$ des langages X_1, \dots, X_n (voir ci-dessous)

Les trois premières opérations (union, concaténation et étoile) donnent les

expressions rationnelles classiques pour les mots finis, la quatrième (itération infinie) est utilisée pour les ω -langages rationnels. Les opérations d'itération ordinale et « diamant » permettent de construire des langages de mots indexés par des ordres dispersés. L'opération de « shuffle » est nécessaire pour inclure les ordres denses.

Définition 17 (shuffle). *Soit X_1, \dots, X_n des ensembles de mots. Le shuffle $sh(X_1, \dots, X_n)$ est l'ensemble des mots de la forme :*

$$\prod_{j \in J} x_j$$

où J est un ordre dense et complet sans premier ni dernier élément, et est partitionné en J_1, \dots, J_n , denses, tels que $x_j \in X_k \Leftrightarrow j \in J_k$.

Dans la suite, on écrira $sh(x_1, \dots, x_n)$ pour « un élément de $sh(\{x_1\}, \dots, \{x_n\})$ ».

Théorème 18 ([BC06]). *Un langage de mots sur des ordres linéaires est rationnel (i.e. définissable par une expression rationnelle) si et seulement s'il est reconnaissable (i.e. reconnu par un automate sur les ordres linéaires).*

La transformation d'expression rationnelle à automate est comme pour les mots finis ou de longueur ω construite par induction sur la formule. Le passage d'automate à formule est plus délicat. Le résultat est donné pour les ordres dispersés et dénombrables dans [BC01], et est étendu à tous les ordres dans [BC06].

Ce théorème étend le lien fort entre expressions rationnelles et automates aux langages de mots sur des ordres linéaires. L'équivalence entre MSO et automates, en revanche, ne s'étend pas aux ordres linéaires : MSO sur les réels est indécidable [She75].

3.1.2 Logique

La logique fournit des moyens d'exprimer les propriétés que l'on souhaite pouvoir décider, vérifier ou spécifier pour le système ou le modèle considéré. Elle doit à la fois être assez expressive pour englober les propriétés intéressantes du modèle, et assez restreinte pour rester décidable. On s'intéresse ici aux propriétés qui parlent d'une exécution particulière, considérée comme un mot indexé par un certain ordre linéaire (ou ordinal). Les lettres de ce

mot sont souvent liées aux propriétés atomiques que l'on veut pouvoir tester (comme les états de contrôle du système).

Dans cette section nous considérons deux familles, d'abord quelques mots sur les logiques propositionnelles du premier ordre et du second ordre monadique, puis on s'intéresse à une logique modale, la logique temporelle linéaire.

3.1.2.1 Logiques propositionnelles

Une première façon d'exprimer les propriétés voulues est la logique propositionnelle du premier ordre, FO(<). La syntaxe des formules est la suivante :

$$\phi, \psi ::= \exists x\phi \mid \neg\phi \mid \phi \vee \psi \mid a(x) \mid x < y$$

On dit qu'un mot w vérifie une formule ϕ avec la valuation F (qui donne, pour chaque variable libre de ϕ , l'indice correspondant dans w), et on note $w, F \models \phi$, en suivant la sémantique suivante :

$$\begin{aligned} w, F \models a(x) & \text{ si } a \in w_{F(x)} \\ w, F \models x < y & \text{ si } F(x) < F(y) \\ w, F \models \phi \vee \psi & \text{ si } w, F \models \phi \text{ ou } w, F \models \psi \\ w, F \models \neg\phi & \text{ si } w, F \not\models \phi \\ w, F \models \exists x\phi & \text{ si } w, F \cup \{x \mapsto i\} \models \phi \text{ pour un certain } i \in |w| \end{aligned}$$

Si ϕ est close on note $w \models \phi$ pour $w, \emptyset \models \phi$.

Certaines propriétés (par exemple « w est de longueur paire ») ne peuvent pas être exprimées dans ce langage, mais peuvent l'être si on introduit des quantifications monadiques du second ordre. On ajoute donc au langage des formules de la forme $\exists X\phi$ et $x \in X$ pour obtenir la logique monadique du second ordre, MSO(<) :

$$\begin{aligned} w, F \models \exists X\phi & \text{ si } w, F \cup \{X \mapsto J\} \models \phi \text{ pour un certain } J \subseteq |w| \\ w, F \models x \in X & \text{ si } F(x) \in F(X) \end{aligned}$$

Il est bien connu que dans le cas des mots finis (resp. infinis) les propriétés définissables en MSO correspondent aux langages reconnus par automates finis (resp. automates de Büchi), et que les langages définissables au premier ordre sont ceux reconnus par des automates apériodiques. La clôture par complémentation de la classe de langages reconnus par un modèle d'automates est une condition nécessaire pour son équivalence avec une logique, et c'est souvent la partie la plus difficile, d'où l'importance des travaux liés à cette complémentation.

3.1.2.2 Logique temporelle

Afin de raisonner sur des structures linéaires, en particulier sur le comportement de systèmes réactifs, la logique temporelle linéaire (« linear temporal logic » ou LTL) est particulièrement adaptée. Sur les mots infinis, la satisfaisabilité est décidable en espace polynomial [SC85], alors que FO (qui est équivalent en terme d'expressivité) a une complexité non élémentaire [Sto74]. De plus la syntaxe est un peu plus légère que celle de la logique propositionnelle, et certaines propriétés courantes y sont plus facilement exprimables. Les variables et les quantificateurs disparaissent des formules, et sont remplacés par des modalités temporelles.

Les formules LTL sont de la forme $\phi, \psi ::= p \mid \phi \vee \psi \mid \phi \mathcal{U} \psi \mid \phi \mathcal{S} \psi$ et peuvent être évaluées à chaque position du modèle. Les connecteurs temporels \mathcal{U} et \mathcal{S} sont appelés « until » et « since ». On dit que la formule ϕ est satisfaite à la position i d'un mot w , et on note $w, i \models \phi$, suivant la sémantique suivante :

$$\begin{array}{ll}
x, i \models p & \text{si } p \in x_i \\
x, i \models \neg \psi & \text{si } x, i \not\models \psi \\
x, i \models \psi_1 \vee \psi_2 & \text{si } x, i \models \psi_1 \text{ ou } x, i \models \psi_2 \\
x, i \models \psi_1 \mathcal{U} \psi_2 & \text{si il existe } j > i \text{ tel que } x, j \models \psi_2, \\
& \text{et } x, k \models \psi_1 \text{ pour tout } k \text{ tel que } i < k < j \\
x, i \models \psi_1 \mathcal{S} \psi_2 & \text{si } \neg x, i \models \psi_1 \mathcal{U} \psi_2 \text{ où } \neg x \text{ est le mot inversé } (a_j)_{j \in -J}
\end{array}$$

La sémantique « stricte » utilisée ici pour « until » est différente de celle utilisée couramment : $x, i \models \psi_1 \mathcal{U}^{ns} \psi_2$ s'il existe $j \geq i$ tel que $x, j \models \psi_2$ et $x, k \models \psi_1$ si $i \leq k < j$. Cependant, on peut exprimer $\psi_1 \mathcal{U}^{ns} \psi_2$ comme $\psi_2 \vee (\psi_1 \wedge \psi_1 \mathcal{U} \psi_2)$, alors que l'inverse n'est pas vrai dans le cas général (dans le cas des mots de longueur ω , $\psi_1 \mathcal{U} \psi_2$ est équivalent à $X(\psi_1 \mathcal{U}^{ns} \psi_2)$).

Les autres connecteurs temporels usuels peuvent être définis à partir de l'opérateur \mathcal{U} : « next ϕ » s'écrit $X\phi = \perp \mathcal{U} \phi$; « eventually ϕ » est $F\phi = \phi \vee (\top \mathcal{U} \phi)$, et « always ϕ » est $G\phi = \neg F \neg \phi$.

Étant donné un mot w de longueur J et une formule LTL ϕ , on définit le *mot de vérité* de ϕ sur w comme le mot $v_\phi(w)$ de longueur J sur l'alphabet $\{0, 1\}$ où la position j est étiquetée par 1 si et seulement si $w, j \models \phi$. On dit qu'une formule est *valide* si pour tout mot w , $v_\phi(w)$ ne contient que des 1.

Elle est *satisfaisable* s'il existe un mot w tel que $v_\phi(w)$ contient au moins un 1.

Exemple 19. *Considérons la formule $\phi = \neg a \wedge (G \neg X a)$. Pour le mot $u = (a\emptyset)^\omega$, le mot de vérité est $v_\phi(u) = 0^\omega$ puisque à chaque position i , soit a est vrai soit il est vrai à la position suivante $i+1$. Pour le mot $w = a\emptyset^\omega a\emptyset^\omega$, le mot de vérité est $v_\phi(w) = 01^\omega 01^\omega 0$: aux positions 0 , ω et $\omega \cdot 2$, la proposition atomique a est vraie donc ϕ est fausse ; à toutes les autres positions, a est fausse, et la sous-formule $X a$ est fausse partout.*

3.1.2.3 Liens avec les automates

Il est courant en vérification d'avoir à assurer que toutes les exécutions d'un système vérifient une certaine contrainte. Si on peut exprimer la contrainte par une formule de LTL et représenter les exécutions possibles du système comme le langage reconnu par un automate, on réduit le problème au vide de l'automate reconnaissant le produit du système et de la négation de la formule, en construisant un automate qui rejette les mots vérifiant la formule.

De la même façon, si l'on recherche un contre-exemple à une propriété, on va chercher à trouver une exécution du système qui vérifie la négation de cette propriété. On se ramène encore une fois au vide du langage correspondant à un produit système-formule.

Souvent on ne construit pas l'automate correspondant à la formule explicitement, mais à la volée, ce qui permet de diminuer la complexité (typiquement PSPACE plutôt que EXPTIME).

3.2 État de l'art

On récapitule ici un certain nombre de résultats connus sur les modèles de temps étendus. On s'intéresse à la fois aux différentes logiques et aux modèles d'automates pour lesquels les problèmes de satisfaisabilité et de test du vide sont primordiaux. On évoque les questions de complexité, mais aussi les liens (et possibles équivalences) entre les formalismes en terme d'expressivité.

Un résultat fondamental est l'indécidabilité de la théorie monadique des réels, montrée par Shelah [She75] en utilisant la théorie des modèles. Cela implique l'indécidabilité de MSO sur les ordres linéaires quelconques, et il faut donc pour obtenir la décidabilité se restreindre soit à une sous-classe

d'ordres (ordinaux, ordres dénombrables, ordres dispersés, ...), soit à un fragment de la logique (comme FO ou LTL).

La décidabilité du vide des automates sur les ordinaux dénombrables a été établie par Büchi et Siefkes [BS73], ainsi que celle de la satisfaisabilité de $LTL(\mathcal{U}, \mathcal{S})$ sur les ordinaux.

On sait depuis Kamp [Kam68] que FO et $LTL(\mathcal{U}, \mathcal{S})$ ont la même expressivité sur les ordres Dedekind-complets. Pour ce qui est des automates, on sait qu'il existe des langages de mots indexés par des ordinaux quelconques qui sont reconnus par automate alors que leur complémentaire ne l'est pas [Woj84]. Cela montre que cette classe d'automates ne peut pas être caractérisée par une logique. En revanche, lorsque l'on considère les ordres dénombrables et dispersés, les automates peuvent être complétés [Ris04, RC05, Col10].

En terme de complexité, LTL est souvent plus attractive que la logique propositionnelle. La logique temporelle sur les ordres a fait l'objet d'un certain nombre de travaux, on peut citer par exemple l'ouvrage de Gabbay, Hodkinson et Reynolds [GHR94]. Reynolds s'est intéressé à la satisfaisabilité de $LTL(\mathcal{U})$ (donc sans l'opérateur passé) sur les ordres quelconques [Rey03]. Il utilise des « mosaïques » pour obtenir une complexité PSPACE. Étant donnée une formule ϕ , on note $Cl\phi$ l'ensemble $\{\psi, \neg\psi \mid \psi \text{ est une sous-formule de } \phi\}$. Une ϕ -mosaïque est un triplet (A, B, C) de sous-ensembles de $Cl\phi$ tel que A et C sont cohérents (ne contiennent pas une formule et sa négation) et maximaux, B est clos sous addition et suppression de double négation. De plus, une mosaïque doit vérifier certaines propriétés dites de cohérence. Intuitivement une mosaïque représente un intervalle du modèle recherché. A est l'ensemble des sous-formules vérifiées au point de départ de l'intervalle, C est l'ensemble des sous-formules vérifiées à l'autre extrémité, et B contient les formules vraies à tous les points intermédiaires. Pour qu'une mosaïque soit satisfaisable, il faut en particulier résoudre les problèmes suivants : si $\psi_1 \mathcal{U} \psi_2 \in A$ et $\psi_1 \notin B$, alors il faut trouver un point dans l'intervalle qui vérifie ψ_2 ; de même si $\psi \notin B$ alors il existe un point dans l'intervalle qui vérifie $\neg\psi$. Une décomposition de la mosaïque qui résout ces « défauts » montre du même coup la satisfaisabilité de la formule initiale (modulo une étape de relativisation). En utilisant des techniques similaires, Reynolds a également montré que la complexité de $LTL(\mathcal{U}, \mathcal{S})$ sur l'ordre des nombres réels était également PSPACE [Rey10].

Demri et Rabinovich ont considéré $LTL(\mathcal{U}, \mathcal{S})$ sur les ordinaux [DR07]. Les ordinaux étant des cas particuliers d'ordres complets, ce fragment a l'ex-

pressivité du premier ordre. La satisfaisabilité est là aussi un problème PS-SPACE-complet. Les auteurs montrent entre autres que s'il existe un modèle pour une formule ϕ , alors il en existe un de taille plus petite que $\omega^{|\phi|+2}$. Ils utilisent pour cela une variante des automates présentés à la section précédente, ajoutant de la structure aux états et transitions des B-automates. Les états sont des sous-ensembles d'une « base », et la traduction d'une formule logique produit un automate de taille exponentielle, mais dont la base est de taille polynomiale en la taille de la formule. De plus, les transitions limites sont exprimées en fonction de cette base, et non de l'ensemble d'états, ce qui réduit la taille de la représentation. Enfin, les auteurs fournissent une procédure permettant de tester le vide de ces automates en espace polynomial dans la taille de la base, et qui peut être utilisée pour tester la satisfaisabilité d'une formule en évitant la construction explicite (potentiellement exponentielle) de l'automate.

Sur les ordres arbitraires, qui sont l'objet de ce chapitre, le théorème de Kamp ne s'applique pas, et $LTL(\mathcal{U}, \mathcal{S})$ est strictement moins expressive que FO. Il est donc nécessaire de rajouter des opérateurs à la logique temporelle afin d'atteindre l'expressivité du premier ordre. Pour cela, on considère LTL étendue avec les connecteurs dits de Stavi [GPSS80], notés \mathcal{U}' et \mathcal{S}' , dont la sémantique est la suivante :

$x, i \models \psi_1 \mathcal{U}' \psi_2$ si on peut trouver un trou $c \in \hat{J}$ avec les propriétés suivantes :

- (1) $x, j \models \psi_1$ pour tout j tel que $i < j < c$
- (2) il n'existe pas d'intervalle commençant en c dans lequel ψ_1 est toujours vraie (*i.e.* $\forall c < k \exists c < j < k x, j \models \neg \psi_1$), et
- (3) ψ_2 est toujours vraie dans un certain intervalle commençant en c

$x, i \models \psi_1 \mathcal{S}' \psi_2$ si $\neg x, i \models \psi_1 \mathcal{U}' \psi_2$ (\mathcal{S}' est le connecteur passé correspondant à \mathcal{U}')

Dans la suite on s'intéressera à cette logique et à sa décidabilité, en utilisant des automates. On en profitera pour examiner les liens entre ces modèles dans le cadre des ordres linéaires arbitraires.

3.3 Satisfaisabilité de LTL

On donne dans cette section une preuve de décidabilité de $LTL(\mathcal{U}, \mathcal{S}, \mathcal{U}', \mathcal{S}')$ sur les ordres linéaires quelconques, en utilisant des automates sur les mots

indexés par des ordres linéaires. On commence par donner une procédure de décision de l'accessibilité dans ces automates, avant de donner un algorithme de transformation des formules en automates.

3.3.1 Accessibilité

L'idée de la preuve du Théorème 20 est due à Olivier Carton.

Théorème 20. *L'accessibilité dans un automate sur les ordres linéaires peut être décidée en temps polynomial.*

Dans un automate fini classique, tester le vide est un simple problème d'accessibilité dans un graphe. Pour les automates de Büchi il faut y ajouter la recherche d'un cycle, mais le problème reste polynomial. Dans le cas des automates (ou transducteurs) sur les ordres linéaires, le calcul nécessite certaines étapes supplémentaires, qui correspondent à la transformation d'un automate en expression rationnelle, comme décrit dans [BC06]. Pour les ordres dispersés, l'accessibilité peut être testée en temps polynomial [Car02]. Pour passer des ordres dispersés au cas général, il reste à gérer l'opération de shuffle, définie Section 3.1.1.

On commence par définir la notion de hauteur de chemin dans un automate, et donner quelques propriétés. On prouve ensuite le résultat d'accessibilité.

3.3.1.1 Hauteur de chemin dans un automate sur les ordres linéaires

On appelle *chemin* dans un automate une suite γ d'états respectant la relation de transition. Le *contenu* d'un chemin est l'ensemble des états visités, i.e. $C\gamma = \{\gamma_i \mid i \in |\gamma|\}$.

Définition 21 (Hauteur d'un chemin γ). *La hauteur $h(\gamma)$ d'un chemin γ de longueur \hat{J} est*

$$h(\gamma) = \max\{|\lim_{c^\epsilon}(\gamma)| \mid c \in \hat{J}, \epsilon \in \{+, -\}\}$$

Elle est nulle si et seulement si γ est un chemin fini.

Définition 22 (Ensembles limites maximaux d'un chemin). *Les ensembles limites maximaux d'un chemin γ sont les ensembles de cardinal maximal qui*

apparaissent dans des limites $q \rightarrow P$ ou $P \rightarrow q$ dans ce chemin, i.e. les ensembles de taille $h(\gamma)$.

$$H(\gamma) = \{P \mid |P| = h(\gamma) \text{ et } \exists c \lim_{c^\epsilon} \gamma = P \text{ avec } \epsilon \in \{+, -\}\}$$

Lemme 23. Soit γ un chemin de hauteur k dans un automate \mathcal{A} . Le chemin γ se décompose en $\gamma = \gamma_1 \dots \gamma_n$ où $h(\gamma_i) = k$ et $|H(\gamma_i)| = 1$.

Démonstration. Supposons qu'on a une suite infinie $(c_i)_{i < \omega}$ de coupures telles que $P_i = \lim_{c_i^{\epsilon_i}} \gamma$, $|P_i| = k$ et $P_i \neq P_{i+1}$. Cela signifie qu'on a une coupure c dont l'ensemble limite contient plusieurs P_i distincts et consécutifs. Cet ensemble est donc de taille au moins $k + 1$, ce qui contredit le fait que γ est de hauteur k . \square

Lemme 24. Soit γ un chemin de p à q dans l'automate \mathcal{A} , de hauteur k et tel que $|H(\gamma)| = 1$. Alors il existe un chemin γ' de p à q de hauteur au plus k , avec $C(\gamma') \subseteq C(\gamma)$, tel que

- soit $h(\gamma') < k$
- soit $\gamma' = \gamma_1 \gamma_2^{\pm\omega} \gamma_3$ où $h(\gamma_i) < k$ pour $i \in \{1, 2, 3\}$
- soit $\gamma' = \gamma_1 \gamma_2^{\zeta} \gamma_3$ où $h(\gamma_i) < k$ pour $i \in \{1, 2, 3\}$
- soit $\gamma' = \gamma_1 sh(\gamma_2 \dots \gamma_n) \gamma_{n+1}$ où $h(\gamma_i) < k$ pour $i \in \{1, 2, \dots, n\}$.

Démonstration. On pose $H(\gamma) = \{P\}$. Soit c_1 (resp. c_2) la première (resp. dernière) coupure de γ telle que $\lim_{c_i^\pm} \gamma = P$. On distingue les cas de limites à gauche et à droite en c_1 et c_2 :

1. $\lim_{c_1^-} \gamma = \lim_{c_2^+} \gamma = P$: on a γ_1 préfixe de $\gamma_{<c_1}$ se terminant en $q \in P$ et γ_2 suffixe de $\gamma_{>c_2}$ démarrant en q . On pose $\gamma' = \gamma_1 \cdot \gamma_2$ et sa hauteur est bien strictement plus petite que k .
2. $\lim_{c_1^-} \gamma = \lim_{c_2^-} \gamma = P$: on a $\gamma_3 = \gamma_{>c_2}$ de hauteur strictement inférieure à k ; de $\gamma_{<c_1}$, on peut extraire un préfixe $\gamma \cdot \gamma_2$ tel que γ_1 et γ_2 soient de hauteur strictement inférieure à k , que γ_2 ait pour contenu P et tel que γ_2^2 soit toujours un chemin. On peut alors poser $\gamma' = \gamma_1 \gamma_2^\omega \gamma_3$ avec $h(\gamma_i) < k$.
3. $\lim_{c_1^+} \gamma = \lim_{c_2^+} \gamma = P$: ce cas est symétrique du précédent, on obtient $\gamma' = \gamma_1 \gamma_2^{-\omega} \gamma_3$.
4. $\lim_{c_1^+} \gamma = \lim_{c_2^-} \gamma = P$: on suppose que $\lim_{c_1^-} \gamma \neq P$ et $\lim_{c_2^+} \gamma \neq P$. On définit sur l'intervalle $K = [c_1, c_2]$ de \hat{J} la congruence :

$$c \sim c' \Leftrightarrow \gamma[c, c'] \text{ ne contient pas de limite } P$$

On considère deux sous-cas : soit K/\sim possède deux éléments consécutifs, soit K/\sim est dense.

- Supposons d'abord que l'on a deux éléments k et k' consécutifs dans K/\sim . k et k' sont deux intervalles de \hat{J} . Comme \hat{J} est complet, soit c la coupure entre k et k' , c'est-à-dire $c = \max k = \min k'$. On a $\lim_{c^+} \gamma = P$ ou $\lim_{c^-} \gamma = P$; on peut sans perte de généralité supposer le second, l'autre cas étant symétrique. Il existe alors deux coupures c_3 et c_4 dans k telles que $\gamma_2 = \gamma[c_3, c_4]$ ait pour contenu P , $h(\gamma_2) < k$, et $\gamma_2 \cdot \gamma_2$ soit un chemin. On pose $\gamma_1 = \gamma[c_{\min}, c_1]$, $\gamma_3 = \gamma[c_2, c_{\max}]$, et $\gamma' = \gamma_1 \gamma_2^\zeta \gamma_3$.
- Supposons maintenant que K/\sim est dense. Chacun des intervalles de K/\sim est donc fermé à gauche et à droite. Toute classe k de K/\sim a la forme $\gamma[c, c']$ pour $c, c' \in \hat{J}$. À chaque $k \in K/\sim$ on associe le contenu. Comme il n'y a qu'un nombre fini de contenus, il y a un intervalle homogène dans K/\sim . On peut choisir $n - 1$ paires $(c'_2, c''_2) \dots (c'_n, c''_n)$ telles que $P = \bigcup_{i=2}^n C(\gamma[c'_i, c''_i])$. On pose alors $\gamma_1 = \gamma[c_{\min}, c_1]$, $\gamma_i = \gamma[c'_i, c''_i]$ pour $2 < i < n$, et $\gamma_{n+1} = \gamma[c_2, c_{\max}]$. La suite $\gamma' = \gamma_1 sh(\gamma_2 \cdot \gamma_n) \gamma_{n+1}$ est un chemin qui vérifie la dernière condition du Lemme 24.

□

3.3.1.2 Décidabilité de l'accessibilité

L'objet de cette section est d'établir le résultat d'accessibilité pour les automates sur les ordres (Théorème 20). Pour cela on va construire, à partir d'un automate \mathcal{A} sur les ordres, des automates finis (au sens classique, *i.e.* acceptant des mots finis) dont les chemins correspondent à ceux qui existent dans \mathcal{A} .

Notre objectif est de construire des automates finis $\mathcal{A}_0, \dots, \mathcal{A}_n$ où $n = |Q|$, vérifiant les propriétés suivantes :

Lemme 25. *Il existe un chemin de p à q de hauteur au plus k dans \mathcal{A} si et seulement s'il y a un chemin de p à q dans \mathcal{A}_k .*

Lemme 26. *On peut calculer \mathcal{A}_n en temps polynomial.*

Avant d'établir les Lemmes 25 et 26, il s'agit de définir les automates $\mathcal{A}_0, \dots, \mathcal{A}_n$. Intuitivement on construit l'automate \mathcal{A}_i en ajoutant aux transitions de \mathcal{A}_{i-1} les chemins de hauteur i possibles dans \mathcal{A} . Ces chemins sont produits

soit par une transition limite à gauche, soit par une transition limite à droite, soit par un shuffle.

Définition des automates \mathcal{A}_i . L'automate \mathcal{A}_i est $(Q, \mathcal{P}(Q), E_i, I, F)$ où E_i est défini comme suit :

$$\begin{aligned}
E_0 &= \{p \xrightarrow{\{p,q\}} q \mid \exists a \ p \xrightarrow{a} q \in E\} \\
E_i &= E_{i-1} \cup \bigcup_{|P|=i} T_P \text{ avec } T_P = T_P^\ell \cup T_P^r \cup T_P^s \\
T_P^\ell &= \{p \xrightarrow{P \cup \{q\}} q \mid p \in P, P \rightarrow q \in E \text{ et il existe un cycle dans } \mathcal{A}_{i-1} \\
&\quad \text{dont l'union des étiquettes est } P\} \\
T_P^r &= \{p \xrightarrow{\{p\} \cup P} q \mid p \rightarrow P \in E, q \in P \text{ et il existe un cycle dans } \mathcal{A}_{i-1} \\
&\quad \text{dont l'union des étiquettes est } P\} \\
T_P^s &= \{p \xrightarrow{\{p,q\} \cup P} q \mid p \rightarrow P, P \rightarrow q \in E \text{ et il existe } n \text{ chemins } \gamma_1 \dots \gamma_n \\
&\quad \text{dans } \mathcal{A}_{i-1} \text{ où } \gamma_k \text{ relie } p_k \text{ à } q_k, P \rightarrow p_k \text{ et } q_k \rightarrow P \in E \text{ et} \\
&\quad P \text{ est l'union des étiquettes des } \gamma_k\}
\end{aligned}$$

Démonstration du Lemme 25. On montre qu'il existe un chemin de p à q de contenu P et de hauteur au plus k dans \mathcal{A} si et seulement s'il existe un chemin de p à q dans \mathcal{A}_k dont l'union des étiquettes est égale à P . On commence par construire un chemin de \mathcal{A}_k pour tout chemin de \mathcal{A} de hauteur k .

On procède par récurrence sur k . Si $k = 0$, le chemin est fini, et comme les transitions de \mathcal{A}_0 sont les transitions successeur de \mathcal{A} , les chemins sont les mêmes dans \mathcal{A} et \mathcal{A}_0 . De plus comme l'étiquette d'une transition $p \rightarrow q$ dans \mathcal{A}_0 est $\{p, q\}$ le contenu d'un chemin dans \mathcal{A} est l'union des étiquettes du même chemin dans \mathcal{A}_0 .

Si $k \geq 1$, soit γ un chemin de hauteur au plus k dans \mathcal{A} . Ce chemin se décompose en $\gamma = \gamma_1 \dots \gamma_n$ où γ_i est un chemin de p_i à p_{i+1} et le contenu de γ est $P = \{p_1, \dots, p_{n+1}\}$. Soit P_i le contenu de γ_i . On commence par montrer que pour chaque i il existe un chemin de p_i à p_{i+1} dans \mathcal{A}_k tel que l'union des étiquettes soit contenue dans (mais pas nécessairement égale à) P_i .

Soit $\lambda = \gamma_i$. Par le Lemme 23 le chemin λ se décompose en $\lambda = \lambda_1 \dots \lambda_m$ où $h(\lambda_i) \leq k$ et $|H(\lambda_i)| = 1$. Soit λ_i de r_i à r_{i+1} . Il existe dans \mathcal{A} un chemin λ'_i de contenu plus petit tel que λ'_i vérifie :

- soit $h(\lambda'_i) < k$

- soit $\lambda'_i = \mu_1 \mu_2^\omega \mu_3$ ou $\mu_1 \mu_2^{-\omega} \mu_3$ ou $\mu_1 \mu_2^\zeta \mu_3$, et $h(\mu_1), h(\mu_2), h(\mu_3) < k$
- soit $\lambda'_i = \mu_1 sh(\mu_2 \dots \mu_\ell) \mu_{\ell+1}$ et $h(\mu_1), h(\mu_2 \dots \mu_\ell), h(\mu_{\ell+1}) < k$

On applique l'hypothèse de récurrence aux chemins μ_i , ce qui donne des chemins équivalents dans \mathcal{A}_{k-1} . Si $h(\lambda'_i) < k$ on a un chemin dans \mathcal{A}_{k-1} et donc dans \mathcal{A}_k . Si $\lambda'_i = \mu_1 \mu_2^\omega \mu_3$, le chemin utilise une transition du type $R \rightarrow q$ avec $|R| \leq k$. Comme μ_2 est une boucle de contenu R la transition a été ajoutée dans \mathcal{A}_k . Les cas $-\omega$ et ζ sont similaires.

Si $\lambda'_i = \mu_1 sh(\mu_2 \dots \mu_n) \mu_{n+1}$ on pose $R = \cup_{i=2}^n C(\mu_i)$. On a des transitions $R \rightarrow q$ et $q \rightarrow R$, et la transition ad-hoc a été ajoutée dans \mathcal{A}_k .

On a finalement trouvé des chemins équivalents pour les λ_i et donc pour $\lambda = \gamma_i$ (mais de contenu plus petit). On conclut en disant qu'en prenant la concaténation des chemins associés aux γ_i on trouve un chemin associé à γ . Comme les états initiaux des γ_i apparaissent nécessairement dans les étiquettes des chemins associés, l'union des étiquettes sur tout le chemin est bien égale au contenu de γ .

Pour la réciproque, il suffit de montrer que les transitions de \mathcal{A}_k correspondent bien à des chemins dans \mathcal{A} . Pour celles qui sont déjà dans \mathcal{A}_{k-1} l'hypothèse de récurrence s'applique, et pour celles qui sont ajoutées on a facilement des chemins de \mathcal{A} par construction grâce aux cycles dans \mathcal{A} . \square

Démonstration du Lemme 26. On désigne par n le nombre d'états de \mathcal{A} , et par m son nombre de transitions. Soit k entre 0 et n .

Chaque transition successeur de \mathcal{A} donne exactement une transition dans \mathcal{A}_k . Une transition limite $P \rightarrow q$ ou $q \rightarrow P$ donne au plus une transition $p \xrightarrow{P \cup \{p\}} q$ dans \mathcal{A}_k . Chaque paire $(p \rightarrow P, P \rightarrow q)$ donne au plus une transition $p \xrightarrow{P \cup \{p, q\}} q$. En résumé, si à toute paire (τ, τ') de transitions de \mathcal{A} on associe τ si τ est successeur, τ' si τ est limite et τ' est successeur, et (τ, τ') si τ et τ' sont limites, on peut borner par m^2 le nombre de transitions dans \mathcal{A}_k . Le nombre d'états de \mathcal{A}_k , lui, est toujours n .

La recherche d'un cycle dans \mathcal{A}_k , ou de ℓ chemins $\gamma_1 \dots \gamma_\ell$ est faite au plus une fois pour chaque ensemble P et donc au plus m fois. La recherche d'un cycle peut se faire en temps $n \cdot m^2$. Reste à déterminer le temps de recherche des ℓ chemins $\gamma_1 \dots \gamma_\ell$ vérifiant $\gamma_i : p_i \rightarrow q_i$ avec $P \rightarrow p_i, q_i \rightarrow P$ dans E , et $P = \cup_i C(\gamma_i)$.

Pour ce faire, on supprime toutes les transitions de \mathcal{A} qui n'ont pas leur étiquette incluse dans P pour obtenir \mathcal{A}' . On pose $I_1 = \{p \mid P \rightarrow p \in E\}$, $F_1 = \{q \mid q \rightarrow P \in E\}$. On ne garde que les états accessibles de I_1 et

co-accessibles de F_1 . On peut trouver les γ_i si et seulement si l'union des étiquettes des transitions qui restent est égale à P .

Cette condition est bien sûr nécessaire, et elle est suffisante car si R est l'étiquette de $r \xrightarrow{R} s$ on peut trouver $p \in I_1$ et $q \in F_1$ avec $\gamma_R : p \rightsquigarrow r \xrightarrow{R} s \rightsquigarrow q$. L'étiquette de γ_R est incluse dans P et contient R , donc si $P = \cup_{r \xrightarrow{R} s} R$ alors $P = \cup_{r \xrightarrow{R} s} C(\gamma_R)$. Les chemins γ_R conviennent, et le temps de recherche est linéaire.

On a donc montré que \mathcal{A}_k est calculé en temps polynomial en la taille de \mathcal{A} , et ceci est toujours valable pour \mathcal{A}_n . \square

En combinant les Lemmes 25 et 26, on sait qu'il existe un chemin de p à q dans \mathcal{A} si et seulement s'il existe un chemin de p à q dans \mathcal{A}_n , et que l'on peut déterminer l'existence de ce chemin en temps polynomial, ce qui termine la preuve du Théorème 20.

3.3.2 Transducteurs

Pour montrer la satisfaisabilité de LTL de simples automates se révèlent moins attractifs que des transducteurs, *i.e.* des automates avec sortie. En effet, une formule LTL a en quelque sorte une variable libre, l'instant courant, qui se prête bien à la transformation en transducteurs dont la sortie est la valeur de vérité de la formule à cette instant.

Curieusement il me semble que ce modèle n'ait pas ou peu été utilisé dans les travaux existants sur LTL.

3.3.2.1 Définition

Afin de décider la satisfaisabilité d'une formule LTL sur les mots infinis, ou pour le model-checking dans ce contexte, une solution consiste à utiliser des automates de Büchi. Le problème que l'on a besoin de résoudre dans ces cas est en général de savoir si une formule est satisfaite à l'instant 0, *i.e.* de savoir si $w, 0 \models \phi$. Dans le cas des mots indexés par des ordres linéaires, on ne peut pas toujours se ramener à cette question. Dans cette thèse, on propose donc d'utiliser des transducteurs pour déterminer, pour chaque position i dans le mot d'entrée w , si $w, i \models \phi$. À chaque formule ϕ on associe un transducteur lettre à lettre \mathcal{A}_ϕ , qui sur un mot d'entrée w fournit en sortie $v_\phi(w)$.

Un *transducteur* est un tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, I, F)$ où Q est un ensemble fini d'états, Σ est l'alphabet d'entrée, Γ est l'alphabet de sortie, I et F sont

des sous-ensembles de Q , et δ est l'ensemble des transitions.

Les transitions sont de trois types : transitions « successeur » $p \xrightarrow{ab} q$ où $(p, a, b, q) \in Q \times \Sigma \times \Gamma \times Q$, « limite à gauche » $P \rightarrow q$ où $(P, q) \in \mathcal{P}(Q) \times Q$, et « limite à droite » $q \rightarrow P$ où $(q, P) \in Q \times \mathcal{P}(Q)$.

Étant donné un mot $\rho = (q_j)_{j \in J}$ sur l'alphabet Q , les ensembles limites de ρ à gauche (resp. à droite) de la position j sont les étiquettes qui apparaissent arbitrairement près de j à sa gauche (resp. droite).

$$\begin{aligned} \lim_{j^-} \rho &= \{q \in Q \mid \forall k < j \exists i \ k < i < j \wedge q_i = q\} \\ \lim_{j^+} \rho &= \{q \in Q \mid \forall k > j \exists i \ j < i < k \wedge q_i = q\} \end{aligned}$$

L'ensemble limite à gauche (resp. à droite) de j est non-vide si et seulement si j n'a pas de prédécesseur (resp. successeur) et j n'est pas une extrémité. Ces ensembles permettent de définir les transitions limites dans une exécution d'un automate ou transducteur.

Une exécution acceptante du transducteur \mathcal{A} sur un mot $x = (a_j)_{j \in J}$ est un mot ρ de longueur \hat{J} sur Q tel que :

- $\rho_{c_{\min}} \in I$ et $\rho_{c_{\max}} \in F$ (état initial et final)
- pour toute position i dans J , il existe une lettre $y_i \in \Gamma$ telle que $\rho_{c_i^-} \xrightarrow{a_i | y_i} \rho_{c_i^+}$ (transition successeur)
- si une coupure $c \in \hat{J}$ n'a pas de prédécesseur, alors $\lim_{c^-} \rho \rightarrow \rho_c$ (limite à gauche)
- si une coupure c n'a pas de successeur, alors $\rho_c \rightarrow \lim_{c^+} \rho$.

3.3.2.2 Composition et produit

Soient $\mathcal{A} = (Q_1, \Sigma_1, \Gamma_1, \delta_1, I_1, F_1)$ et $\mathcal{B} = (Q_2, \Sigma_2, \Gamma_2, \delta_2, I_2, F_2)$ deux transducteurs. S'ils ont le même alphabet d'entrée ($\Sigma_1 = \Sigma_2$) on peut définir le transducteur *produit* $\mathcal{A} \times \mathcal{B}$, d'alphabet d'entrée Σ_1 et d'alphabet de sortie $(\Gamma_1 \times \Gamma_2)$. Si l'alphabet de sortie de \mathcal{A} est l'alphabet d'entrée de \mathcal{B} on définit la *composition* $\mathcal{B} \circ \mathcal{A}$, dont l'alphabet d'entrée est Σ_1 (celui de \mathcal{A}) et l'alphabet de sortie est Γ_2 (celui de \mathcal{B}).

Définition 27. *Supposons que \mathcal{A} et \mathcal{B} ont le même alphabet d'entrée. Le transducteur $\mathcal{A} \times \mathcal{B} = (Q, \Sigma, \Gamma, \delta, I, F)$ a pour alphabet d'entrée $\Sigma_1 = \Sigma_2$, et pour alphabet de sortie $\Gamma_1 \times \Gamma_2$. L'ensemble d'états Q est $Q_1 \times Q_2$, les états initiaux sont $I = I_1 \times I_2$, et les états finaux sont $F = F_1 \times F_2$. Les transitions de δ sont données par :*

- si $q_1 \xrightarrow{ab} q'_1$ et $q_2 \xrightarrow{ac} q'_2$ alors $(q_1, q_2) \xrightarrow{a|(b,c)} (q'_1, q'_2)$
- si $\pi_1(P) \rightarrow q_1$ et $\pi_2(P) \rightarrow q_2$ alors $P \rightarrow (q_1, q_2)$
- si $q_1 \rightarrow \pi_1(P)$ et $q_2 \rightarrow \pi_2(P)$ alors $(q_1, q_2) \rightarrow P$.

Définition 28. *Supposons que l'alphabet de sortie de \mathcal{A} soit l'alphabet d'entrée de \mathcal{B} . La composition de \mathcal{A} et \mathcal{B} est le transducteur $\mathcal{B} \circ \mathcal{A} = (Q, \Sigma, \Gamma, \delta, I, F)$ où $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1$, $\Gamma = \Gamma_2$, les états initiaux sont $I = I_1 \times I_2$, les états finaux sont $F = F_1 \times F_2$, et les transitions sont données par :*

- s'il existe b tel que $q_1 \xrightarrow{ab} q'_1$ et $q_2 \xrightarrow{b|c} q'_2$, alors $(q_1, q_2) \xrightarrow{a|c} (q'_1, q'_2)$
- si $\pi_1(P) \rightarrow q_1$ et $\pi_2(P) \rightarrow q_2$ alors $P \rightarrow (q_1, q_2)$
- si $q_1 \rightarrow \pi_1(P)$ et $q_2 \rightarrow \pi_2(P)$ alors $(q_1, q_2) \rightarrow P$.

Ces constructions correspondent bien au produit et à la composition des relations définies par les transducteurs. Dans le cas du produit, s'il existe une exécution de \mathcal{A} sur le mot x ayant pour sortie y , et une exécution de \mathcal{B} sur x ayant pour sortie z , alors $\mathcal{A} \times \mathcal{B}$ a une exécution sur x ayant pour sortie (y, z) . De même pour la composition, si \mathcal{A} peut avoir y en sortie sur le mot x , et \mathcal{B} peut lire y et écrire z , alors $\mathcal{B} \circ \mathcal{A}$ a une exécution sur x avec pour sortie z .

3.3.3 Satisfaisabilité

Nous présentons ici un algorithme permettant de décider la satisfaisabilité d'une formule LTL sur des mots de longueur arbitraire. Le passage par des automates permet également de décider la satisfaisabilité avec une contrainte régulière.

Théorème 29 ([Cri09]). *La satisfaisabilité de LTL sur des ordres arbitraires est décidable en espace doublement exponentiel.*

On commence par donner un algorithme qui, à partir d'une formule ϕ de LTL, construit un transducteur par induction sur la structure de ϕ . Ce transducteur produit sur tout mot d'entrée x le mot de vérité de la formule ϕ sur x . En utilisant le Théorème 20, on obtient la décidabilité de la logique. La complexité de cette procédure est détaillée en Section 3.3.3.3.

3.3.3.1 Algorithme

Les opérations de produit et de composition sur les transducteurs, définies dans la Section 3.3.2.2 vont permettre de construire un transducteur pour

chaque formule logique, en partant de quelques transducteurs de base pour les propositions atomiques et pour les opérateurs logiques.

Formules atomiques et connecteurs logiques. Pour une formule atomique $p \in AP$, le transducteur \mathcal{A}_p est $(\{q\}, 2^{AP}, \{0, 1\}, \delta, \{q\}, \{q\})$ où $\delta = \{(q \xrightarrow{a|0} q \mid p \notin a) \cup \{q \xrightarrow{a|1} q \mid p \in a\} \cup \{q \rightarrow \{q\}\} \cup \{\{q\} \rightarrow q\}$. Il regarde simplement l'entrée courante et écrit 1 si p est vraie, 0 sinon.

Les transducteurs pour la négation (\neg) et la disjonction (\vee) sont représentés sur les Figures 3.5 et 3.6.

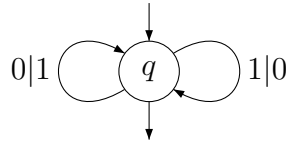


FIGURE 3.5 – Transducteur pour la négation

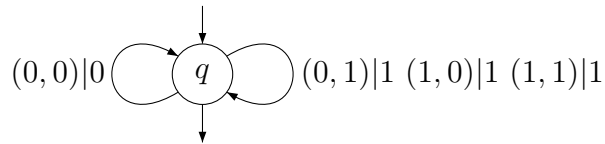


FIGURE 3.6 – Transducteur pour la disjonction

Opérateur temporel « until ». Pour l'opérateur \mathcal{U} , le transducteur est plus complexe, puisqu'il doit gérer l'aspect temporel. Rappelons que $\phi\mathcal{U}\psi$ est vraie à la position i d'un mot w s'il existe $j > i$ tel que ϕ est vraie à chaque position strictement comprise entre i et j , et ψ est vraie à la position j .

Le transducteur $\mathcal{A}_{\mathcal{U}}$ a comme alphabet d'entrée $\{0, 1\}^2$, et comme alphabet de sortie $\{0, 1\}$. Le mot d'entrée est interprété comme le couple

$(v_\phi(w), v_\psi(w))$ pour un certain mot w , et la sortie devra correspondre à $v_{\phi\mathcal{U}\psi}(w)$.

On construit un transducteur à cinq états, qui correspondent aux situations suivantes pour la coupure c :

- q_0 : la coupure c est suivie d'une position où ϕ et ψ sont vraies, *i.e.* la lettre d'entrée est $(1, 1)$
- q_1 : autres cas où $c = c_j^-$ pour un certain j , et $w, j \models \neg\phi \wedge \psi$
- q_2 : la coupure c n'a pas de successeur, mais $\phi\mathcal{U}\psi$ est vraie à la limite à droite de c
- q_3 : la coupure c est suivie d'une position où ϕ et ψ sont fausses, *i.e.* la lettre d'entrée est $(0, 0)$
- q_4 : autres cas

La structure du transducteur et ses transitions limites sont données Figure 3.7. Étant donnés deux états q et q' il existe une transition de q à q' , sauf de q_3 à q_4 et de q_4 à q_0, q_1 ou q_2 . La lettre d'entrée de chaque transition est déterminée par l'état de départ de la transition : $(1, 1)$ pour q_0 , $(0, 1)$ pour q_1 , $(0, 0)$ pour q_3 , et $(1, 0)$ pour q_2 et q_4 . La lettre de sortie dépend de l'état d'arrivée : 1 pour q_0, q_1 et q_2 , et 0 pour q_3 et q_4 . Tous les états sont initiaux, seul q_4 est final.

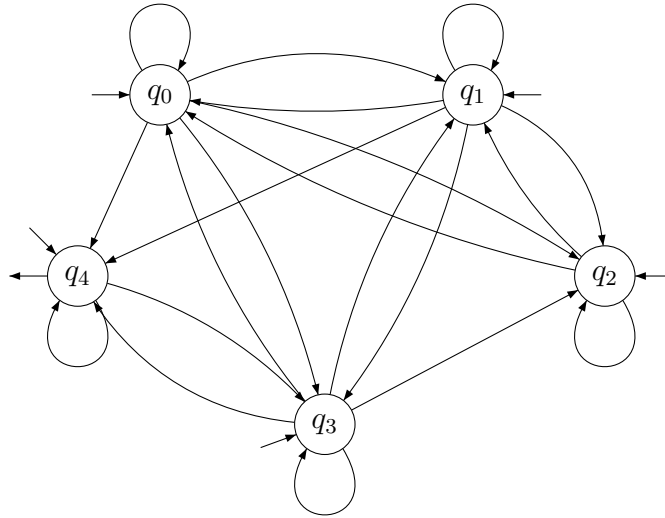
Lemme 30. *Soient ϕ et ψ deux formules. Soient x et y les mots de vérité de ϕ et ψ sur un mot w de longueur J . Le mot de sortie de $\mathcal{A}_{\mathcal{U}}$ sur l'entrée (x, y) est unique, et est le mot de vérité de $\phi\mathcal{U}\psi$ sur w .*

Démonstration. Soit ρ le mot de longueur \hat{J} sur Q défini comme suit :

- si $x_j = y_j = 1$ alors $\rho(c_j^-) = q_0$
- si $x_j = 0$ et $y_j = 1$ alors $\rho(c_j^-) = q_1$
- si $x_j = y_j = 0$ alors $\rho(c_j^-) = q_3$
- sinon, s'il existe $j > c$ tel que $y_j = 1$ et tel que $x_i = 1$ pour tout i strictement compris entre c et j , alors $\rho(c) = q_2$
- sinon $\rho(c) = q_4$

On montre que ρ est une exécution valide de $\mathcal{A}_{\mathcal{U}}$ sur (x, y) , que c'est la seule possible, et que le mot de sortie correspondant est bien le mot de vérité de $\phi\mathcal{U}\psi$ sur w .

Par définition le dernier état de ρ est q_4 , l'état final de $\mathcal{A}_{\mathcal{U}}$. Soit $c \in \hat{J}$. Si $\rho(c)$ est q_0, q_1 ou q_3 alors $c = c_j^-$ pour un certain j , et la transition successeur entre c et la coupure suivante c_j^+ est autorisée par le transducteur. Si $\rho(c) = q_2$



La transition successeur $q_i \rightarrow q_j$ est étiquetée entrée/sortie, où l'entrée est :

$$\begin{array}{ll}
 (1, 1) & \text{si } i = 0 \\
 (0, 1) & \text{si } i = 1 \\
 (0, 0) & \text{si } i = 3 \\
 (1, 0) & \text{si } i = 2 \text{ ou } 4
 \end{array}$$

et la sortie est :

$$\begin{array}{ll}
 1 & \text{si } j = 0, 1 \text{ ou } 2 \\
 0 & \text{si } j = 3 \text{ ou } 4
 \end{array}$$

Les transitions limites sont données par :

$$\begin{array}{ll}
 P & \rightarrow q_0, q_1, q_2, q_3, q_4 \text{ si } q_0, q_1 \text{ ou } q_3 \in P \\
 \{q_2\} & \rightarrow q_0, q_1, q_2 \\
 \{q_4\} & \rightarrow q_3, q_4 \\
 q_2 & \rightarrow \{q_0\}, \{q_2\}, \{q_0, q_2\} \\
 q_4 & \rightarrow P \text{ si } q_1 \text{ ou } q_3 \in P \\
 q_4 & \rightarrow \{q_4\}
 \end{array}$$

FIGURE 3.7 – Transducteur pour l'opérateur \mathcal{U}

et $c = c_j^-$ pour un certain j , alors $x_j = 1$ et $y_j = 0$, et $\rho(c_j^+)$ est q_0 , q_1 ou q_2 , ce qui est également une transition valide. Si $\rho(c_j^-) = q_4$ alors on a de nouveau $x_j = 1$ et $y_j = 0$, et $\rho(c_j^+)$ est forcément q_3 ou q_4 . Chaque transition successeur dans ρ est donc présente dans \mathcal{A}_U .

Considérons maintenant les transitions limites. Si une transition limite à gauche mène vers une coupure c , soit ψ est vraie à des positions arbitrairement proches de c , auquel cas l'ensemble limite contient q_0 ou q_1 , ou bien ψ est fausse dans un intervalle à gauche de c , et l'ensemble limite est $\{q_2\}$ ou bien un sous-ensemble de $\{q_3, q_4\}$. Si l'ensemble limite contient q_0 , q_1 ou q_3 n'importe quel état pour c est possible. S'il est égal à $\{q_2\}$, $\rho(c)$ ne peut être q_3 ou q_4 sans violer la définition de ρ , et au contraire si l'ensemble limite est $\{q_4\}$ alors $\rho(c)$ est forcément soit q_3 soit q_4 , ce qui correspond aux transitions de \mathcal{A}_U .

Si la coupure c est limite à droite, $\rho(c)$ est q_2 ou q_4 . Dans le premier cas, ϕ est vraie dans tout l'ensemble limite, qui est donc forcément inclus dans $\{q_0, q_2\}$. Dans le second cas, soit ϕ est faux à des positions arbitrairement proches de la limite, ou ψ est toujours fausse dans la limite, ce qui veut dire que soit l'ensemble limite contient q_1 ou q_3 , soit il est restreint à q_4 . Ces transitions sont autorisées par le transducteur.

Supposons que γ soit une exécution de \mathcal{A}_U sur (x, y) . Les contraintes sur les transitions successeurs imposent que si $\gamma(c)$ est q_0 , q_1 ou q_3 , alors il en est de même pour $\rho(c)$, et il faut donc montrer que les coupures marquées q_2 et q_4 sont les mêmes dans ρ et dans γ .

Si $\gamma(c) = q_2$, alors c n'est pas la dernière coupure, et il existe donc $c' > c$ tel que $\gamma(c') \neq q_2$. S'il existe une première telle coupure, alors son étiquette dans γ est q_0 ou q_1 , et on y arrive par une transition successeur depuis q_2 ou une transition limite depuis $\{q_2\}$. Sinon, il y a une transition $q_2 \rightarrow \{q_0\}$ ou $q_2 \rightarrow \{q_0, q_2\}$. Dans chacun de ces cas, la définition de ρ impose $\rho(c) = q_2$.

Si $\gamma(c) = q_4$, et $\{c' > c \mid \gamma(c') \neq q_4\}$ est vide alors $\rho(c)$ est nécessairement q_4 (ϕ est toujours vraie à droite de c , et ψ est toujours fausse). S'il existe une première coupure c' à droite de c telle que $\gamma(c') \neq q_4$ alors $\gamma(c') = q_5$ et encore une fois $\rho(c) = q_4$. Enfin si l'on a une transition $q_4 \rightarrow P$ avec q_1 ou $q_3 \in P$, la seule possibilité d'après la table de transitions du transducteur est encore $\rho(c) = q_4$. Ceci prouve que $\rho = \gamma$, et donc que \mathcal{A}_U admet exactement une exécution (et donc un mot de sortie) pour chaque mot en entrée.

La définition de ρ implique par ailleurs que ce mot de sortie unique est bien le mot de vérité de $\phi\mathcal{U}\psi$ sur w si le mot d'entrée est $(v_\phi(w), v_\psi(w))$. \square

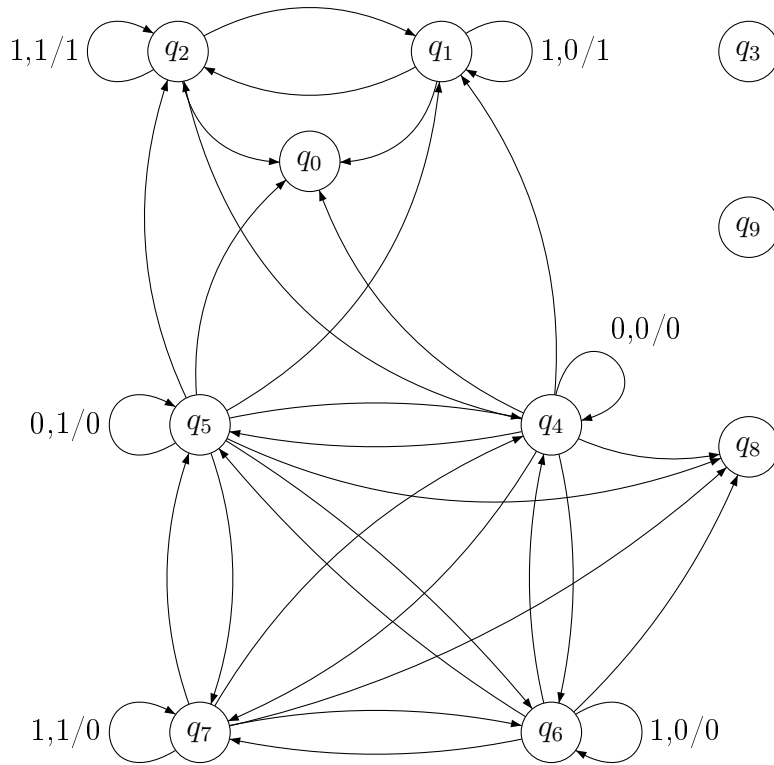
Opérateur temporel de Stavi. Le dernier opérateur à considérer est le connecteur de Stavi (\mathcal{U}'). Rappelons que $\phi\mathcal{U}'\psi$ est vraie à la position i s'il existe une coupure $g > i$ qui soit limite à gauche et à droite (*i.e.* un trou), telle que ϕ soit vraie à toute position strictement comprise entre i et g , telle que ψ soit vraie dans un intervalle débutant en g , et telle que ϕ soit fausse à des positions arbitrairement proches à droite de g .

Le point central de cette définition est la coupure g ; dans l'automate, elle correspond à l'état q_3 . Les états q_0, q_1 et q_2 correspondent à l'intervalle précédant cette coupure, dans lequel la formule est vraie. Les états q_4, q_5, q_6, q_7, q_8 et q_9 correspondent aux positions où la formule est fausse. La seule façon de quitter la région $\{q_0, q_1, q_2\}$ est donc par une transition limite vers q_3 . Les transitions successeur dans cette région ont comme entrée 1, 0 ou 1, 1. La structure de l'automate est décrite Figure 3.8. Tous les états sauf q_3 et q_9 sont initiaux; q_8 et q_9 sont finaux. L'étiquette d'entrée de chaque transition successeur est déterminée par son état de départ : (1, 1) pour q_1 et q_7 , (1, 0) pour q_2 et q_6 , (0, 0) pour q_0 , et (0, 1) pour q_5 . L'étiquette de sortie dépend de l'état d'arrivée : 1 pour q_0, q_1 et q_2 , et 0 pour q_4, q_5, q_6, q_7 et q_8 .

Étant donné un mot w sur $\{0, 1\}^2$, on définit un étiquetage des coupures de $|w|$ par les états du transducteur. Soit c une coupure, on pose :

- $\rho(c) = q_0$ si c n'a pas de successeur, et $\phi\mathcal{U}'\psi$ est vraie
- $\rho(c) = q_1$ si $c = c_j^-$ pour un certain j avec $w_j = (1, 0)$, et $\phi\mathcal{U}'\psi$ est vraie en j
- $\rho(c) = q_2$ si $c = c_j^-$ pour un certain j avec $w_j = (1, 1)$, et $\phi\mathcal{U}'\psi$ est vraie en j
- $\rho(c) = q_3$ si c est un trou, avec $\phi\mathcal{U}'\psi$ vraie avant c , et fausse après
- $\rho(c) = q_4$ si $c = c_j^-$ pour un certain j avec $w_j = (0, 0)$
- $\rho(c) = q_5$ si $c = c_j^-$ pour un certain j avec $w_j = (0, 1)$
- $\rho(c) = q_6$ si $c = c_j^-$ pour un certain j avec $w_j = (1, 0)$, et $\phi\mathcal{U}'\psi$ est fausse en j
- $\rho(c) = q_7$ si $c = c_j^-$ pour un certain j avec $w_j = (1, 1)$, et $\phi\mathcal{U}'\psi$ est fausse en j
- $\rho(c) = q_8$ si c n'a pas de successeur, soit a un prédécesseur soit ϕ n'est pas toujours vraie dans la limite à gauche, et $\phi\mathcal{U}'\psi$ est fausse à droite de c
- $\rho(c) = q_9$ si c est un trou ou est c_{\max} , si ϕ est vraie dans un intervalle à gauche de c , et si $\phi\mathcal{U}'\psi$ est fausse

Lemme 31. ρ est l'unique exécution du transducteur $\mathcal{A}_{\mathcal{U}'}$ sur son mot d'en-



Transitions limites :

- $P \rightarrow q_0, q_1, q_2$ si $P \cap \{q_4, q_5\} \neq \emptyset$ ou $P \subseteq \{q_0, q_1, q_2\}$
- $P \rightarrow q_3$ si $P \subseteq \{q_0, q_1, q_2\}$
- $P \rightarrow q_4, q_5, q_6, q_7$ si $P \not\subseteq \{q_0, q_1, q_2\}$
- $P \rightarrow q_8$ si $P \cap \{q_4, q_5\} \neq \emptyset$
- $P \rightarrow q_9$ si $P \cap \{q_4, q_5\} = \emptyset$ et $P \not\subseteq \{q_0, q_1, q_2\}$
- $q_0 \rightarrow P$ si $P \subseteq \{q_0, q_1, q_2\}$
- $q_3 \rightarrow P$ si $P \cap \{q_1, q_4, q_6\} = \emptyset$ et $q_5 \in P$
- $q_8 \rightarrow P$ si $P \cap \{q_4, q_5, q_6, q_7\} \neq \emptyset$
- $q_9 \rightarrow P$ si $P \cap \{q_4, q_5, q_6, q_7\} \neq \emptyset$ et soit $P \cap \{q_4, q_5\} = \emptyset$, soit P intersecte $\{q_1, q_4, q_6\}$

FIGURE 3.8 – Automate pour l'opérateur de Stavi futur

trée. Si cette entrée est $(v_\phi(w), v_\psi(w))$ pour un certain mot w , alors la sortie du transducteur est $v_{\phi\mathcal{U}'\psi}(w)$.

Démonstration. Commençons par prouver que ρ est bien une exécution valide. Les transitions successeurs sont vérifiables facilement, de façon similaire au transducteur $\mathcal{A}_\mathcal{U}$, de même que les états initiaux et finaux.

Supposons qu'une coupure c est limite à gauche, et notons P l'ensemble limite. Si son étiquette (dans ρ) est q_0, q_1 ou q_2 , alors soit ϕ est toujours vraie dans la limite, et donc $\phi\mathcal{U}'\psi$ l'est aussi (puisqu'elle est vraie en c), d'où $P \subseteq \{q_0, q_1, q_2\}$, soit $\neg\phi$ est vraie de façon répétée dans la limite, et donc P contient q_4 ou q_5 . Ceci correspond aux transitions du transducteur.

Si $\rho(c) = q_3$, l'ensemble P doit être inclus dans $\{q_0, q_1, q_2\}$ pour assurer que $\phi\mathcal{U}'\psi$ soit vraie dans la limite.

Si $\rho(c)$ est q_4, q_5, q_6 ou q_7 , alors $\phi\mathcal{U}'\psi$ n'est pas toujours vraie dans la limite (sans quoi elle serait toujours vraie après), donc $P \not\subseteq \{q_0, q_1, q_2\}$.

Si $\rho(c) = q_8$, le raisonnement ci-dessus s'applique toujours, et de plus comme $\neg\phi$ est vraie de façon répétée dans la limite, P contient q_4 ou q_5 .

Si $\rho(c) = q_9$, on a de même $P \not\subseteq \{q_0, q_1, q_2\}$. Comme de plus ϕ est vraie dans la limite, P ne peut contenir ni q_4 ni q_5 .

Considérons maintenant le cas où c est limite à droite, avec P pour ensemble limite dans ρ . Les seuls états possibles en c sont q_0, q_3, q_8 et q_9 . Si $\rho(c) = q_0$, alors $\phi\mathcal{U}'\psi$ est vraie dans la limite et donc $P \subseteq \{q_0, q_1, q_2\}$. Si $\rho(c) = q_3$, ψ est toujours vraie dans la limite donc P ne peut pas contenir q_1, q_4 ou q_6 . De plus $\neg\phi$ est répétée dans la limite donc $q_5 \in P$. Si $\rho(c)$ est q_8 ou q_9 , l'ensemble limite ne peut être inclus indique que P ne contient ni q_4 ni q_5 , soit $\neg\psi$ est répété, et donc P contient q_1, q_4 ou q_6 . Dans tous les cas, la transition est bien autorisée par le transducteur, et ρ est donc une exécution valide.

Supposons maintenant que γ est une exécution de $\mathcal{A}_{\mathcal{U}'}$ sur w , on peut montrer que γ et ρ sont identiques, comme on l'a fait pour $\mathcal{A}_\mathcal{U}$.

Le fait que la sortie correspond bien à la valeur de vérité de $\phi\mathcal{U}'\psi$ est directe étant donnée la définition de ρ . \square

3.3.3.2 Construction du transducteur \mathcal{A}_ϕ

Nous avons maintenant les blocs de base permettant de construire pour toute formule un transducteur donnant son mot de vérité. Si ϕ est une proposition atomique p , on utilise le transducteur \mathcal{A}_p . Si $\phi = \neg\psi$, alors

$\mathcal{A}_\phi = \mathcal{A}_\neg \circ \mathcal{A}_\psi$. Si $\phi = \psi_1 \vee \psi_2$, alors $\mathcal{A}_\phi = \mathcal{A}_\vee \circ (\mathcal{A}_{\psi_1} \times \mathcal{A}_{\psi_2})$. De même $\mathcal{A}_{\phi \mathcal{U} \psi} = \mathcal{A}_\mathcal{U} \circ (\mathcal{A}_\phi \times \mathcal{A}_\psi)$, et les équivalents pour \mathcal{S} et les opérateurs de Stavi. Par construction, on a le résultat suivant :

Lemme 32. *Pour toute formule ϕ , le transducteur \mathcal{A}_ϕ admet exactement une exécution par mot d'entrée, et sa sortie correspond au mot de vérité de ϕ .*

3.3.3.3 Complexité

Le nombre d'états du transducteur \mathcal{A}_ϕ est le produit des nombres d'états des automates élémentaires impliqués dans sa construction, et est donc exponentiel en la taille de la formule. La taille du transducteur inclut ses transitions limites, ce qui induit un passage exponentiel supplémentaire, si elles sont représentées explicitement. Le test de satisfaisabilité se réduit alors à un test d'accessibilité et co-accessibilité d'une transition étiquetée par 1 en sortie, ce qui est polynomial en la taille du transducteur par le Théorème 20. On en déduit la borne doublement exponentielle pour le problème de satisfaisabilité (Théorème 29).

On peut réduire un peu la taille du transducteur construit en utilisant d'autres connecteurs au lieu de ceux de Stavi, mais on resterait dans la même classe.

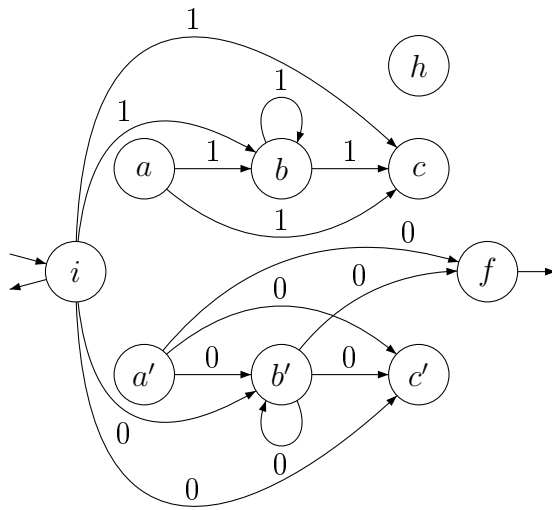
3.4 Discussion

La procédure de traduction d'une formule en transducteur a un corollaire immédiat :

Corollaire 33. *La satisfaisabilité d'une formule LTL ϕ dans un mot reconnu par un automate \mathcal{B} sur les ordres linéaires est décidable en espace doublement exponentiel.*

Il suffit en effet de calculer le produit de \mathcal{A}_ϕ et de \mathcal{B} , et de vérifier s'il existe une transition où la sortie (de \mathcal{A}_ϕ) est 1 et qui soit à la fois accessible et co-accessible. Si c'est le cas, on a une exécution dont le mot d'entrée est accepté par \mathcal{B} et dans lequel ϕ est vraie à un certain point.

Dans de récents travaux, Rabinovich [Rab10] a amélioré le résultat de complexité de la satisfaisabilité en montrant que le problème est PSPACE-complet, soit la même complexité que pour les mots infinis. Il utilise, comme



Transitions limites :

$P \rightarrow a, h$ si $P \subseteq \{a, b, c, h\}$

$c \rightarrow P$ si $P \subseteq \{a, b, c, h\}$

$P \rightarrow a', f$ pour tout P

$c' \rightarrow P$ si $P \cap \{a, b, c, h\} = \emptyset$

FIGURE 3.9 – Automate testant s’il existe un « trou » dans le futur

dans [DR07], des automates avec une base, qui permettent de réduire la taille de la représentation. La preuve consiste à réduire dans un premier temps le problème de satisfaisabilité de la logique temporelle au problème de satisfaisabilité pour les formules Φ -conjonctives pour un certain ensemble fini Φ de formules du premier ordre. Dans un deuxième temps on montre que la satisfaisabilité d’une formule ϕ sur les structures de rang $p(|\phi|)$ est dans PSPACE pour tout polynôme p , et par ailleurs qu’une formule est satisfaisable si et seulement si elle l’est sur les structures de rang polynomial en sa taille.

La construction de la Section 3.3.3.1 montre qu’il est possible de représenter toute formule de $LTL(\mathcal{U}, \mathcal{S}, \mathcal{U}', \mathcal{S}')$, et donc de FO, comme un transducteur non ambigu. Mais il est possible de construire un tel transducteur avec comme sortie le mot de vérité d’une propriété qui ne peut s’exprimer au premier ordre. Ainsi le transducteur représenté Figure 3.9 écrit 1 à la position i s’il existe un trou $g > i$, propriété qui n’est pas exprimable dans FO. Il serait intéressant d’obtenir une caractérisation logique des propriétés pouvant être exprimées avec ces transducteurs.

Sur les mots finis ou de longueur ω , LTL restreinte aux opérateurs unaires (X , F et leurs équivalents passé) est équivalente en terme d’expressivité à la logique du premier ordre avec deux variables, $FO^2(<, +1)$ [EVW02]. Si on restreint à F et son équivalent passé, on obtient une logique d’ex-

pressivité équivalente à $\text{FO}^2(<)$. Dans le cas des mots finis, $\text{FO}^2(<)$ correspond aux automates bidirectionnels « partiellement ordonnés » [STV02]. La preuve d'équivalence entre logique temporelle avec connecteurs unaires et FO^2 s'étend facilement à tous les ordres linéaires. Il serait donc intéressant d'étudier ces logiques plus en détails dans le cas des ordres, en cherchant une sous-classe d'automates de même expressivité.

Dans son travail sur $\text{LTL}(\mathcal{U})$, Reynolds utilise des mosaïques pour enregistrer quelles sous-formules doivent être satisfaites dans des intervalles particuliers et à leurs bornes, et pour trouver une décomposition qui prouve la satisfaisabilité de la formule initiale. Malheureusement, il n'est pas clair que cette technique peut s'étendre à un fragment plus important de la logique.

Chapitre 4

Jeux de longueur ordinale sur graphes finis

4.1 Introduction

Les jeux infinis sur les graphes ont été largement étudiés, notamment dans le contexte de la vérification de programmes et de synthèse de contrôleurs. Afin d'étendre ces jeux à des parties transfinies, on utilise des arènes comportant, outre des arêtes décrivant les coups possibles comme pour les jeux infinis, des transitions limites. Les notions de stratégie et de stratégie avec mémoire s'appliquent à ce nouveau modèle. Le problème de la détermination de ces jeux reste entier, puisqu'ils ne se présentent pas comme des jeux boréliens. Lorsque l'on obtient des résultats de détermination, il reste encore à déterminer s'il existe des stratégies gagnantes à mémoire finie, et à calculer effectivement ces stratégies.

Dans ce chapitre nous introduisons un modèle de jeux de longueur ordinale. On se place dans le cas de graphes (ou *arènes*) finis, et l'on montre que ces jeux sont déterminés, et que trouver le vainqueur est décidable en espace polynomial. On introduit ensuite une sous-classe pour laquelle le vainqueur peut gagner sans mémoire, et on l'utilise pour montrer qu'une mémoire finie est suffisante dans le cas général.

4.2 Définitions

Un jeu à deux joueurs est donné par un graphe fini, dont les sommets sont partitionnés entre ceux appartenant à Ève (représentés par un rond, \bigcirc) et ceux appartenant à Adam (représentés par un carré, \square).

Dans un jeu infini traditionnel, une partie est un chemin infini dans ce graphe, suivant les arêtes (transitions) autorisées. À chaque instant, le propriétaire du sommet courant choisit une transition sortante, ce qui détermine le prochain sommet.

Formellement, un jeu infini à deux joueurs est la donnée d'une arène, c'est-à-dire d'un graphe (Q, E) où Q est partitionné en Q_A et Q_E , et d'une condition de victoire W , sous-ensemble de Q^ω .

Afin d'étendre ce modèle à des parties de longueur ordinale, on ajoute au graphe des transitions limites, qui permettent d'étendre les parties après les ordinaux limites. Ces transitions limites sont semblables à celles utilisées dans les automates sur les mots indexés par des ordinaux.

Une *arène* pour un jeu de longueur ordinale est la donnée d'un graphe dont l'ensemble de sommets Q est partitionné en Q_A et Q_E , dont l'ensemble d'arêtes, ou transitions successeur, est $T \subseteq Q \times Q$, et enrichi d'une fonction de transition limite $\lambda : \mathcal{P}(Q) \rightarrow Q$. Nous considérons ici des arènes finies. On supposera par ailleurs, sans perte de généralité, que tout sommet est l'origine d'au moins une transition successeur.

Un jeu d'accessibilité est donné par une arène $(Q, Q_A, Q_E, T, \lambda)$, un sommet initial $q_0 \in Q$, et un sommet cible $\odot \in Q$.

Une partie de longueur I est une suite $\rho = (\rho_i)_{i < I}$ où I est un ordinal, vérifiant les conditions suivantes :

- $\rho_0 = q_0$
- si $i = j + 1$ alors $(\rho_j, \rho_i) \in T$
- si i est un ordinal limite, alors $\lambda(\lim_i \rho) = \rho_i$

La partie ρ est gagnée par Ève si le sommet \odot est atteint.

L'ensemble $\lim_i \rho$ désigne les sommets apparaissant arbitrairement proches de i dans ρ , *i.e.* $\{q \in Q \mid \forall j < i \exists j < k < i \rho_k = q\}$. Cette définition est analogue à celle de condition de Muller pour les jeux infinis, où le vainqueur d'une partie dépend uniquement de l'ensemble des sommets visités infiniment souvent. La notion de « infiniment souvent » est ici remplacée par « arbitrairement proche de la limite ».

Un cas particulier intéressant est obtenu en définissant les transitions

limites en utilisant des priorités. À chaque sommet q est associé une priorité $\chi(q)$, et les transitions limites sont définies par une fonction δ telle que $\lambda(P) = \delta(\min\{\chi(q) \mid q \in P\})$. Ce modèle est l'analogie des jeux de parité.

Sur un graphe donné, certaines fonctions de transition qui peuvent s'écrire comme $\lambda : \mathcal{P}(Q) \rightarrow Q$ ne sont pas représentables en utilisant des priorités. C'est le cas sur l'exemple de la figure 4.1, où la priorité donnée à l'ensemble $\{a, b, c, d\}$ ne pourrait être à la fois différente de celle donnée à $\{a, b, c\}$ et de celle de $\{a, b, d\}$ (la priorité d'un ensemble étant le minimum des priorités de ses éléments).

Les définitions ci-dessous rappellent quelques concepts classiques en théorie des jeux (en les adaptant légèrement aux jeux définis ci-dessus), qui nous seront utiles par la suite.

Définition 34 (Sous-jeu). *Soit $G = (Q, Q_E, Q_A, T, \lambda)$ un jeu (ordinal), et $\mathbb{Q} \subseteq Q$. On dit que $\mathbb{G} = (\mathbb{Q}, \mathbb{Q}_E, \mathbb{Q}_A, \mathbb{T}, \mu)$ — où $\mathbb{Q}_E, \mathbb{Q}_A, \mathbb{T}$ et μ sont les restrictions de Q_E, Q_A, T et λ à \mathbb{Q} — est un sous-jeu de G si tout sommet de \mathbb{G} a un successeur dans \mathbb{G} . On note $\mathbb{G} = G \setminus P$ si $\mathbb{Q} = Q \setminus P$.*

Cette définition assure que si l'on contraint les joueurs à rester dans \mathbb{Q} , la partie peut se poursuivre pour au moins ω coups.

Définition 35 (Attracteur). *Soit P un sous-ensemble de sommets de G . L'attracteur vers P pour Ève, noté $\text{Attr}_E^G(P)$, est l'ensemble des sommets depuis lesquels Ève peut assurer une visite à P en un nombre fini de coups. Il s'agit de l'ensemble $\cup_{i \geq 0} \text{Attr}_i$, où Attr_i est le plus petit ensemble vérifiant :*

- $\text{Attr}_0 = P$;
- $\text{Attr}_i \subseteq \text{Attr}_{i+1}$;
- si $q \in Q_E$ et si q a un successeur dans Attr_i , alors $q \in \text{Attr}_{i+1}$;
- si $q \in Q_A$ et si tous ses successeurs sont dans Attr_i alors $q \in \text{Attr}_{i+1}$.

Définition 36 (Piège). *Au contraire, un piège P pour Ève est une région dans laquelle Adam peut assurer que le jeu reste pendant ω coups. L'ensemble P doit donc vérifier les conditions (nécessaires et suffisantes) suivantes :*

- si $q \in P \cap Q_E$ tous ses successeurs sont dans P ;
- si $q \in P \cap Q_A$ il a un successeur dans P .

De la même façon on peut définir l'attracteur pour Adam vers P , et un piège pour Adam.

Définition 37 (Stratégie). Une stratégie pour Ève (resp. Adam) est une fonction σ de Q^*Q_E dans Q (resp. τ de Q^*Q_A dans Q), telle que pour toute suite w de sommets et tout sommet q , $(q, \sigma(wq)) \in T$ (resp. $(q, \tau(wq)) \in T$).

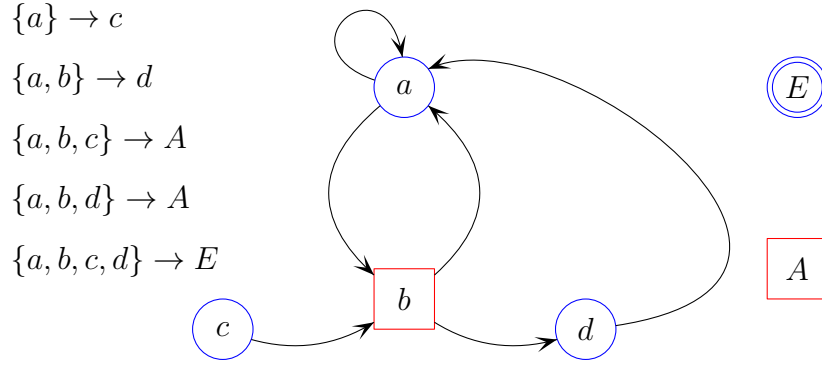


FIGURE 4.1 – Jeu ordinal

Exemple 38. La Figure 4.1 montre un exemple de jeu ordinal, avec un sommet cible pour Ève (E), un sommet cible pour Adam (A), et des transitions limites. Seules les transitions limites pouvant réellement se produire sont représentées.

Ève doit, afin d'atteindre son sommet cible, utiliser la transition $\{a, b, c, d\} \rightarrow E$. Elle doit donc s'assurer que tous les états sont visités de façon répétée, ce qui est impossible sans mémoire : soit elle joue toujours la transition $a \rightarrow a$, et Adam peut jouer $b \rightarrow a$ et empêcher toute visite à d , soit elle joue toujours $a \rightarrow b$, auquel cas Adam joue $b \rightarrow d$ et c n'est jamais visité.

En revanche, Ève peut se souvenir si d a été visité depuis la dernière visite à b . Si c'est le cas, elle joue la boucle $a \rightarrow a$, ce qui force une visite à c (et par conséquent à b). Sinon, elle joue $a \rightarrow b$, ce qui force Adam soit à jouer vers d soit à prendre la transition limite $\{a, b\} \rightarrow d$, et dans tous les cas d est visité. Cette stratégie permet donc à Ève de gagner le jeu quel que soit le comportement d'Adam.

4.3 Détermination

Le résultat principal de cette section est la détermination d'une classe de jeux ordinaux d'accessibilité. Ce résultat est obtenu en réduisant le problème

à la victoire dans un jeu de Muller. La preuve introduit une limitation sur l'arène, qui force les parties à une longueur inférieure à ω^ω .

Théorème 39 ([CH08b]). *Les jeux d'accessibilité à deux joueurs de longueur ordinale sur des arènes finies sont déterminés.*

À partir d'un jeu ordinal G on construit un jeu de Muller \mathbb{G} . La section 4.3.2 décrit cette construction, et la section 4.3.3 donne les preuves des lemmes 40 et 41.

Lemme 40. *Si Ève gagne dans \mathbb{G} depuis un sommet $q \in Q$, elle gagne aussi depuis q dans G .*

Lemme 41. *Si Adam gagne dans \mathbb{G} depuis un sommet $q \in Q$, il gagne aussi depuis q dans G .*

Le théorème 39 est obtenu grâce à ces deux lemmes et au théorème 42.

Théorème 42 ([Mar75]). *Les jeux de Muller sont déterminés.*

4.3.1 Restriction sur les arènes

On retire dans cette réduction les arènes contenant des transitions limites $P \rightarrow q$ avec $q \in P$. On les remplace par des transitions vers un sommet gagnant pour Adam.

4.3.2 Réduction aux jeux de Muller

Nous décrivons ici une réduction permettant de passer d'un jeu de longueur ordinale G à un jeu de Muller \mathbb{G} en préservant le vainqueur. L'idée est d'obliger les deux joueurs à simuler les transitions limites en prenant des « raccourcis ». De ce point de vue, cette approche est similaire à celle de Chatterjee, de Alfaro, Jurdzinski et Henzinger dans [CJH03] et [CdAH05], où les auteurs utilisent des conditions de parité pour simuler l'aléatoire pour le calcul des régions gagnantes qualitatives. Dans leur approche, comme dans celle présentée ici, bien que le vainqueur soit conservé il n'y a pas de bijection entre les parties dans le jeu réduit et dans le jeu d'origine.

L'idée principale est qu'une partie de longueur inférieure à ω^ω peut être décrite par un mot fini, où les transitions limites ont été remplacées par des

« raccourcis ». Le graphe est donc étendu avec de nouveaux sommets et transitions pour représenter ces raccourcis. Ils sont découpés en deux étapes, pour donner aux deux joueurs la possibilité de les refuser. La figure 4.2 montre le gadget utilisé pour la simulation d'une transition limite : pour chaque ensemble de sommets P contenant un sommet d'Ève, un sommet $o(P) \in P \cap Q_E$ est distingué. Deux nouveaux sommets sont ajoutés : $\chi(P)$ appartenant à Adam, et $\xi(o(P))$ appartenant à Ève. Une transition $o(P) \rightarrow \chi(P)$ permet à Ève de proposer la transition limite $P \rightarrow t(P)$, les transitions $\chi(P) \rightarrow t(P)$ et $\chi(P) \rightarrow \xi(o(P))$ permettent à Adam de respectivement accepter et refuser cette transition limite, et enfin les transitions sortantes en $\xi(o(P))$ sont les mêmes que celles de $o(P)$ dans le jeu d'origine, ce qui oblige Ève à faire avancer le jeu si jamais Adam refuse une transition limite. On garantit qu'aucun des joueurs ne peut bloquer le jeu sans le perdre, grâce à la condition de Muller : l'ensemble gagnant pour Ève contient tous les ensembles de la forme $P \cup \{\chi(P)\}$, ainsi Adam perd s'il refuse systématiquement une proposition légitime. En revanche si les sommets de P sont visités à la limite, et si Ève ne choisit pas infiniment souvent $\chi(P)$, alors Adam gagne.

Formellement, le jeu \mathbb{G} est défini à partir de $G = (Q, Q_E, Q_A, \mathcal{E}, \mathcal{T}, \odot, \otimes)$ de la façon suivante :

$$\begin{aligned}
\mathbb{Q}_E &= Q_E \cup \{\xi(q) \mid q \in Q_E\} \\
\mathbb{Q}_A &= Q_A \cup \{\chi(P) \mid P \in \mathcal{P}(Q)\} \\
\mathbb{E} &= \mathcal{E} \\
&\cup \{(o(P), \chi(P)) \mid P \in \mathcal{P}(Q)\} \\
&\cup \{(\chi(P), t(P)) \mid P \in \mathcal{P}(Q)\} \\
&\cup \{(\chi(P), \xi(o(P))) \mid P \in \mathcal{P}(Q)\} \\
&\cup \{(\xi(p), q) \mid (p, q) \in \mathcal{E}\} \\
&\cup \{(\odot, \odot), (\otimes, \otimes)\} \\
\mathbb{M} &= \{P \cup \mathbb{H} \mid P \subseteq Q, \mathbb{H} \cap Q = \emptyset, \chi(P) \in \mathbb{H}\}
\end{aligned}$$

4.3.3 Traduction de stratégies : de Muller aux ordinaux

Pour prouver les Lemmes 40 et 41, on va devoir traduire les stratégies : à partir d'une stratégie gagnante dans le jeu de Muller \mathbb{G} pour l'un des joueurs, construire une stratégie gagnante pour ce joueur dans le jeu ordinal G . Cette

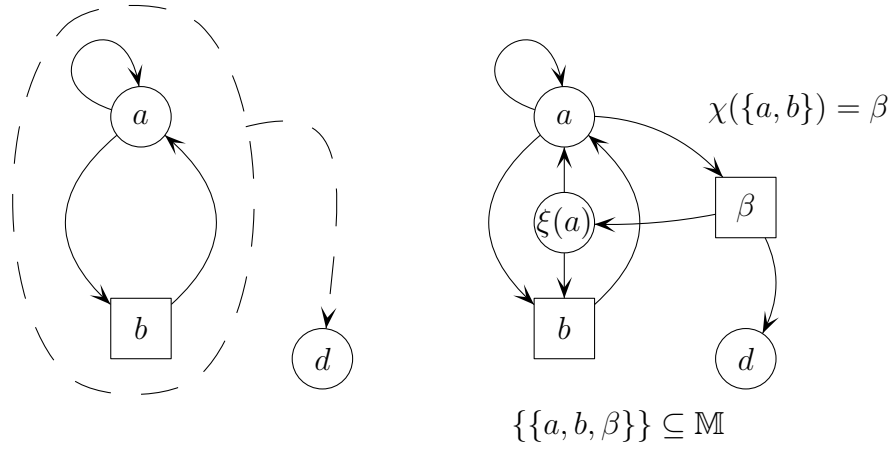


FIGURE 4.2 – Gadget pour la transition $\{a, b\} \xrightarrow{\text{lim}} d$

nouvelle stratégie a comme états de mémoire les parties de \mathbb{G} cohérentes avec la stratégie gagnante d'origine.

La mémoire évolue au cours d'une partie de G en parcourant l'arbre des parties de \mathbb{G} cohérentes avec la stratégie pour \mathbb{G} . Les transitions successeur augmentent la mémoire en suivant les arêtes de l'arbre. Les transitions limites permettent de changer de branche.

4.3.3.1 Ève

Étant donnée une stratégie σ pour Ève dans \mathbb{G} , supposée gagnante, on construit une stratégie ς pour Ève dans G , donc les états de mémoire sont des parties finies de \mathbb{G} . La Figure 4.3 donne un exemple de cette construction.

Lors d'une transition successeur, le sommet courant étant $q \in Q$ et la mémoire courante $\Omega \in \mathbb{Q}^*$, si $\sigma(\Omega \cdot q)$ est un sommet de Q alors ς suit σ . Si $\sigma(\Omega \cdot q)$ est $\chi(P)$, *i.e.* Ève propose un raccourci dans \mathbb{G} , alors ς simule le fait que Adam refuse ce raccourci, ajoute $q \cdot \chi(P) \cdot \xi(q)$ à sa mémoire, et joue $\sigma(\Omega \cdot q \cdot \chi(P)\xi(q))$, qui est nécessairement un sommet de Q , successeur de q dans G . C'est par exemple ce qui se passe sur la Figure 4.3 : $\sigma(a) = \beta$ donc ς joue vers $\sigma(a\beta\xi(a)) = b$.

Lorsque la longueur de la partie (dans G) atteint un ordinal limite j , la mémoire de ς tend vers une partie infinie de \mathbb{G} , que l'on note $\Omega_{<j}$. Cette

partie est cohérente avec σ , donc gagnante (Proposition 43). Afin de continuer le jeu la mémoire doit être réduite à une partie finie, et elle est tronquée à un préfixe correspondant à un raccourci proposé dans le passé, pour ensuite continuer le jeu dans une autre branche de l'arbre. Ce raccourci est choisi pour correspondre à l'ensemble cofinal de $\Omega_{<j}$ intersecté avec Q . Cet ensemble est égal à l'ensemble cofinal de la partie dans G , donc correspond à la transition limite prise dans G à la position j (Proposition 44). Dans l'exemple l'ensemble limite (intersecté avec Q) est $\{a, b\}$, qui mène au sommet d . Lors de la transition limite, la mémoire est tronquée à une visite précédente de $\beta = \chi(\{a, b\})$, et la partie continue après la transition limite vers $d = \lambda(\{a, b\})$.

On considère donc une partie ρ dans G cohérente avec ς , et une transition limite $P \xrightarrow{\text{lim}} q$ en position j (qui est un ordinal limite). On note $(\Omega_i)_{i<j}$ la suite transfinie d'états de mémoire de ς au cours de ρ , chaque Ω_i étant une partie finie dans \mathbb{G} cohérente avec σ .

Le premier problème est qu'il n'y a pas de « dernier » état de mémoire avant j . Les Propositions 43 et 44 résolvent cette difficulté en montrant que la séquence d'états de mémoire admet une limite.

Proposition 43. *La suite $(\Omega_i)_{i<j}$ admet une limite, notée $\Omega_{<j}$; cette limite est une partie infinie dans \mathbb{G} cohérente avec la stratégie gagnante d'origine.*

Proposition 44. $\lim \rho_{<j} = \text{Inf}(\Omega_{<j}) \cap Q$

Ces propositions permettent de mettre à jour la mémoire de la stratégie lors d'une transition limite. On coupe $\Omega_{<j}$ en trois facteurs v , w et ψ :

- $\Omega_{<j} = v \cdot w \cdot \psi$
- v est le plus court préfixe de $\Omega_{<j}$ qui contient toutes les occurrences des sommets de $Q \setminus \text{Inf}(\Omega_{<j})$
- w contient au moins une occurrence de chaque sommet de $\text{Inf}(\Omega_{<j}) \cap Q$
- w se termine à la première occurrence de $\chi(P)$ qui préserve la condition précédente

Le facteur $v \cdot w$ reste préfixe du nouvel état de mémoire, et ψ est remplacé par $t(P)$. Dans l'exemple de la Figure 4.3, on a donc $v = a \cdot \beta \cdot \xi(a) \cdot b \cdot d$, $w = a \cdot b \cdot a \cdot \beta$, et ψ contient une infinité de a , b et β .

Une fois la correction de cette construction démontrée, il est facile de montrer qu'elle aboutit à des stratégies gagnantes : une partie complète se termine toujours en \odot ou en \otimes , et le sommet courant est systématiquement

on ne fait que rallonger Ω_i , en restant cohérent avec σ , donc la propriété s'obtient facilement, comme dans le cas où $j = \omega$.

Si tout suffixe contient une transition limite, soit $P \rightarrow q$ une transition qui apparaît cofinalement et telle que P soit maximal. Si la transition $P \rightarrow q$ est utilisée à une position i , alors Ω_i a un préfixe de la forme $v \cdot w \cdot q$ où $\text{Occ}(w \cap Q) = P$. Nous allons montrer que ce préfixe est préservé à partir du point où aucune transition du type $P' \rightarrow q'$ avec $P \subsetneq P'$ n'intervient. La maximalité de P nous donne l'existence de ce point i et de la mémoire $v \cdot w \cdot \psi$ associée.

Si une transition $P' \rightarrow q'$ est effectuée à la position $k > i$, on sait que $P \cup \{q\} \not\subseteq P'$ par maximalité de P . La mise à jour temporaire de la mémoire à la position k donne le mot $\tilde{v} \cdot \tilde{w} \cdot q'$. On peut écrire w comme $x \cdot a \cdot y$, avec $a \notin P'$. Ce xa est donc un facteur de \tilde{v} . □

Démonstration de la Propriété 44. Par construction de Ω , tout sommet de Q qui apparaît dans $\Omega_{<j}$ est aussi dans $\rho_{<j}$. Si un sommet apparaît infiniment souvent dans $\Omega_{<j}$, il apparaît également dans $\lim \rho_{<j}$.

Si un sommet a est dans $\lim \rho_{<j}$ il apparaît forcément dans l'origine d'une transition $P \rightarrow q$ maximale, arbitrairement proche de j . D'après la preuve de la Propriété 43, a est donc ajouté un nombre infini de fois à la mémoire Ω , dans un préfixe qui ne sera pas effacé avant j . □

4.3.3.2 Adam

On veut maintenant construire, à partir d'une stratégie τ gagnante pour Adam dans \mathbb{G} , une nouvelle stratégie dans G . Pour Ève, un problème venait de l'existence dans \mathbb{G} d'arêtes entre un sommet de Q_E et un sommet qui n'existe pas dans G . Ce problème n'existe pas pour Adam, mais il ne suffit pas pour autant de toujours suivre la stratégie τ , il faut aussi interpréter les choix de son adversaire. On ne peut pas supposer, en mettant à jour la mémoire, qu'Ève restera toujours dans les sommets de Q , donc on va considérer qu'elle propose toujours un raccourci si τ dit à Adam de le refuser. Si on se trouve dans le sommet $q \in Q_E$ et qu'Ève joue vers q' (dans G), on commence par chercher un ensemble P tel que $o(P) = q$, et $\tau(\Omega \cdot q \cdot \chi(P)) = \xi(q)$. S'il n'existe pas de tel ensemble, alors on met à jour la mémoire en $\Omega \cdot q$. Sinon, on considère l'ensemble P vérifiant ces conditions tel que $\chi(P)$ apparaît le moins récemment dans Ω , et on met à jour la mémoire en $\Omega \cdot q \cdot \chi(P) \cdot \xi(q)$,

Ces propositions permettent de mettre à jour la mémoire de la stratégie lors d'une transition limite. On coupe $\Omega_{<j}$ en trois facteurs v , w et ψ :

- $\Omega_{<j} = v \cdot w \cdot \psi$;
- v est le plus court préfixe de $\Omega_{<j}$ qui contient toutes les occurrences des sommets de $Q \setminus \text{Inf}(\Omega_{<j})$;
- w contient une occurrence de chaque sommet de $\text{Inf}(\Omega_{<j}) \cap Q$;
- w se termine en $o(P)$, et $\tau(v \cdot w \cdot \chi(P)) = t(P)$, et est le mot le plus court qui vérifie ces conditions.

Après la transition limite, le nouvel état de mémoire est $v \cdot w \cdot \chi(P) \cdot t(P)$. Dans l'exemple de la Figure 4.4, v est d , w est $\delta \cdot \chi(d) \cdot a \cdot a \cdot b \cdot a$, et ψ contient une infinité de a et b . Après la transition limite, la partie de G est $d \cdot a \cdot b \cdot a \cdot b \cdot a \cdot b \cdot a \cdots d$ et la mémoire est $v \cdot w \cdot \beta \cdot d$.

Démonstration de la Propriété 45. On procède par induction sur j . Pour $j = \omega$, ou plus généralement si j peut s'écrire sous la forme $\alpha + \omega$, alors la propriété est triviale.

Considérons la suite de transitions limites dans $\rho_{<j}$. Soit $P \rightarrow q$ l'une de ces transitions qui apparaît arbitrairement proche de j , et telle que P soit maximal. Soit i une position où cette transition est prise, Ω_i a un préfixe de la forme $v \cdot w \cdot q$ où $\text{Occ}(w \cap Q) = P$. Nous allons montrer que ce préfixe est préservé si aucune transition $P' \rightarrow q'$ avec $P \subsetneq P'$ n'intervient après i . L'existence de ce point est garantie par la maximalité de P , et on y parvient avec une mémoire $\Omega_{<i} = v \cdot w \cdot \psi$ par induction.

Si une transition $P' \rightarrow q'$ est effectuée à la position $k > i$, on sait que $P \cup \{q\} \not\subseteq P'$ par maximalité de P . La mise à jour temporaire de la mémoire à la position k donne un mot $\tilde{v} \cdot \tilde{w} \cdot q'$. On peut écrire $w = x \cdot a \cdot y$ avec $a \notin P'$, et on a donc $x \cdot a$ facteur de \tilde{v} . \square

Démonstration de la Propriété 46. Par construction de Ω , tout sommet de Q apparaissant dans $\Omega_{<j}$ est aussi dans $\rho_{<j}$. Si un sommet apparaît infiniment souvent dans $\Omega_{<j}$ il apparaît également dans $\lim \rho_{<j}$.

Si un sommet a est dans $\lim \rho_{<j}$ il apparaît forcément dans l'origine d'une transition $P \rightarrow q$ maximale arbitrairement proche de j . D'après la preuve de la Propriété 45, a est donc ajouté un nombre infini de fois à la mémoire Ω dans un préfixe qui ne sera pas effacé avant j . \square

4.3.4 Complexité

Afin d'étudier la complexité de la résolution des jeux d'accessibilité ordinaux, il est nécessaire de préciser la représentation des entrées, et en particulier des transitions limites.

Soit une arène $\mathcal{A} = (Q, Q_A, Q_E, \mathcal{E}, t)$. La représentation de t peut être explicite, ce qui lui donne une taille exponentielle. Pour la rendre plus succincte, on peut se donner un ensemble de sommets $R \subseteq Q$ (sommets pertinents), et représenter t comme une fonction $t' : \mathcal{P}(R) \rightarrow Q$ avec $t(P) = t'(P \cap R)$.

Une autre possibilité est de définir une fonction de coloriage $\kappa : \rightarrow \{0, \dots, k\}$ et une fonction de transition $t' : \mathcal{P}(\{0, \dots, k\}) \rightarrow Q$ avec $t(P) = t'(\kappa(P))$.

On peut également mentionner la représentation sous forme d'arbre de Zielonka, ou du DAG associé. On représente les transitions limites par un arbre (DAG) étiqueté par un couple dans $\mathcal{P}(Q) \times Q$, avec à la racine $(Q, t(Q))$. Les fils de la racine sont étiquetés par les sous-ensembles maximaux de Q dont l'image par t n'est pas $t(Q)$, et par leur image. Les sous-arbres sont les arbres de Zielonka des sous-jeux associés. Le DAG est obtenu en identifiant les sommets de même étiquette.

Enfin on peut exprimer t comme une formule Booléenne sur les sommets (Emerson-Lei).

Ces différentes représentations ont été étudiées par Hunter et Dawar pour les jeux de Muller dans [HD05]. En particulier, pour la plupart d'entre elles trouver le vainqueur est PSPACE-complet. Il est polynomial si les transitions sont représentées explicitement [Hor08].

Théorème 47. *Trouver le vainqueur dans un jeu d'accessibilité ordinal est PSPACE-complet, si les transitions limites sont représentées comme ensembles pertinents, ensembles de couleurs, DAG de Zielonka ou par des formules Booléennes.*

4.3.4.1 Borne supérieure : réduction aux jeux d'Emerson-Lei

On donne ici une borne supérieure à la complexité de la résolution des jeux d'accessibilité ordinaux.

Lemme 48. *Décider le vainqueur d'un jeu ordinal d'accessibilité dont les transitions sont représentées par des formules Booléennes est PSPACE.*

Proposition 49. *Le jeu réduit \mathbb{G} est équivalent à un jeu d'Emerson-Lei \mathbb{L} de taille polynomiale en la taille de G si les transitions de G sont représentées par des formules Booléennes.*

Démonstration. Afin d'obtenir une réduction polynomiale il faut éviter d'avoir un sommet $\chi(P)$ pour chaque ensemble P de sommets. Pour cela, on identifie $\chi(P)$ et $\chi(P')$ si $o(P) = o(P')$ et $t(P) = t(P')$ (ils ont exactement les mêmes successeurs dans \mathbb{G}), et on note ce sommet $\iota(o(P), t(P))$. Cela limite le nombre de nouveaux sommets à $|Q|^2 + |Q|$.

La condition de victoire de \mathbb{L} peut être écrite comme suit, en notant ϕ_q la formule qui exprime qu'une transition limite mène vers q :

$$\phi = \odot \vee \bigvee_{q \in Q \cup \{\odot, \otimes\}} \left(\phi_q \wedge \bigvee_{p \in Q} \iota(p, q) \right)$$

La taille de ϕ est $O(\sum_{q \in Q \cup \{\odot, \otimes\}} (|\phi_q| + n))$, et donc polynomiale en $|G|$. \square

L'étape suivante est le Théorème 50 :

Théorème 50 ([HD05]). *Décider du vainqueur dans un jeu d'Emerson-Lei est PSPACE-complet.*

Le Lemme 48 découle directement de la Propriété 49 et du Théorème 50. Le Corollaire 51 provient des résultats de [HD05].

Corollaire 51. *Décider le vainqueur d'un jeu ordinal d'accessibilité dont les transitions limites sont représentées par des ensembles pertinents, des ensembles de couleurs, ou un DAG de Zielonka est PSPACE.*

4.3.4.2 Borne inférieure

La borne inférieure découle également du Théorème 50. En effet tout jeu de Muller peut être représenté comme un jeu ordinal d'accessibilité où la fonction de transition t associe \odot à tout ensemble gagnant du jeu de Muller, et \otimes à tout ensemble perdant. Les stratégies et les parties sont les mêmes dans les deux jeux, la seule différence étant une transition limite vers \otimes ou \odot après ω coups.

Lemme 52. *Dans un jeu ordinal d'accessibilité dont les transitions limites sont représentées par des ensembles pertinents, décider du vainqueur est PSPACE-difficile.*

4.3.5 Conclusion

L'algorithme présenté dans cette section fournit une borne PSPACE pour déterminer le vainqueur d'un jeu ordinal d'accessibilité, soit la même borne que pour les jeux infinis de Muller. En revanche il ne fournit pas d'indications sur les stratégies gagnantes. La section suivante comble ce manque via une autre approche.

4.4 Stratégies et mémoire

Dans la section précédente on a obtenu la détermination pour une sous-classe de jeux, mais pas de borne sur la mémoire nécessaire pour les stratégies gagnantes. On propose ici une seconde approche qui nous permettra d'obtenir d'une part la détermination dans le cas général, et d'autre part des stratégies à mémoire finie. On introduit pour cela un nouveau type de transitions limites.

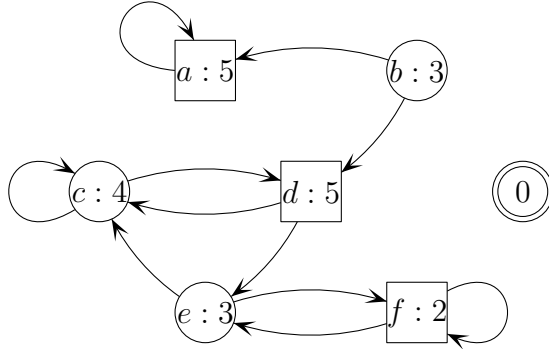
Au lieu de considérer tous les sommets répétés cofinalement lors d'une transition limite, on peut associer une couleur (ou priorité) à chaque sommet, et définir la fonction de transition à partir de la plus petite couleur répétée cofinalement. Ce modèle est analogue à celui des jeux de parité dans le cadre des jeux infinis, et l'algorithme qui suit est similaire à l'algorithme de Zielonka pour les jeux de parité.

4.4.1 Priorités

Un jeu ordinal à d couleurs est donné par une arène $(Q, Q_A, Q_E, \chi, \delta)$ où $\chi : Q \rightarrow \{0, \dots, d-1\}$ est une fonction de coloriage, et $\delta : \{0, \dots, d-1\} \rightarrow Q$ est une fonction de transition. La fonction de transition dans le sens de la Section 4.2 est $\lambda : P \mapsto \delta(\min\{\chi(q) \mid q \in P\})$.

La Figure 4.5 donne un exemple de jeu à priorités avec $d = 6$. Dans ce jeu, Adam gagne depuis les sommets c et d : il peut atteindre c depuis d , et chaque transition limite va ensuite renvoyer le jeton en d . Ève gagne depuis tous les autres sommets : depuis b elle peut atteindre a , la transition limite suivante la renvoie en b , et enfin elle atteint la cible après avoir joué $(ba^\omega)^\omega$; depuis e elle rejoint f , et une transition limite envoie ensuite vers a .

Tous les jeux ordinaux ne peuvent pas être représentés dans la même arène avec des transitions par priorité. Par exemple dans le jeu de la Figure 4.1 la transition depuis l'ensemble $\{a, b, c, d\}$ ne mène ni vers $t(\{a, b, c\})$



$$\delta(2) = a \quad \delta(3) = \odot \quad \delta(4) = d \quad \delta(5) = b$$

FIGURE 4.5 – Jeu à priorités

ni vers $t(\{a, b, d\})$. Ceci ne peut se produire avec des priorités, puisque $\min \chi\{a, b, c, d\}$ est soit $\min \chi\{a, b, c\}$ soit $\min \chi\{a, b, d\}$.

4.4.2 Résolution des jeux ordinaux à priorités

L'objet de cette section est d'établir le résultat suivant :

Théorème 53 ([CH08a]). *Dans un jeu ordinal à priorités, le vainqueur peut gagner sans mémoire.*

On suppose sans perte de généralité que \odot est le seul sommet de priorité 0, et qu'aucun sommet n'a priorité 1. On se sert de la priorité 1 comme un sommet « puits » permettant de terminer le calcul. L'algorithme donné Figure 4.6 calcule de façon incrémentale les régions gagnantes des deux joueurs. Chaque itération consiste en des calculs d'attracteurs.

L'algorithme de Zielonka pour résoudre les jeux de parité procède de la façon suivante. On pose $X = \text{Attr}_E^G(\chi^{-1}(0))$. L'ensemble $Q \setminus X$ est un piège pour Ève, et le jeu restreint à $Q \setminus X$ est une sous-arène à $d - 1$ couleurs, et peut donc être résolu par induction. La région gagnante pour Adam dans ce sous-jeu reste gagnante dans G , et son attracteur pour Adam l'est aussi. Si elle est vide, alors tout le reste du jeu est gagnant pour Ève : soit le jeu reste infiniment dans $Q \setminus X$ (où Ève a une stratégie gagnante), soit $\chi^{-1}(0)$ est visité infiniment souvent. Sinon, on peut alors recommencer la procédure sur

un jeu strictement plus petit, et ainsi construire progressivement les régions gagnantes des deux joueurs.

Pour résoudre un jeu ordinal, on va de même calculer une série d'attracteurs, et résoudre le jeu réduit au complémentaire. Cependant on ne peut pas s'en tenir là car d'une part la partie peut continuer après ω tours, et d'autre part on ne sait pas à l'avance quel joueur va vouloir répéter telle ou telle couleur.

La grande différence entre l'algorithme de Zielonka et l'algorithme pour les jeux ordinaux de priorité est donc que le joueur qui cherche à atteindre chaque couleur n'est pas connu à l'avance. On utilise deux tableaux \mathbf{v} et To indexés par les couleurs pour stocker cette information. À chaque étape, $\mathbf{v}[j]$ est le joueur censé chercher à atteindre j . Cela peut changer à chaque fois que l'attracteur vers j est calculé ; $\mathbf{v}[j]$ est Ève si et seulement s'il existe une couleur $i < j$ telle que $\mathbf{v}[i] = \text{Ève}$ et $\delta(j) \in \text{To}[i]$. Pour chaque couleur j , $\text{To}[j]$ contient les sommets où le joueur $\mathbf{v}[j]$ peut assurer une certaine propriété, précisée plus loin.

L'algorithme calcule une série d'attracteurs, à partir de la plus petite couleur (*i.e.* la plus importante). Après avoir calculé un attracteur $\text{To}[i]$, les sommets correspondants sont supprimés du graphe, et on recommence avec la couleur suivante $i + 1$. Lorsqu'un attracteur $\text{To}[k]$ contient les derniers sommets, on le regroupe avec l'un des attracteurs calculés précédemment. On ajoute donc $\text{To}[k]$ à l'ensemble $\text{To}[j]$ tel que $j < k$ soit le plus grand indice tel que $\mathbf{v}[j] = \mathbf{v}[k]$. On recalcule ensuite les attracteurs vers ce nouveau $\text{To}[j]$ et vers les couleurs supérieures à j . L'algorithme s'arrête lorsque chaque sommet est dans $\text{To}[0]$ ou $\text{To}[1]$, qui sont alors les régions gagnantes respectivement pour Ève et pour Adam. L'algorithme est donné en pseudo-code Figure 4.6.

La terminaison est assurée par le fait qu'à chaque itération, les ensembles $\text{To}[0]$ à $\text{To}[j - 1]$ sont inchangés et $\text{To}[j]$ grandit, assurant une croissance stricte dans l'ordre lexicographique du tuple $(\text{To}[i])_{0 \leq i < d}$.

4.4.2.1 Correction de l'algorithme

Dans la preuve de correction, on utilise la notation H^j pour $Q \setminus \cup_{k < j} \text{To}[k]$, et les propriétés suivantes obtenues par construction de To .

Proposition 54. *H^j est un sous-jeu.*

Proposition 55. *Pour toute couleur j , l'ensemble $\text{To}[j]$ est son propre attracteur pour le joueur $\mathbf{v}[j]$ dans l'arène H^j . Son complémentaire est donc*

un piège pour $\mathbf{v}[j]$.

On prouve la correction séparément pour les deux joueurs. Les arguments nécessaires, bien que similaires, ne sont pas tout à fait symétriques (Ève a un objectif d'accessibilité, Adam un objectif de sûreté). Dans les deux cas, on définit un prédicat sur les parties, et une propriété basée sur ce prédicat, puis on montre que la propriété reste vraie au cours de l'exécution de l'algorithme. Pour Ève, l'invariant de boucle \mathcal{I} et les prédicats \mathcal{M}^j sur les parties sont utilisées. Dans le cas d'Adam on utilise \mathcal{J} et \mathcal{N}^j .

Dans le cas d'Ève, il faut assurer que l'état cible est effectivement atteint. Pour le prédicat \mathcal{M}^j cela veut dire s'assurer qu'un état de couleur j (ou inférieure) est atteint, ce qui est similaire à un « until » fort.

Définition 56. Le prédicat $\mathcal{M}^j(\rho)$ est défini comme : $\mathbf{v}[j] = \text{Ève}$ et $\exists \alpha < \omega^{d-j}$ tel que $\text{Occ}(\rho_{<\alpha}) \subseteq \text{To}[j]$ et :

- $|\rho| = \alpha$, ou
- $\rho_\alpha \in \text{To}[j] \cap \chi^{-1}(j)$, ou
- $\exists k < j$ tel que $\mathcal{M}^k(\rho_{\geq\alpha})$ est vrai.

Définition 57. \mathcal{I} est la propriété « Ève a une stratégie positionnelle σ telle que pour tout j et toute partie ρ démarrant dans $\text{To}[j]$, si $\mathbf{v}[j] = \text{Ève}$ et ρ est cohérente avec σ alors $\mathcal{M}^j(\rho)$ est vrai ».

Cette propriété est vraie au début de l'exécution de l'algorithme : avant la ligne 7 $\text{To}[0]$ est le seul ensemble non-vide, et ne contient que $\text{Attr}_E(\chi^{-1}(0))$. Les Propositions 58 et 59 garantissent qu'elle reste vraie par la suite.

Proposition 58. Supposons que \mathcal{I} soit vraie à la ligne 9. Alors elle l'est toujours à la prochaine exécution de la ligne 15.

Démonstration. On note i la valeur de la variable i avant l'incrémentaion en ligne 9.

Les valeurs de To et \mathbf{v} sont les mêmes avant et après l'itération, sauf $\text{To}[i+1]$ et $\mathbf{v}[i+1]$. $\text{To}[i+1]$ est vide avant la boucle, et $\mathbf{v}[i+1]$ est Adam. Lorsqu'on écrit $\text{To}[i+1]$ ou $\mathbf{v}[i+1]$, on se réfère donc à leurs valeurs après la boucle (ligne 15).

Comme \mathcal{I} est vraie au début de la boucle, Ève a une stratégie positionnelle σ qui fonctionne jusqu'à i .

On définit une stratégie positionnelle ς :

- si $\mathbf{v}[i+1] = \text{\`Eve}$, $\forall q \in \text{To}[i+1]$, $\varsigma(q)$ est la stratégie attracteur vers $\chi^{-1}(i+1)$ dans H^{i+1}
- dans tous les autres sommets ς coïncide avec σ .

Toute partie cohérente avec σ et débutant en $q \in \text{To}[j]$ avec $\mathbf{v}[j] = \text{\`Eve}$ et $j \leq i$ vérifie \mathcal{M}^j donc son plus long préfixe entièrement contenu dans $\cup_{k \leq i} \text{To}[k]$ vérifie encore \mathcal{M}^j . Par définition de ς cette propriété reste vraie.

Si $\mathbf{v}[i+1] = \text{Adam}$, \mathcal{I} n'impose aucune contrainte supplémentaire et est donc toujours vraie.

Supposons maintenant que $\mathbf{v}[i+1] = \text{\`Eve}$ et considérons une partie cohérente avec ς et débutant dans $\text{To}[i+1]$. Par définition de $\text{To}[i+1]$ comme attracteur et de $\varsigma_{|\text{To}[i+1]}$ comme une stratégie correspondante, toute partie restant dans $\text{To}[i+1]$ visite $\chi^{-1}(i+1)$ après un nombre fini de coups, et vérifie donc \mathcal{M}^{i+1} .

Si le jeton quitte $\text{To}[i+1]$ avant d'atteindre un sommet de couleur $i+1$, il le fait lors d'un coup d'Adam, et par la Proposition 55 il atteint un $\text{To}[j]$ avec $j \leq i$ et $\mathbf{v}[j] = \text{\`Eve}$. On a $\text{Occ}(\rho_{<n}) \subseteq \text{To}[i+1]$, et par définition de ς , $\rho_{\geq n}$ est cohérente avec σ donc $\rho_{>n}$ vérifie \mathcal{M}^j , et donc ρ vérifie \mathcal{M}^{i+1} . \square

Proposition 59. *Supposons que \mathcal{I} soit vraie à la ligne 18. Alors elle l'est toujours à la prochaine exécution de la ligne 25.*

Démonstration. On note i la valeur de la variable \mathbf{i} au début de la boucle (avant la ligne 18), et i' sa valeur après la fin de la boucle.

Les valeurs de $\text{To}[j]$ et $\mathbf{v}[j]$ ne changent pas pour $j < i'$. Pour $j > i'$, $\text{To}[j]$ est vide après la boucle, et $\mathbf{v}[j] = \text{Adam}$, donc on se référera à leurs valeurs avant la boucle.

Soit σ une stratégie réalisant \mathcal{I} avant la boucle. On définit une nouvelle stratégie positionnelle ς :

- si $\mathbf{v}[i] = \text{Adam}$ alors ς coïncide avec σ ;
- si $\mathbf{v}[i] = \text{\`Eve}$ alors dans $\text{Attr}_E^{H^{i'+1}}(\text{To}[i]) \setminus \text{To}[i]$, ς est une stratégie attracteur d'Ève vers $\text{To}[i]$ dans $H^{i'+1}$
- dans le reste du jeu ς coïncide avec σ .

Soit ρ une partie démarrant dans $\text{To}[i]$ et cohérente avec ς . Soit $\alpha = \min(\{\beta \mid \rho_\beta \notin \text{To}[i]\} \cup \{|\rho|\})$.

On note (x_β) la suite de positions avant α telles que $\chi(\rho_{x_\beta}) = i$.

Supposons d'abord que (x_β) est une suite infinie. Soit γ le plus petit ordinal plus grand que tous les $(x_n)_{n < \omega}$. On a $\min \chi(\lim \rho_{<\gamma}) = i$ et donc $\rho_\gamma = \delta(i)$. De plus comme $\delta(i) \in \text{To}[j]$ pour un $j < i$ avec $\mathbf{v}[j] = \text{\`Eve}$, $\rho_{\geq \gamma}$

vérifie \mathcal{M}^j . Pour tout n on a $\mathcal{M}^i(\rho_{>x_n})$ donc $x_{n+1} = x_n + \epsilon$ où $\epsilon < \omega^{d-i}$. Par conséquent $\gamma \leq \omega^{d-i} < \omega^{d-i'}$ et donc après la boucle $\mathcal{M}^{i'}(\rho)$ est vraie.

Si par contre la suite (x_β) est finie, soit γ le plus petit ordinal plus grand que tous les x_n . Comme ci-dessus $\mathcal{M}^i(\rho_{>x_n})$ pour tout n ce qui donne à la fois $\gamma < \omega^{d-i}$ et $\alpha = \gamma + \epsilon$ avec $\epsilon < \omega^{d-i}$. De plus $\rho_{\geq\alpha}$ doit vérifier \mathcal{M}^j pour un certain $j \leq i'$. On en déduit $\text{Occ}(\rho_{<\alpha}) \subseteq \text{To}[i]$ avec $\alpha < \omega^{d-i} < \omega^{d-i'}$ et $\mathcal{M}^j(\rho_{>\alpha})$, et donc $\mathcal{M}^{i'}(\rho)$ est vraie après la boucle. \square

La structure de la preuve pour les états d'Adam est similaire, avec des prédicats plus faibles.

Définition 60. Le prédicat $\mathcal{N}^j(\rho)$ est défini comme $\mathbf{v}[j] = \text{Adam}$ et $\exists\alpha$ tel que $\text{Occ}(\rho_{<\alpha}) \subseteq \text{To}[j]$ et :

- $|\rho| = \alpha$ ou
- $\rho_\alpha \in \text{To}[j] \cap \chi^{-1}(j)$ ou
- $\exists k < j$ tel que $\mathcal{N}^k(\rho_{\geq\alpha})$ est vrai.

Définition 61. \mathcal{J} est la propriété « Adam a une stratégie positionnelle τ telle que pour toute couleur j et toute partie ρ démarrant dans $\text{To}[j]$, si $\mathbf{v}[j] = \text{Adam}$ et ρ est cohérente avec τ , alors $\mathcal{N}^j(\rho)$ est vrai ».

Proposition 62. Supposons que \mathcal{J} soit vraie à la ligne 9. Alors elle l'est toujours à l'exécution suivante de la ligne 15.

Démonstration. On note i la valeur de la variable i avant l'incrémentacion en ligne 9.

Comme dans la preuve de la Proposition 58 on utilise les valeurs de To et \mathbf{v} lorsque la ligne 15 est atteinte.

Comme \mathcal{J} est vraie au début de la boucle, on a une stratégie positionnelle τ correspondante. On définit une nouvelle stratégie positionnelle θ :

- si $\mathbf{v}[i+1] = \text{Adam}$, $\forall q \in \text{To}[i+1]$, $\theta(q)$ est une stratégie attracteur vers $\chi^{-1}(i+1)$ dans H^{i+1}
- ailleurs θ coïncide avec τ .

Toute partie cohérente avec τ et démarrant dans $\text{To}[j]$ avec $\mathbf{v}[j] = \text{Adam}$ et $j \leq i$ vérifie \mathcal{N}^j donc son plus long préfixe entièrement contenu dans $\cup_{k \leq i} \text{To}[k]$ le vérifie également. Par définition de θ tout fonctionne bien jusqu'à i .

Si $\mathbf{v}[i+1] = \text{Ève}$, \mathcal{J} n'impose pas de contrainte supplémentaire donc elle est toujours vérifiée.

Si maintenant $v[i+1] = \text{Adam}$, considérons une partie ρ cohérente avec θ et démarrante en $\text{To}[i+1]$. Comme $\text{To}[i+1]$ est un attracteur et θ une stratégie correspondante, toute partie restant dans $\text{To}[i+1]$ visite $\chi^{-1}(i+1)$ après un nombre fini de coups, et vérifie donc \mathcal{N}^{i+1} . Si la partie quitte $\text{To}[i+1]$ avant de visiter $\chi^{-1}(i+1)$ elle ne peut le faire que par une action d'Ève, et par la Proposition 55 la destination est dans un $\text{To}[j]$ avec $j \leq i$ et $v[j] = \text{Adam}$. On a $\text{Occ}(\rho_{<n}) \subseteq \text{To}[i+1]$, et par définition de θ , $\rho_{\geq n}$ est cohérente avec τ donc $\rho_{\geq n}$ vérifie \mathcal{N}^j , d'où on déduit que ρ vérifie \mathcal{N}^{i+1} . \square

Proposition 63. *Supposons que \mathcal{J} soit vraie avant la ligne 18. Alors elle l'est toujours à l'exécution suivante de la ligne 25.*

Démonstration. On note i la valeur de la variable i au début de la boucle (avant la ligne 18), et i' sa valeur après la fin de la boucle. Comme dans la preuve de la Proposition 59, on considère les valeurs de To et v avant la boucle.

Soit τ une stratégie réalisant \mathcal{J} avant la boucle. On définit une nouvelle stratégie positionnelle θ par :

- si $v[i] = \text{Adam}$, alors dans $\text{Attr}_A^{H^{i'+1}}(\text{To}[i]) \setminus \text{To}[i]$, θ est une stratégie attracteur d'Adam vers $\text{To}[i]$ dans le jeu $H^{i'+1}$
- ailleurs θ coïncide avec τ .

Si $v[i] = \text{Ève}$, rien n'a changé pour Adam donc \mathcal{J} reste vraie. On suppose donc que $v[i] = \text{Adam}$. Soit ρ une partie cohérente avec θ débutant dans $\text{To}[i]$. On pose $\alpha = \min(\{\beta \mid \rho_\beta \notin \text{To}[i]\} \cup \{|\rho|\})$.

Si $\alpha = |\rho|$ alors $\mathcal{M}^{i'}(\rho)$ est vraie. Sinon, soit (x_β) la suite de positions avant α telles que $\chi(\rho_{x_\beta}) = i$. Si cette suite a un dernier élément γ on a $\mathcal{N}^i(\rho_{>\gamma})$ avant la boucle et $\rho_{>\gamma}$ quitte $\text{To}[i]$ avant d'atteindre $\chi^{-1}(i)$ donc $\exists j < i$ tel que $\mathcal{N}^j(\rho_{\geq\alpha})$. Après la boucle $\rho_{<\alpha}$ reste donc tout le temps dans $\text{To}[i']$ et $\rho_{\geq\alpha}$ vérifie \mathcal{N}^j pour un $j < i'$ donc ρ vérifie $\mathcal{N}^{i'}$.

Si (x_β) n'a pas de dernier élément, on note γ le premier ordinal plus grand que tous les x_β . On a $\rho_\gamma = \delta(i)$. Si $\gamma < \alpha$ alors comme ci-dessus $\rho_{\geq\gamma}$ vérifie \mathcal{N}^i avant la boucle donc $\mathcal{N}^{i'}(\rho)$ est vraie après la boucle. Si $\gamma = \alpha$ alors $j = \text{To}^{-1}[\delta(i)] < i$ donc $\rho_{\geq\gamma}$ vérifie \mathcal{N}^j et comme $j \leq i'$ on a encore $\mathcal{N}^{i'}(\rho)$ après la boucle. \square

4.4.2.2 Conséquences

L'algorithme de la Figure 4.6 permet non seulement de résoudre les jeux ordinaux de priorité, mais il fournit aussi des stratégies positionnelles pour

les joueurs. Cela fournit donc une nouvelle classe de jeux positionnels, qui correspond aux conditions de victoire d'accessibilité (pour Ève) et de sûreté (pour Adam). Par ailleurs l'algorithme fournit le corollaire suivant :

Corollaire 64. *Si Ève gagne un jeu d'accessibilité à priorités, alors elle peut s'assurer qu'une partie sera toujours plus courte que ω^ω .*

4.4.3 LAR ordinaux

Les résultats de la section 4.4.2 peuvent être étendus à tous les jeux ordinaux d'accessibilité. On peut en effet réduire un jeu ordinal à un jeu de priorité qui lui est bisimilaire : les sommets équivalents appartiennent au même joueur, et deux parties équivalentes ont des transitions limites menant à des sommets équivalents.

Théorème 65. *Dans un jeu de longueur ordinaire, le vainqueur a une stratégie gagnante avec mémoire finie.*

Les LAR, ou *Latest Appearance Records*, ont été introduits par Gurevich et Harrington [GH82] pour prouver l'existence de stratégies à mémoire finie dans les jeux infinis de Muller (*Forgetful Determinacy*). Un LAR dans un jeu G à n sommets est une paire (π, i) où π est une permutation des sommets de G , et $1 \leq i \leq n$.

On réduit un jeu ordinal $G = (Q, Q_E, Q_A, T, \lambda)$ à un jeu à priorités $\mathbb{G} = (Q, Q_E, Q_A, \mathbb{T}, \chi, \delta)$. Les sommets et les transitions successeurs sont définis comme dans la construction d'origine :

- $Q = \{(\pi, i) \mid \pi \in \mathcal{S}(Q), 1 \leq i \leq n\}$
- $Q_E = \{(\pi, i) \in Q \mid \pi(1) \in Q_E\}$
- $Q_A = \{(\pi, i) \in Q \mid \pi(1) \in Q_A\}$
- $(\pi, i) \xrightarrow{\mathbb{T}} (\pi', i')$ si :
 - $\pi(1) \xrightarrow{T} \pi'(1)$
 - $\forall q, r \in Q \setminus \{\pi'(1)\}, \pi^{-1}(q) < \pi^{-1}(r) \Leftrightarrow \pi'^{-1}(q) < \pi'^{-1}(r)$
 - $\pi(i') = \pi'(1)$

Les transitions limites demandent un peu plus de travail. Il nous faut conserver en mémoire certaines informations même après une transition limite, ce qui n'était pas le cas pour les jeux infinis ; on va donc utiliser une couleur (priorité) différente pour chaque LAR. Ces priorités doivent vérifier $i < i' \Rightarrow \chi(\pi, i) > \chi(\pi', i')$. Par abus de notation on peut alors écrire δ comme une fonction de Q dans Q , définie par :

- $(\cup_{j=1}^i \{\pi(j)\}) \xrightarrow{\lambda} \pi'(1)$
- $\forall q, r \in Q \setminus \{\pi'(1)\}, \pi^{-1}(q) < \pi^{-1}(r) \Leftrightarrow \pi'^{-1}(q) < \pi'^{-1}(r)$
- $\pi(i') = q'$

La Figure 4.7 montre une partie du jeu réduit correspondant au jeu de la figure 4.1. La bisimulation est montrée comme dans le cas infini, en ajoutant la gestion des transitions limites.

Lemme 66. *Il y a une bisimulation entre G et \mathbb{G} telle que deux parties partielles sans dernier sommet peuvent se prolonger par une transition limite vers des sommets bisimilaires.*

La bisimulation \sim est définie simplement : un sommet $q \in Q$ est bisimilaire aux LAR $(\pi, i) \in \mathbb{Q}$ tels que $\pi(q) = 1$. La définition de \mathbb{G} donne trivialement la correction de cette relation pour les transitions successeur. Encore une fois, la difficulté est de montrer que les transitions limites se comportent bien. On étend la relation \sim aux parties de G et \mathbb{G} en posant $\rho \sim \rho'$ si elles ont la même longueur et chaque sommet le long de ρ est bisimilaire au sommet correspondant de ρ' .

Remarquons pour commencer que la bisimulation ne respecte pas l'équivalence au niveau des ensembles : deux ensembles de sommets équivalents ne mènent pas à des sommets équivalents. Dans l'exemple de la Figure 4.7, l'ensemble $\{(abcd, 2)\}$, bisimilaire à a , donne une transition limite vers $(dabc, 4)$, bisimilaire à d , et non vers $c = \lambda(\{a\})$.

En revanche, si l'on considère les chemins menant à une transition limite, la bisimulation est conservée. Il n'y a pas de boucle sur le sommet $(abcd, 2)$ dans \mathbb{T} , alors qu'il y en avait une sur a dans T , donc l'exemple donné ci-dessus n'a pas de sens dans la réduction. La transition correspondant à cette boucle mène à $(abcd, 1)$, qui lui boucle bien sur lui-même dans \mathbb{G} , et on a bien $\delta(\{(abcd, 1)\}) = (cabd, 3) \sim c$. Afin de prouver la préservation de \sim par transition limite, on va donc procéder par induction transfinie sur la longueur des chemins. Les arguments utilisés sont classiques : chaque sommet n'apparaissant pas dans l'ensemble limite va finir par être relégué à la fin de la permutation, et l'entier du LAR va prendre comme valeur répétée maximale le cardinal de l'ensemble limite.

Avant d'entrer dans la preuve proprement dite, fixons quelques notations : ρ est une partie de G sans dernier sommet, et ρ' est un jeu de \mathbb{G} tel que $\rho \sim \rho'$. On considère également les suites transfinies π et i , définies comme projections de $\rho' : \rho'_\alpha = (\pi_\alpha, i_\alpha)$. La notation $\pi_\alpha(q)$ désigne la position de q dans la permutation π_α .

On note L l'ensemble limite de ρ et t le sommet $\lambda(L)$ (qui prolonge ρ après la transition limite). On introduit de même L' et $(p', \iota') = \delta(p, \iota)$, où (p, ι) est le sommet de L' de couleur minimale. Afin de prouver le Lemme 66, il faut donc montrer que $t \sim (p', \iota')$.

La première étape est de se débarrasser d'un préfixe des parties pendant lequel les sommets hors de L sont encore visités. On note α le premier ordinal tel que $\forall q \in L, \forall r \notin L, \exists \beta \leq \alpha, \rho_\beta = q \wedge \forall \gamma, \rho_\gamma = r \Rightarrow \gamma < \beta$. Ceci correspond à un préfixe qui contient toutes les occurrences de sommets hors de L , suivi d'au moins une occurrence de chaque élément de L . L'existence de cette position α est garantie : tout sommet hors de la limite L cesse d'apparaître à partir d'un certain point, et chaque sommet de L apparaît forcément après une position quelconque.

Les Propositions 67 et 68 utilisent l'existence de cette position α :

Proposition 67. *Pour tout ordinal $\gamma \geq \alpha, \pi_\gamma(L) = \{1, \dots, k\}$.*

Proposition 68. *Pour tout ordinal $\gamma > \alpha, i_\gamma \leq k$.*

Démonstration de la Proposition 67. Soient $q \in L$ et $r \notin L$. On pose définit un prédicat $\mathcal{R}_r^q(\gamma)$ comme « $\pi_\gamma(q) < \pi_\gamma(r)$ ». Par définition d' α , il y a un ordinal $\beta(q, r) \leq \alpha$ tel que $\beta(q, r)$ soit la première occurrence de q qui suive toutes les occurrences de r . Par bisimilarité de ρ et ρ' , on a $\pi_{\beta(q, r)}(q) = 1$, et donc $\mathcal{R}_r^q(\beta(q, r))$. Afin de prouver que \mathcal{R}_r^q reste vrai après $\beta(q, r)$, on procède par l'absurde. Supposons donc que γ soit le premier ordinal après $\beta(q, r)$ tel que $\mathcal{R}_r^q(\gamma)$ soit faux. On considère deux cas :

γ est un ordinal successeur ($\zeta + 1$) : Par définition de \mathbb{T} , $\pi_\zeta(q) < \pi_\zeta(r)$ et $\pi_\gamma(r) < \pi_\gamma(q)$ entraînent $\pi_\gamma(r) = 1$. Par bisimilarité on a également $\rho_\gamma = r$ ce qui contredit $\gamma > \beta(q, r)$.

γ est un ordinal limite : Il y a un sommet (m, z) dans la limite de $\rho'_{<\gamma}$ tel que $\rho'_\gamma = \delta(m, z)$. Soit ζ une position telle que $\beta(q, r) < \zeta < \gamma$ et $\rho'_\zeta = (m, z)$. Par définition de δ , $\pi_\zeta(q) < \pi_\zeta(r)$ et $\pi_\gamma(r) < \pi_\gamma(q)$ forcent $\pi_\gamma(r) = 1$. Encore une fois par bisimilarité on a $\rho_\gamma = r$, en contradiction avec la définition de γ .

On a donc bien $\mathcal{R}_r^q(\gamma)$ pour tout $q \in L$, tout $r \notin L$, et tout $\gamma > \beta(q, r)$. \square

Démonstration de la Proposition 68. Soit $\gamma > \alpha$. Ici encore on considère deux cas selon que γ est un ordinal successeur ou limite.

$\gamma = \zeta + 1$: Soit q tel que $\pi_\gamma(q) = 1$. La Proposition 67 appliquée en γ donne $q \in L$. Appliquée en ζ , elle donne $\pi_\zeta(q) \leq k$. Par définition de \mathbb{T} , $i_\gamma = \pi_\zeta(q)$ donc $i_\gamma \leq k$.

γ **est limite** : Il existe un sommet (m, z) dans la limite de $\rho'_{<\gamma}$ tel que $\rho'_\gamma = \delta(m, z)$. Soit ζ tel que $\alpha \leq \zeta < \gamma$ et $\rho'_\zeta = (m, z)$. Soit q tel que $\pi_\gamma(q) = 1$. La Proposition 67 en γ donne $q \in L$. En ζ elle donne $\pi_\zeta(q) \leq k$. Par définition de δ , $i_\gamma = \pi_\zeta(q)$, d'où $i_\gamma \leq k$.

On en déduit la Proposition 68. □

On remarque en particulier que les deux propositions ci-dessus sont vraies aux positions où ρ' est égal à (p, ι) . On a donc $p(L) = \{1, \dots, k\}$ et $\iota \leq k$. Il reste à montrer que ι est plus grand que k . Par la définition de la fonction de coloriage χ , ι doit être l'entier maximal dans la limite : $\iota = \max\{i \mid \exists \pi, (\pi, i) \in L'\}$. Comme \mathbb{Q} est fini, il suffit de montrer la Proposition 69.

Proposition 69. *Pour tout ordinal $\gamma \geq \alpha$ il existe un ordinal $\zeta > \gamma$ tel que $i_\zeta \geq k$.*

Démonstration. Soit γ un ordinal plus grand qu' α et q tel que $\pi_\gamma(q) = k$. Par la Proposition 67, on a $q \in L$ donc on peut prendre ζ le plus petit ordinal plus grand que γ tel que $\rho_\zeta = q$. Par une induction transfinie proche de celle utilisée pour prouver la Proposition 67, on peut montrer que n'importe quel ordinal η tel que $\gamma \leq \eta < \zeta$, $\pi_\eta(q) = k$. Par des arguments proches de ceux de la preuve de la Proposition 68, on peut montrer que $i_\zeta \geq k$. □

4.5 Conclusion

On a introduit dans ce chapitre deux algorithmes pour les jeux de longueur ordinaire sur un graphe fini. Le premier donne une borne PSPACE pour trouver le vainqueur (Théorème 47), par réduction aux jeux de Muller, dont on connaît la complexité [HD05]. En revanche il ne fournit que des stratégies à mémoire non bornée, donc peu intéressantes dans une optique de synthèse.

Le second algorithme utilise une technique similaire à l'algorithme de Zielonka pour les jeux de parité, et permet de résoudre des jeux de longueur ordinaire à priorités. Comme dans le cas des jeux de parité, il fournit une stratégie positionnelle au vainqueur (Théorème 53).

On peut alors adapter le LAR de Gurevich et Harrington pour résoudre les jeux de longueur ordinaire plus généraux. Cette réduction permet de montrer qu'une mémoire finie est suffisante pour gagner ces jeux (Théorème 65).

Entrées: Le jeu G
Sorties: Les régions gagnantes pour Ève et Adam

```

1  $v[0] \leftarrow E$ 
2  $To[0] \leftarrow \text{Attr}_E^G(\chi^{-1}(0))$ 
3 pour  $0 < i < d$  faire
4   |  $To[i] \leftarrow \emptyset$ 
5  $H \leftarrow G \setminus To[0]$ 
6  $i \leftarrow 0$ 
7 tant que  $(To[0] \cup To[1]) \neq G$  faire
8   | tant que  $(H \neq \emptyset)$  faire
9     |  $i \leftarrow i + 1$ 
10    | si  $\exists j \mid \delta(i) \in To[j]$  alors
11      |  $v[i] \leftarrow v[j]$ 
12    | sinon
13      |  $v[i] \leftarrow A$ 
14      |  $To[i] \leftarrow \text{Attr}_{v[i]}^H(\chi^{-1}(i))$ 
15      |  $H \leftarrow H \setminus To[i]$ 
16    |  $Tmpto \leftarrow To[i]$ 
17    |  $Tmpv \leftarrow v[i]$ 
18    | répéter
19      |  $H \leftarrow H \cup To[i]$ 
20      |  $To[i] \leftarrow \emptyset$ 
21      |  $v[i] \leftarrow A$ 
22      |  $i \leftarrow i - 1$ 
23    | jusqu'à  $(v[i] = Tmpv)$ 
24    |  $To[i] \leftarrow To[i] \cup \text{Attr}_{Tmpv}^H(Tmpto)$ 
25    |  $H \leftarrow H \setminus To[i]$ 
26 retourner  $(To[0], To[1])$ 

```

FIGURE 4.6 – Algorithme pour résoudre les jeux à priorités

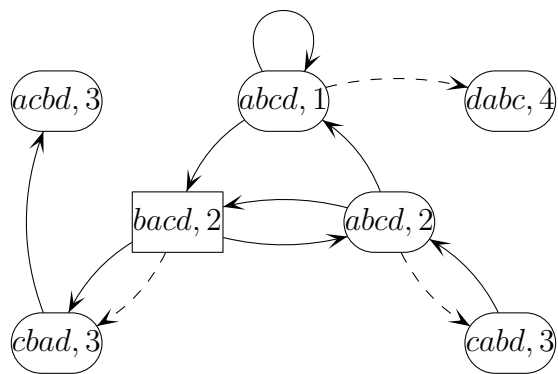


FIGURE 4.7 – Détail de la réduction LAR du jeu de la Figure 4.1 (les pointillés représentent les transitions limites)

Table des matières

1	Introduction	3
2	Généralités	7
2.1	Automates et logique	7
2.1.1	ω -automates	8
2.1.2	Logique	10
2.2	Jeux	11
2.3	Ordinaux et ordres	15
2.3.1	Ordres linéaires	15
3	Automates et logique temporelle	17
3.1	Définitions	18
3.1.1	Automates	18
3.1.2	Logique	24
3.2	État de l'art	27
3.3	Satisfaisabilité de LTL	29
3.3.1	Accessibilité	30
3.3.2	Transducteurs	35
3.3.3	Satisfaisabilité	37
3.4	Discussion	45
4	Jeux	49
4.1	Introduction	49
4.2	Définitions	50
4.3	Détermination	52
4.3.1	Restriction sur les arènes	53
4.3.2	Réduction aux jeux de Muller	53
4.3.3	Traduction de stratégies : de Muller aux ordinaux	54

4.3.4	Complexité	61
4.3.5	Conclusion	63
4.4	Stratégies et mémoire	63
4.4.1	Priorités	63
4.4.2	Résolution des jeux ordinaux à priorités	64
4.4.3	LAR ordinaux	70
4.5	Conclusion	73

Bibliographie

- [BC01] Véronique Bruyère and Olivier Carton. Automata on linear orderings. In Jiri Sgall, Ales Pultr, and Petr Kolman, editors, *MFCS*, volume 2136 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2001.
- [BC06] Alexis Bès and Olivier Carton. A Kleene theorem for languages of words indexed by linear orderings. *Int. J. Found. Comput. Sci.*, 17(3) :519–542, 2006.
- [BC07] Véronique Bruyère and Olivier Carton. Automata on linear orderings. *J. Comput. Syst. Sci.*, 73(1) :1–24, 2007.
- [Bed98] Nicolas Bedon. *Langages reconnaissables de mots indexés par des ordinaux*. PhD thesis, Université de Marne-la-Vallée, 1998.
- [BL69] J. Richard Büchi and Lawrence H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2) :166–170, 1969.
- [Bla92] Andreas Blass. A Game Semantics for Linear Logic. *Annals of Pure and Applied Logic*, 56(1–3) :183–220, 1992.
- [BS73] J. Richard Büchi and Dirk Siefkes. The monadic second order theory of all countable ordinals. volume 328 of *Lecture notes in mathematics*. Springer, 1973.
- [Bü62] J. Richard Büchi. On a decision method in restricted second order arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, June 1962.
- [Bü65] J. Richard Büchi. Transfinite automata recursions and weak second order theory of ordinals. In *Proc. Int. Congress Logic, Me-*

- thodology, and Philosophy of Science*, pages 2–23. North-Holland, 1965.
- [Car02] Olivier Carton. Accessibility in automata on scattered linear orderings. In Krzysztof Diks and Wojciech Rytter, editors, *MFCS*, volume 2420 of *Lecture Notes in Computer Science*, pages 155–164. Springer, 2002.
- [CdAH05] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. The complexity of stochastic rabin and streett games’. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 878–890. Springer, 2005.
- [CH08a] Julien Cristau and Florian Horn. Graph games on ordinals. In Hariharan et al. [HMV08], pages 143–154.
- [CH08b] Julien Cristau and Florian Horn. On reachability games of ordinal length. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM*, volume 4910 of *Lecture Notes in Computer Science*, pages 211–221. Springer, 2008.
- [Cho78] Yaacov Choueka. Finite automata, definable sets, and regular expressions over ω^n -tapes. *J. Comput. Syst. Sci.*, 17(1) :81–97, 1978.
- [Chr03] Juliusz Chroboczek. *Game Semantics and Subtyping*. PhD thesis, University of Edinburgh, 2003.
- [CJH03] Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Simple stochastic parity games. In Matthias Baaz and Johann A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 100–113. Springer, 2003.
- [Col10] Thomas Colcombet. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theor. Comput. Sci.*, 411(4-5) :751–764, 2010.
- [Cou38] A. Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses*. 1838.
- [Cri09] Julien Cristau. Automata and temporal logic over arbitrary linear time. In Ravi Kannan and K. Narayan Kumar, editors, *FSTTCS*, volume 4 of *LIPICs*, pages 133–144. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.

- [DR07] Stéphane Demri and Alexander Rabinovich. The complexity of temporal logic with until and since over ordinals. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2007.
- [EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 368–377, San Juan, Porto Rico, October 1991. IEEE Computer Society Press.
- [EVW02] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2) :279–295, 2002.
- [Far01] Berndt Farwer. ω -automata. In Grädel et al. [GTW02], pages 3–20.
- [GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65. ACM, 1982.
- [GHR94] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic, Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC’87*, pages 218–229. ACM Press, 1987.
- [GPSS80] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In *POPL*, pages 163–173, 1980.
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games : A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
- [HD05] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2005.

- [HMV08] Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, December 9-11, 2008, Bangalore, India*, volume 2 of *LIPICs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2008.
- [Hor08] Florian Horn. Explicit Muller Games are PTIME. In Hariharan et al. [HMV08], pages 235–243.
- [Kam68] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.
- [Kav86] Gregory S. Kavka. *Hobbesian moral and political theory*. Princeton University Press, 1986.
- [Kla01] Felix Klaedtke. Complementation of Büchi automata using alternation. In Grädel et al. [GTW02], pages 61–78.
- [Kle56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Shannon and McCarthy, editors, *Automata studies*, pages 3–42. Princeton University Press, 1956.
- [Mar75] Donald A. Martin. Borel determinacy. *Ann. Math.*, 102 :363–371, 1975.
- [McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5) :521–530, 1966.
- [Mos91] Andrzej Włodzimierz Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991.
- [Mul63] David E. Muller. Infinite sequences and finite machines. In *FOCS*, pages 3–16. IEEE, 1963.
- [Nie01] Frank Nießner. Nondeterministic tree automata. In Grädel et al. [GTW02], pages 135–152.
- [Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 1969.
- [Rab10] Alexander Rabinovich. Temporal logics over linear time domains are in PSPACE. Under submission, 2010.

- [RC05] Chloe Rispal and Olivier Carton. Complementation of rational sets on countable scattered linear orderings. *Int. J. Found. Comput. Sci.*, 16(4) :767–786, 2005.
- [Rey03] Mark Reynolds. The complexity of the temporal logic with "until" over general linear time. *J. Comput. Syst. Sci.*, 66(2) :393–426, 2003.
- [Rey10] M. Reynolds. The complexity of temporal logic over the reals. *Ann. Pure Appl. Logic*, 161(8) :1063–1096, 2010.
- [Ris04] Chloé Rispal. *Automates sur les ordres linéaires : complémentation*. PhD thesis, Université de Marne-la-Vallée, 2004.
- [RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, pages 114–125, 1959.
- [SC85] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3) :733–749, 1985.
- [She75] Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, 102 :379–419, 1975.
- [Smi82] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.
- [Sto74] Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, MIT, Cambridge, Massachusetts, USA, 1974.
- [STV02] Thomas Schwentick, Denis Thérien, and Heribert Vollmer. Partially-ordered two-way automata : A new characterization of DA. In *DLT '01 : Revised Papers from the 5th International Conference on Developments in Language Theory*, pages 239–250, London, UK, 2002. Springer-Verlag.
- [Tho97] Wolfgang Thomas. Languages, automata, and logic. In G. Rosenbergs and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.
- [Woj84] Jerzy Wojciechowski. Classes of transfinite sequences accepted by finite automata. *Fundamenta informaticæ*, 7(2) :191–223, 1984.
- [Woj85] Jerzy Wojciechowski. Finite automata on transfinite sequences and regular expressions. *Fundamenta informaticæ*, 8(3-4) :379–396, 1985.

- [Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2) :135–183, 1998.