Caractérisation et programmation en théorie des langages et en logique des classes de complexité efficace des automates cellulaires

Theo Grente

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Caractérisation et programmation en théorie des langages et en logique des classes de complexité efficace des automates cellulaires

Présentée et soutenue par
Theo GRENTE

Thèse soutenue publiquement le 08/12/2020 devant le jury composé de		
M. JULIEN CERVELLE	Professeur des universités, Université Paris-Est Créteil (UPEC)	Rapporteur du jury
M. ARNAUD DURAND	Professeur des universités, Université Paris 7 Paris Diderot	Rapporteur du jury
Mme NATACHA PORTIER	Maître de conférences, ENS Lyon Sciences	Membre du jury
Mme SOPHIE TISON	Professeur des universités, Université Lille 1 Sciences Et Technolog	Membre du jury
M. JEAN-BAPTISTE YUNES	Maître de conférences HDR, Université Paris 7 Paris Diderot	Membre du jury
M. ETIENNE GRANDJEAN	Professeur des universités, Université Caen Normandie	Directeur de thèse
Mme VERONIQUE TERRIER	Maître de conférences HDR, Université Caen Normandie	Co-directeur de thèse

Thèse dirigée par ETIENNE GRANDJEAN et VERONIQUE TERRIER, Groupe de recherche en informatique, image, automatique et instrumentation

UNIVERSITÉ CAEN NORMANDIE

GREYC

Table des matières

Introduction

Une *classe de complexité* est l'ensemble des problèmes dont la résolution sur une machine donnée, modélisée par un modèle de calcul, nécessite une quantité de ressources bornée par une fonction donnée. Les deux ressources les plus étudiées sont le *temps* et *l'espace*. Une classe de complexité est d'autant plus intéressante qu'elle respecte les deux critères suivants : elle contient plusieurs problèmes "naturels" et admet une caractérisation "naturelle" *indépendante des machines*. C'est dans le but d'obtenir une telle caractérisation que la *complexité descriptive* intervient. La *complexité descriptive* est l'emploi de la logique sur des structures finies pour caractériser une classe de complexité. Un des plus célèbres exemples de caractérisation d'une classe de complexité par la logique est le théorème de Fagin [13, 31] qui caractérise NP comme la classe des problèmes définissables en *logique existentielle du second-ordre* (ESO). De la même façon, le théorème d'Immerman-Vardi [26, 31] et le théorème de Grädel [15, 16] caractérisent la classe P comme la classe des problèmes définissables, respectivement, par la *logique du premier-ordre plus point-fixe minimal*, et par la *logique du second-ordre restreinte aux formules de Horn* (ESO-HORN).

Un autre intérêt de la complexité descriptive est de permettre de déduire *automatiquement* d'une description logique d'un problème, un programme le résolvant. Ce qui s'avère particulièrement intéressant dans la conception de *programmes parallèles*, considérée comme une tâche difficile [30, 40].

Les automates cellulaires comme modèle de calcul parallèle

Introduits par John von Neumann qui cherchait à développer un modèle de calcul capable d'auto-réplication, puis simplifiés sur une suggestion de Stanislaw Ulam, les *automates cellulaires* (AC) virent le jour au début des années 50. Bien que majoritairement étudiés en informatique, on retrouve les AC dans divers domaines comme la biologie, la chimie, la physique...

Dans cette thèse, on s'intéresse aux classes de complexité définies par les automates cellulaires. On peut voir les automates cellulaires comme un modèle de calcul massivement parallèle utilisant une communication locale : chaque cellule de l'automate ne peut communiquer qu'avec les cellules proches d'elle. Ce modèle de communication local est particulièrement intéressant car contrairement à d'autres modèles de programmation parallèle comme les machines PRAM, la communication des AC est locale, donc réaliste, et uniforme : toutes les cellules de l'automate communiquent localement et suivent le même programme, invariant le long du calcul.
De plus, les classes de complexité définies par les AC sont intéressantes. Par exemple, le temps linéaire des AC permet de résoudre de nombreux problèmes algorithmiques comme le produit d'entiers [2, 34], le produit de matrices, le tri, etc. et admet une caractérisation logique indépendante. Cette thèse qui étend les résultats de nos articles [20, 22] s'inscrit comme ces articles dans le prolongement d'un article de Bacquey et al. [3]. Dans celui-ci, les auteurs prouvent que la classe des AC en temps linéaire est exactement caractérisée par une logique obtenue à partir de la logique ESO-HORN, conçue par Grädel [16] pour caractériser P, en restreignant en particulier le nombre de variables du premier-ordre et l'arité des prédicats du second-ordre (voir aussi [21] qui traite le cas non déterministe linéaire). La classe du temps linéaire des AC est si expressive qu'il n'existe pas, à ce jour, d'exemple permettant de la différencier de façon prouvée de la classe DSPACE(n) des problèmes décidables en mémoire linéaire sur machines de Turing ou sur machines RAM.

En plus du temps linéaire, l'autre notion de complexité très étudiée dans la littérature des automates cellulaires est le *temps-réel* ou *temps minimal* [6, 12, 43].

Pourquoi s'intéresser au temps-réel des automates cellulaires ?

On dit d'un AC qu'il travaille en *temps-réel* si il donne une réponse, oui ou non, au temps *minimal* (le premier pas de temps) pour que la cellule de sortie (celle qui donne la réponse) reçoive l'intégralité de l'entrée. Cette notion dépend donc du type d'automate cellulaire. Les variantes de la

définition d'un AC reposent essentiellement sur le choix du *voisinage* de l'automate ainsi que sur la présentation *parallèle* ou *séquentielle* du mot d'entrée.
Il est important de souligner que, pour les AC de dimension 1, ces variantes définissent *exactement trois* classes de complexité, prouvées *distinctes* :

1. $\texttt{RealTime}_{\texttt{CA}} = \texttt{RealTime}_{\texttt{OIA}}$;
2. $\texttt{RealTime}_{\texttt{IA}}$;
3. $\texttt{Trellis} = \texttt{RealTime}_{\texttt{OCA}}$.

L'étape finale et décisive pour établir cette classification est une dichotomie sur les voisinages des AC [41]. Cette dichotomie peut être résumée comme suit : toute classe de complexité temps-réel sur un automate cellulaire de dimension 1 avec entrée parallèle et voisinage quelconque est :

— soit égale à la classe $\texttt{RealTime}_{\texttt{CA}}$;
— soit égale à la classe Trellis.

De plus, il est intéressant de noter que

— la hiérarchie entre les trois classes temps-réel données ci-dessus est entièrement connue : les classes $\texttt{RealTime}_{\texttt{IA}}$ et Trellis sont incomparables pour l'inclusion, tandis qu'on a l'inclusion stricte $\texttt{Trellis} \cup \texttt{RealTime}_{\texttt{IA}} \subsetneq \texttt{RealTime}_{\texttt{CA}}$ [6, 8, 44, 49] ;
— à l'inverse, on ne sait pas si l'inclusion triviale $\texttt{RealTime}_{\texttt{CA}} \subseteq \texttt{LinTime}_{\texttt{CA}}$ (où $\texttt{LinTime}_{\texttt{CA}}$ désigne la classe des problèmes décidables en temps linéaire sur AC) est stricte ou non ; pire, on ne sait même pas si l'inclusion $\texttt{RealTime}_{\texttt{CA}} \subseteq \texttt{DSPACE}(\texttt{n})$ est stricte ou non.

Alors qu'en dimension 2 ou supérieure, la notion de temps-réel apparaît comme fragile [17, 18, 19, 45, 47], notamment par sa sensibilité au voisinage choisi, en dimension 1, la complexité temps-réel avec ces variantes (1-3) est une notion *robuste* comme l'attestent les résultats ci-dessus.

Pourquoi caractériser en logique le temps-réel des automates cellulaires ?

Les trois classes de complexité temps-réel données ci-dessus contiennent des problèmes comme le produit d'entiers, la reconnaissance des mots dont la longueur est un nombre premier, le langage des palindromes, etc., et sont donc pertinentes à caractériser. De plus, chacune de ces trois classes admet deux ou trois définitions différentes mais équivalentes. Par exemple, la classe $\texttt{RealTime}_{\texttt{CA}}$ est égale à la classe du temps linéaire sur automate cellulaire avec entrée parallèle et voisinage unidirectionnelle [5, 48]. De même, Okhotin donne une caractérisation, indépendante des machines, de la classe Trellis ; c'est exactement celle des langages générés par les grammaires linéaires conjonctives [37, 38]. Dans cette thèse, on redémontrera ce résultat par l'intermédiaire de la logique. Enfin, Terrier prouve qu'un langage L appartient à $\texttt{RealTime}_{\texttt{IA}}$ si et seulement si le miroir de ce langage L^R est reconnu en temps-réel par un automate alternant avec voisinage unidirectionnel et un compteur [46].
Obtenir une caractérisation logique de ces trois classes d'automates cellulaires permet de faire d'une pierre deux coups :

— d'un côté celle-ci permet d'établir une caractérisation uniforme et indépendante des machines de chacune de ces classes ;
— de l'autre côté elle permet de définir des programmes logiques prouvés résolvant les problèmes de ces classes, ce qui, on le rappelle, est une tâche difficile ; ces programmes maintiennent la complexité minimale.

Ainsi, on définit dans le chapitre 1 la sémantique des logiques que l'on utilisera, avant de définir une logique de Horn pour chacune de nos trois classes d'AC dans le chapitre 2. On introduira dans ce même chapitre des variantes dites inductives de ces trois logiques, celles-ci autorisant un usage contrôlé de la négation (à comparer avec la négation utilisée dans Datalog stratifié [1].

Caractériser en logique une classe de complexité comporte deux avantages majeurs : première-

ment, les logiques sont *flexibles*, donc permettent une *expression* naturelle ; deuxièmement, elles peuvent être mises sous *forme normale* par un processus automatique.

Notre méthode pour construire un programme d'automate cellulaire

On donne dans cette thèse une méthode facilitant la programmation des AC en permettant la transformation automatique d'une formule (programme) logique en un programme d'AC. Cette méthode est la première méthode générique connue qui automatise en partie la construction des automates cellulaires. La figure 1 synthétise le fonctionnement de cette méthode.

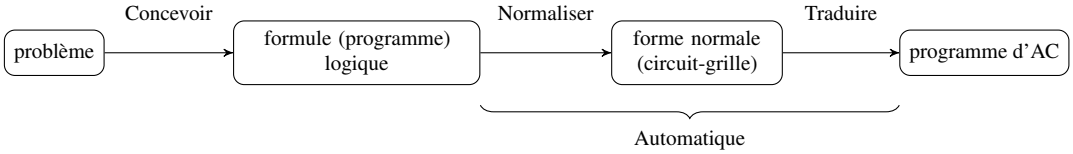


FIGURE 1 – Notre méthode

Concevoir.

Dans la littérature, les programmes d'automates cellulaires ne sont que rarement donnés explicitement. On y préfère souvent, pour un problème donné, présenter une stratégie sous forme de signaux et collisions et donner le programme de l'automate uniquement de manière informelle [7, 10, 11, 14, 33, 34, 51]. Cette présentation sous forme de signaux, même si elle ne donne pas explicitement le programme de l'AC, spécifie des schémas d'induction sur lesquels peut se baser un tel programme. Nos logiques, par leur flexibilité, permettent d'exprimer naturellement ces schémas d'induction.
À noter que le schéma d'induction résolvant un problème ne provient pas nécessairement d'un raisonnement géométrique comme les signaux. Le produit de deux entiers peut être défini inductivement par la méthode de multiplication scolaire usuelle. On donne dans cette thèse de nombreux exemples illustrant différentes méthodes de programmation en logique. On peut classer ces exemples selon trois sources de conception :

— induction directe :
  - le langage des palindromes (section 1.2.3),
  - le langage `Unbordered` (section 2.2.1),
  - le produit d'entiers (section 2.2.1),
  - le langage `Culik` (section 4.4),
  - le langage $\texttt{Dyck}_k$ (sections 2.2.2 et 6.4.3),
  - les langages linéaires conjonctifs (section 6.4) ;

— géométrie des signaux :
  - le langage `Disj` (section 2.2.1),
  - le langage `Prime` (section 2.2.2),
  - le langage $\texttt{Dyck}_k$ (section 6.4.3),
  - le langage `Culik` (section 4.4),
  - les langages conjonctifs (section 6.2.2),
  - les langages conjonctifs unaires (section 6.3) ;

— grammaires formelles :
  - le langage `HWB` (section 2.2.1),

- les langages conjonctifs (section 6.2.2 pour le cas général, section 6.3 pour la restriction unaire et section 6.4 pour la restriction linéaire).

Comme l'illustrent les listes ci-dessus, plusieurs exemples que l'on présentera dans cette thèse ne reposent pas sur une unique source de conception. C'est le cas du langage `Culik`, langage lié au problème de la synchronisation, problème central en programmation parallèle. On construit dans le chapitre 4 à partir d'un schéma d'induction complexe, une induction globale mise en place par des signaux, une formule logique définissant ce problème.
À travers ces multiples exemples, on espère convaincre le lecteur que la flexibilité de nos logiques permet une programmation naturelle.

Normaliser.

La méthode de normalisation donnée dans le chapitre 3 est la contribution majeure de cette thèse. Cette méthode de normalisation, entièrement *automatisable*, permet de transformer toutes les formules de nos logiques (variantes inductives incluses), en formules équivalentes dites *formules de grille*. En plus de nous permettre d'établir une équivalence entre chacune de nos logiques de Horn et sa variante inductive, cette forme normale paraphrase le calcul d'un *circuit-grille* : un modèle de calcul défini dans le chapitre 2.

Traduire.

Le circuit-grille est un modèle de calcul local dont le principal avantage est de pouvoir être traduit naturellement en programme d'automate cellulaire. Une fois le programme du circuit-grille déduit de la formule normalisée, le passage à l'automate se fait de façon uniforme par des transformations géométriques simples : changement de variables, pliage, etc...
Il est important de noter que cette traduction est bijective et qu'il est donc tout aussi naturel de déduire d'un programme d'automate cellulaire reconnaissant un langage, un programme de circuit-grille et donc une formule (programme) logique définissant ce même langage. Cette bijectivité nous permet ainsi d'établir pour chacune des trois classes temps-réel des AC son égalité avec d'une part la classe des problèmes définis par la logique de Horn correspondante, d'autre part avec la classe similaire pour la logique inductive.

Liens entre logique, automates cellulaires et grammaires formelles

Cette thèse établit trois nouvelles caractérisations de classes de complexité par la logique. Bien que dans la littérature la majeure partie des caractérisations soient réalisées en logique, les *grammaires formelles* permettent parfois elles aussi de caractériser des classes de complexité de manière indépendante des machines. On peut ainsi trouver dans la littérature quelques caractérisations, à la fois par la logique et par les grammaires formelles, de classes de complexité. La plus célèbre d'entre elles est celle des langages rationnels, caractérisés à la fois par leur complexité minimale en espace (espace constant sur machine de Turing, équivalent à la dimension 0 pour les automates cellulaires avec entrée séquentielle), par les grammaires rationnelles et par la logique *MSO* (logique monadique du second-ordre).
En ce qui concerne les classes de complexité d'automates cellulaires étudiées dans cette thèse, plusieurs résultats existent déjà dans la littérature : les langages linéaires algébriques sont inclus dans la classe `Trellis` [9], classe montrée égale, par Okhotin, aux langages engendrés par une grammaire linéaire conjonctive [37]. On redémontre cette égalité dans le chapitre 6 en montrant que les formules d'une de nos logiques de Horn (logique de Horn inclusive) paraphrasent les grammaires linéaires conjonctives et réciproquement. La classe des problèmes définis en logique inclusive étant prouvée égale à la classe `Trellis` dans le chapitre 3, on retrouve le résultat d'Okhotin. On obtient ainsi, comme pour la classe des langages réguliers, une caractérisation multiple de la classe

Trellis, renforçant l'intérêt de celle-ci.
Un des principaux résultats concernant les grammaires formelles obtenu durant cette thèse est l'inclusion de la classe des langages conjonctifs (Conj) dans celle des langages pouvant être reconnus en temps-réel par un automate cellulaire de dimension 2 avec entrée séquentielle et voisinage unidirectionnel (2-$\texttt{RealTime}_{\texttt{OIA}}$). Ce dernier résultat est obtenu en exprimant l'induction mise en œuvre dans l'algorithme de Cocke-Kasami-Younger dans une logique (incl-pred-ESO-HORN) équivalente à un circuit-grille de dimension 3 ($\texttt{Cube}_{\texttt{diag}}$), lui même inclus dans la classe 2-$\texttt{RealTime}_{\texttt{OIA}}$:

$$\texttt{Conj} \subseteq \texttt{incl-pred-ESO-HORN} = \texttt{Cube}_{\texttt{diag}} \subseteq \text{2-}\texttt{RealTime}_{\texttt{OIA}}$$

Ce résultat améliore un théorème de Kosaraju [28] datant de 1975, théorème disant que tout langage algébrique est reconnu en temps linéaire par un "2-dimensional array automaton" (variante des AC avec entrée séquentielle et voisinage unidirectionnel).
On montre de la même manière, à l'aide de la logique, que les langages unaires générés par les grammaires conjonctives (UnaryConj) sont reconnaissables en temps réel par un automate cellulaire avec entrée séquentielle.

Structure de la thèse

En plus de ce chapitre d'introduction, cette thèse est composée de six chapitres.

- Le chapitre 1 définit les notions centrales de cette thèse. On y donne une définition formelle des automates cellulaires ainsi que de la notion de temps-réel. La sémantique des formules de Horn que l'on utilisera par la suite ainsi que la structure de mot sur laquelle porteront ces formules y sont aussi définies.
- Le chapitre 2 se décompose en trois sections. La première section est dédiée au circuit-grille, modèle de calcul nous servant à faire le lien entre la logique et les AC. On y détaille son fonctionnement et les trois classes de complexité qu'il définit. Dans la seconde section, on introduit trois logiques de Horn (et leurs variantes inductives). La programmation dans chacune de ces logiques est illustrée par un exemple que l'on a voulu représentatif. Enfin, la dernière section du chapitre 2 donne un exemple de programmation d'AC à l'aide d'une formule logique.
- Le chapitre 3 contient le coeur de cette thèse. On y détaille pour chacune de nos trois logiques une méthode de normalisation aboutissant à une formule paraphrasant le calcul d'un circuit-grille. En plus de cette normalisation, on donne pour chacun des trois circuits-grille une méthode permettant de traduire leur calcul en calcul d'AC et réciproquement. Ainsi, on prouve dans ce chapitre trois théorèmes établissant chacun une égalité entre une classe de définissabilité logique et une classe de complexité d'AC.
- Le chapitre 4 est consacré au langage Culik. Ce langage est particulièrement intéressant pour son lien avec le problème de synchronisation, central en programmation parallèle. On construit dans ce chapitre une formule logique définissant ce langage. Le but de ce chapitre n'est pas de redémontrer l'appartenance de ce langage à la classe Trellis mais plutôt d'illustrer, sur un exemple représentatif, les différents types d'inductions que peut exprimer notre logique.
- Le chapitre 5 aborde, par le cas du circuit-grille de dimension 3, la question de la généralisation de nos résultats aux dimensions supérieures à 2. On y définit une généralisation du circuit-grille à toute dimension. On s'appuie ensuite sur cette définition pour exhiber les différentes classes de complexité pouvant être définies sur le circuit-grille de dimension 3.
- Le chapitre 6 s'intéresse aux relations entre les grammaires conjonctives, nos logiques et les classes de complexité temps-réel des AC. On y démontre notamment, par l'intermédiaire de la logique, que les langages conjonctifs sont reconnus en temps-réel par un automate

cellulaire de dimension 2 avec entrée séquentielle et voisinage unidirectionnel ; on y prouve aussi que les langages conjonctifs sur un alphabet unaire sont eux reconnaissables par un AC de dimension 1 avec entrée séquentielle. Enfin, on redémontre dans ce chapitre le résultat d'Okhotin sur l'équivalence entre être reconnaissable par un automate treillis et être générable par une grammaire linéaire conjonctive.

La figure 2 résume les égalités, inclusions et appartenances établies dans cette thèse. Les classes temps-réel et les classes logiques se trouvant sur une même ligne ont été montrées égales dans cette thèse, à l'exception de la dernière ligne où la logique est seulement incluse dans la classe temps-réel. Les problèmes se trouvant dans la colonne de droite appartiennent à la classe de complexité correspondant à la ligne. Certaines de ces appartenances ont déjà été obtenues dans la littérature mais toutes sont démontrées dans cette thèse.

Temps réel des AC	**Logique**	**Problèmes**
$\texttt{RealTime}_{\texttt{CA}}$ $= \texttt{RealTime}_{\texttt{OIA}}$	pred-ESO-HORN $=$ pred-ESO-IND	produit d'entiers, Unbordered
$\texttt{RealTime}_{\texttt{IA}}$	pred-dio-ESO-HORN $=$ pred-dio-ESO-IND	Disj, Prime, UnaryConj
Trellis $= \texttt{RealTime}_{\texttt{OCA}}$	incl-ESO-HORN $=$ incl-ESO-IND	langage des palindromes, HWB, $\texttt{Dyck}_{\texttt{k}}$, Culik, LinConj
2-$\texttt{RealTime}_{\texttt{OIA}}$	incl-pred-ESO-HORN	Conj

FIGURE 2 – Tableau récapitulatif des égalités, inclusions et appartenances montrées dans cette thèse

1 Préliminaires

1.1 Les automates cellulaires et le temps-réel

Hormis au chapitre 5 et, partiellement, au chapitre 6 où nous évoquerons des automates cellulaires de dimension 2 ou plus, les automates cellulaires que nous étudierons dans cette thèse sont de dimension 1.

1.1.1 Automates cellulaires en dimension 1

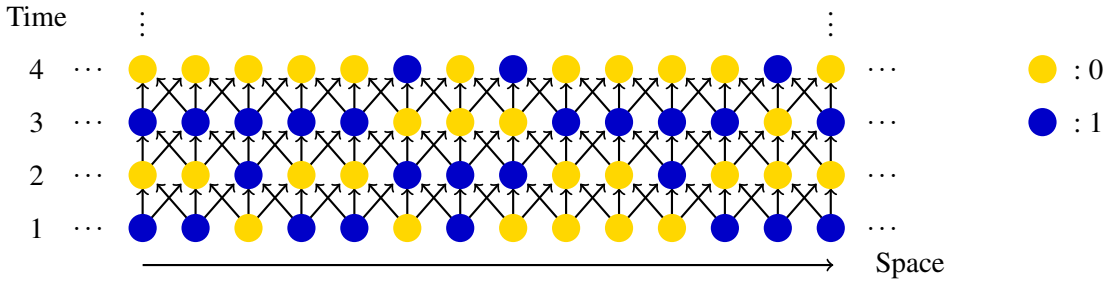
Définition

Un automate cellulaire (AC) de dimension 1 consiste en une ligne d'automates finis tous identiques appelés cellules. Chacune de ces cellules tire sa valeur d'un *ensemble d'états* fini $\mathtt{S}$ et interagit avec les cellules voisines à des pas de temps discrets. Au temps initial $t = 1$, chaque cellule reçoit un état de $\mathtt{S}$. Le calcul est ensuite réalisé localement : le nouvel état d'une cellule est le résultat d'une *fonction de transition* qui dépend des états de son *voisinage* au pas de temps précédent. Cette fonction de transition s'applique de façon synchrone à toutes les cellules et à chaque pas de temps. Formellement :

Définition 1.1.1 Un automate cellulaire (AC) est un triplet $(\mathtt{S}, \mathscr{N}, \mathtt{f})$ où :
- $\mathtt{S}$ est un ensemble fini d'états ;
- $\mathscr{N} = \{n_1, \dots, n_{|\mathscr{N}|}\} \subset \mathbb{Z}$ est le voisinage ;
- $\mathtt{f} : \mathtt{S}^{|\mathscr{N}|} \to \mathtt{S}$ est la fonction de transition.

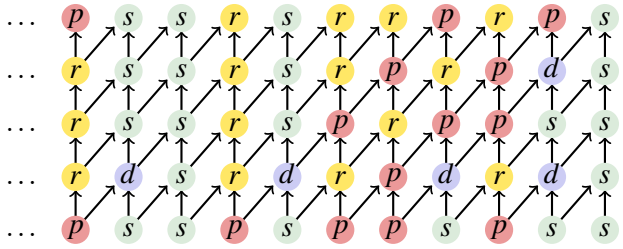
En notant $\langle c,t \rangle$ l'état de la cellule c au temps t, l'évolution de l'état d'une cellule suit le schéma suivant : $\langle c, t+1 \rangle = \mathtt{f}(\langle c+n_1, t \rangle, \dots, \langle c+n_{|\mathscr{N}|}, t \rangle)$

■ **Exemple 1.1** Considérons l'automate cellulaire $\mathscr{A} = (\{0,1\}, \{-1,0,1\}, \mathtt{f})$ où $\mathtt{f} : \{0,1\}^3 \to \{0,1\}$ est définie par $\mathtt{f}(x,y,z) = \max(x,z) \times (1-y) + \min(1-x, 1-z) \times y$. Les premières étapes de calcul sur la configuration initiale $\dots 11011010000111 \dots$ sont représentées par le *diagramme espace-temps* suivant.



Un autre exemple d'automate cellulaire est l'automate $\mathscr{T} = (\{d,p,s,r\}, \{-1,0\}, \mathtt{f})$ dont la fonction de transition $\mathtt{f} : \{p,s,d,r\}^2 \to \{p,s,d,r\}$ est donnée par la table ci-dessous. Sur la droite, le diagramme espace-temps représente les premières étapes de calcul sur l'entrée $\dots psspsppspss$.

	p	s	d	r
p	p	d	p	p
s	r	s	r	r
d	r	s	r	p
r	r	s	s	r



■

La principale différence entre les deux AC donnés en exemple est la circulation de l'information. Là où l'information peut circuler dans les deux directions pour $\mathscr{A}$ grâce au voisinage $\{-1,0,1\}$, le voisinage $\{-1,0\}$ de $\mathscr{T}$ contraint l'information à circuler de la gauche vers la droite. On parle alors d'automates avec communication bidirectionnelle pour le premier (dont le voisinage canonique est $\{-1,0,1\}$) et d'automates avec communication unidirectionnelle (avec $\{-1,0\}$ pour voisinage canonique) pour le second.

Les automates cellulaires comme reconnaisseurs de langage

Dans cette thèse, on s'intéresse aux automates cellulaires comme *reconnaisseurs de langage*. De tels automates opèrent sur un mot d'entrée $w \in \Sigma^+$ (avec Σ un alphabet) et donnent en sortie l'une des deux réponses "acceptation" ou "rejet". Il convient alors de préciser comment le mot d'entrée est transmis à l'automate et comment le résultat du calcul est obtenu.

Tout d'abord, on donne certaines conditions sur l'ensemble des états de l'automate $\mathtt{S}$.

— Les lettres du mot d'entrée appartiennent à l'ensemble des états : $\Sigma \subset \mathtt{S}$.
— Un sous-ensemble d'*états acceptants* $\mathtt{S}_{acc} \subset \mathtt{S}$ est identifié.
— Deux états spéciaux sont aussi distingués : un *état permanent* $\sharp$ et un *état quiescent* λ. Une cellule dans l'état permanent $\sharp$ reste dans cet état indéfiniment. Une cellule dans l'état quiescent λ reste dans cet état tant que son voisinage est lui aussi quiescent.

En ce qui concerne la transmission du mot d'entrée à l'automate, on considère généralement les deux modes de communication suivants : le *mode parallèle* et le *mode séquentiel*. En mode parallèle, l'intégralité de l'entrée est fournie à l'automate au temps initial : le i-ème symbole de l'entrée $w = w_1 \dots w_n$ est donné à la cellule i au temps 1 : $\langle i,1\rangle = w_i$. En mode séquentiel, pour une entrée de taille n, toutes les cellules d'indice dans $[1,n]$ se trouvent initialement dans l'état quiescent λ et les n symboles de l'entrée sont lus l'un après l'autre par la cellule d'indice 1 : w_t est donné à la cellule 1 au temps t. Ce mode d'entrée requiert une fonction de transition spécifique $\mathtt{f}_{input} : \Sigma \times \mathtt{S}^{|\mathscr{N}|} \to \mathtt{S}$ appliquée uniquement à la cellule 1 : $\langle 1,1\rangle = \mathtt{f}_{input}(w_1,\lambda,\dots,\lambda)$, $\langle 1,t\rangle = \mathtt{f}_{input}(w_t,\langle 1+n_1,t-1\rangle,\dots,\langle 1+n_{|\mathscr{N}|},t-1\rangle)$. De plus, dans ces deux modes, le calcul est délimité par la longueur de l'entrée : initialisées dans l'état permanent $\sharp$, les cellules d'indice ne se trouvant pas dans $[1,n]$ restent inactives. Dans la littérature, un automate cellulaire dont l'entrée est transmise de façon séquentielle est aussi appelé *"iterative array"* (IA).

La sortie de l'automate n'étant qu'une des deux réponses "acceptation" ou "rejet", une unique cellule, appelée *cellule de sortie*, est suffisante pour lire le résultat du calcul. Pour le voisinage $\{-1,0,1\}$ qui permet une communication bidirectionnelle, la cellule de sortie est généralement la cellule 1. Pour le voisinage $\{-1,0\}$ qui implique une communication unidirectionnelle la cellule de sortie est la cellule n, la seule pouvant avoir accès à la totalité de l'information de l'entrée. Enfin, on dit qu'un mot w est *accepté* en temps t par un automate si au temps t la cellule de sortie se trouve dans un état de $\mathtt{S}_{acc}$.

1.1.2 Le temps-réel des automates cellulaires

Le calcul d'un automate s'effectue en *temps-réel* (ou *temps minimal*) sur une entrée w si ce calcul s'arrête dès que la cellule de sortie a reçu la totalité de l'information de w. On dit alors qu'un langage est reconnu en temps-réel si chacun des mots de ce langage est accepté en temps-réel.

En ce qui concerne les AC de dimension 1, il existe 4 classes de langages temps-réel, selon si :

— le mode d'entrée est parallèle ou séquentiel ;
— la communication est bidirectionnelle (avec le voisinage canonique $\{-1,0,1\}$) ou unidirectionnelle (avec le voisinage $\{-1,0\}$).

Entrée parallèle et communication bidirectionnelle.

La classe $\texttt{RealTime}_{\texttt{CA}}$ est l'ensemble des langages reconnaissables en temps-réel par un automate cellulaire avec voisinage $\{-1,0,1\}$ et entrée parallèle. Cette classe est équivalente à la classe $\texttt{RealTime}_{\texttt{OIA}}$ des langages reconnaissables en temps-réel par un automate avec voisinage $\{-1,0\}$ et entrée séquentielle.

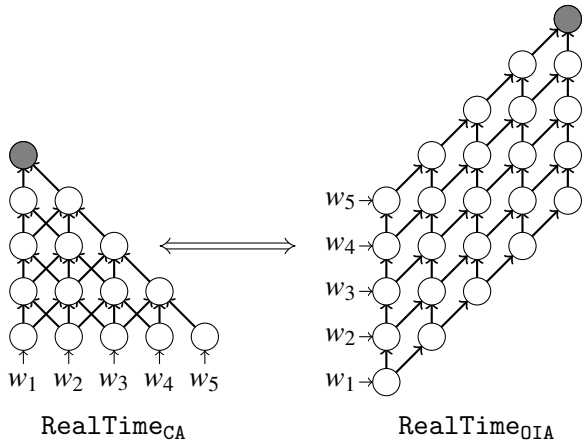


FIGURE 1.1 – Le diagramme espace-temps des classes $\texttt{RealTime}_{\texttt{CA}}$ et $\texttt{RealTime}_{\texttt{OIA}}$

Entrée séquentielle et communication bidirectionnelle.

La classe $\texttt{RealTime}_{\texttt{IA}}$ est l'ensemble des langages reconnaissables par un automate cellulaire avec voisinage $\{-1,0,1\}$ et entrée séquentielle.

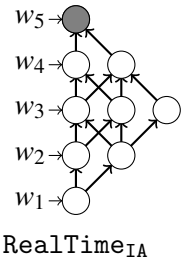


FIGURE 1.2 – Le diagramme espace-temps de la classe $\texttt{RealTime}_{\texttt{IA}}$

Entrée parallèle et communication unidirectionnelle.

La classe $\texttt{RealTime}_{\texttt{OCA}}$ est l'ensemble des langages reconnaissable en temps-réel par un automate cellulaire avec voisinage $\{-1,0\}$ et entrée parallèle. Cette classe de complexité est équivalente à `Trellis`, l'ensemble des langages reconnaissable par un automate treillis.

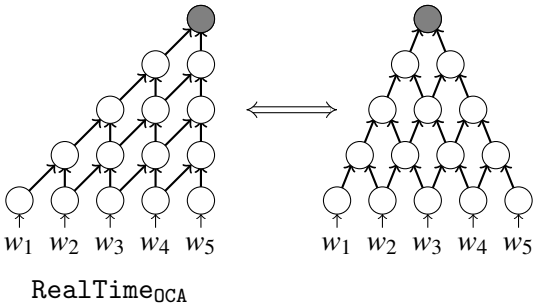


FIGURE 1.3 – Le diagramme espace-temps des classes RealTime$_{\text{OCA}}$ et Trellis

1.2 Formules de Horn sur des structures de mot

En vis-à-vis des automates cellulaires opérant en temps-réel, on introduit deux types de logiques. Le premier est celui des formules de Horn. Le second est une extension des formules de Horn avec une contrainte d'acyclicité : il s'agit des formules inductives. On présente d'abord ces notions dans le cadre du calcul propositionnel, ceci pour des raisons de simplicité pédagogique et aussi pour les preuves.

1.2.1 Sémantique propositionnelle

Une *formule de Horn propositionnelle* consiste en une conjonction de clauses appelées *clauses de Horn* portant sur des *propositions*, appelées aussi *variables propositionnelles* ou simplement *variables*. Ces clauses peuvent être

— soit des *clauses de Horn strictes* de la forme $p_1 \wedge p_2 \wedge \cdots \wedge p_h \rightarrow p_c$ où les propositions $(p_i)_{i\in[1,h]}$ avec $h \geq 0$ jouent le rôle d'hypothèses et la proposition p_c celui de conclusion, cette clause se lisant ainsi : "SI p_1 est vraie ET p_2 est vraie … ET p_h est vraie ALORS p_c est vraie" ;

— soit des *clauses (de Horn) négatives* de la forme $p_1 \wedge p_2 \wedge \ldots p_h \rightarrow \bot$ où $\bot$ est la valeur *faux* et $h \geq 1$.

Pour une formule propositionnelle F, on appelle *interprétation* une fonction I associant à chaque variable de F une valeur de vérité (1 pour le vrai et 0 pour le faux).

I est un *modèle* de F, ce qu'on note $I \models F$, si F est évaluée à vrai quand chaque variable p est remplacée par son interprétation $I(p)$.

Une clause de Horn stricte sans hypothèse, notée $\rightarrow p$ ou simplement p, qu'on appelle un *fait*, exprime que p est vraie.

Modèle minimal

À toute conjonction de clauses de Horn strictes appelée *formule de Horn stricte* est associé un *modèle minimal* unique. Le modèle minimal d'une formule de Horn stricte F est défini comme le modèle I_0 de l'équivalence : $I_0(p) = 1 \iff$ l'implication $F \rightarrow p$ est une tautologie. Ainsi, pour toute proposition p de F on a l'implication $I_0(p) = 1 \Rightarrow (\forall I : I \models F \Rightarrow I(p) = 1)$. Le modèle minimal de F est obtenu à partir des faits de F par la clôture de l'opération d'implication. On associe à une interprétation I l'ensemble des variables p telles que $I(p) = 1$, ensemble qu'on appelle aussi I par abus de notation : le modèle minimal de F est alors l'intersection de tous les modèles de F.

■ **Exemple 1.2** Soit F la formule de Horn stricte suivante :

$$(\rightarrow p_1)\bigwedge(\rightarrow p_2)\bigwedge(p_1 \wedge p_2 \rightarrow p_3)\bigwedge(p_2 \wedge p_4 \rightarrow p_5)$$

En représentant une interprétation comme l'ensemble des indices des propositions vraies dans cette interprétation, les modèles satisfaisant F sont : $m_1 = \{1,2,3,4,5\}$, $m_2 = \{1,2,3,5\}$ et le modèle minimal $m = \{1,2,3\}$. La minimalité de m peut alors aussi être vue de façon ensembliste car, pour tout modèle m_i satisfaisant F, on a $m \subseteq m_i$.

■

La notion de modèle minimal est essentielle pour tester si une formule de Horn (conjonction de clauses de Horn strictes et de clauses négatives) est satisfaisable, à partir de ce modèle. C'est ce que dit le lemme suivant :

Lemme 1.2.1 — Modèle minimal et satisfaisabilité. Une formule de Horn propositionnelle $F := \bigwedge_i \theta_i \wedge \bigwedge_j C_j$ où les θ_i sont des clauses de Horn strictes et les C_j sont des clauses de Horn négatives est satisfaisable *ssi* le modèle minimal I_0 de la conjonction $\bigwedge_i \theta_i$ satisfait la conjonction des clauses négatives $\bigwedge_j C_j$.

Preuve. On n'a qu'à démontrer l'implication $\Rightarrow$, la réciproque étant évidente.

Supposons donc F satisfaisable. Soit I un modèle de F. En particulier, $I \models \bigwedge_i \theta_i$ et donc $I \supseteq I_0$, autrement dit, pour chaque variable propositionnelle p_h telle que $I_0(p_h) = 1$, on a $I(p_h) = 1$. En d'autres termes, $I(p_h) = 0$ implique $I_0(p_h) = 0$.

Puisque I est modèle de la conjonction de clauses négatives $\bigwedge_j C_j$, il existe, pour chaque clause négative C_j, donc de la forme $C_j := p_{i_1} \wedge \ldots \wedge p_{i_\ell} \rightarrow \bot$, une variable p_{i_r} telle que $I(p_{i_r}) = 0$, donc telle que $I_0(p_{i_r}) = 0$. En conséquence, $I_0 \models \bigwedge_j C_j$.

□

Formule de Horn propositionnelle acyclique

L'acyclicité sera une caractéristique essentielle pour définir la sémantique de nos formules inductives. Pour commencer, on a besoin d'un résultat préliminaire utile pour justifier une caractérisation alternative du modèle minimal d'une formule de Horn acyclique.

Le résultat suivant exprime que si une variable propositionnelle p est conséquence d'un ensemble A d'implications, alors il y a toujours dans A une implication $\alpha \rightarrow p$ dont la conclusion est p et dont l'hypothèse α est conséquence de l'ensemble des implications de A privé de l'implication $\alpha \rightarrow p$.

Lemme 1.2.2 Soit F une formule de Horn stricte et $\alpha := \bigvee_{i=1}^{k} \alpha_i$ une disjonction de conjonctions α_i de variables propositionnelles. Soit p une variable propositionnelle qui n'apparaît pas dans α et n'est conclusion d'aucune clause de F.

1. Si $(F \wedge (\alpha \rightarrow p)) \rightarrow p$ est une tautologie alors $F \rightarrow \alpha$ est aussi une tautologie.
2. Soit I_0 le modèle minimal de la formule de Horn stricte $F \wedge \bigwedge_{i=1}^{k} (\alpha_i \rightarrow p)$. Alors I_0 est modèle de l'équivalence $p \leftrightarrow \bigvee_{i=1}^{k} \alpha_i$.

Preuve du 1 par contraposée. Supposons que $F \rightarrow \alpha$ n'est pas une tautologie. Il existe donc une interprétation I telle que $I \models F$ et $I \models \neg\alpha$. Soit I_p l'interprétation définie par $I_p(p) = 0$ et $I_p(q) = I(q)$, pour toute variable $q \neq p$. Puisque p n'apparaît pas dans α, on a $I_p \models \neg\alpha$. Puisque $I \models F$ et puisque p n'est conclusion d'aucune clause de F, on a $I_p \models F$. On en déduit $I_p \models F \wedge (\alpha \rightarrow p)$ et donc $I_p \not\models (F \wedge (\alpha \rightarrow p)) \rightarrow p$, ce qui implique que la formule $(F \wedge (\alpha \rightarrow p)) \rightarrow p$ n'est pas une tautologie. □

Preuve du 2. La seule chose à montrer est que I_0, le modèle minimal de $F \wedge \bigwedge_{i=1}^{k}(\alpha_i \to p)$, est modèle de l'implication $p \to \alpha$ (c'est-à-dire $p \to \bigvee_{i=1}^{k} \alpha_i$).

Supposons donc $I_0 \models p$. Soit I un modèle quelconque de $F \wedge (\alpha \to p)$, donc de $F \wedge \bigwedge_{i=1}^{k}(\alpha_i \to p)$. On a $I \supseteq I_0$, par définition de I_0, donc $I \models p$, ce qui établit que l'implication $(F \wedge (\alpha \to p)) \to p$ est une tautologie. Utilisant le point 1, on en déduit que $F \to \alpha$ est aussi une tautologie. Comme I_0 est modèle de F, on en tire $I_0 \models \alpha$. On a donc prouvé $I_0 \models p \to \alpha$. □

Définition 1.2.1 — formule de Horn acyclique. On appelle *graphe* d'une formule de Horn stricte F le graphe orienté G_F dont l'ensemble des sommets est l'ensemble des variables de F et l'ensemble des arcs est constitué des couples (p,q) tels qu'il existe une clause $C := \alpha \to q$ de F où p appartient à la conjonction α.

La formule F est dite *acyclique* si son graphe G_F est acyclique.

Du lemme 1.2.2 on déduit le corollaire suivant qui donne une caractérisation alternative du modèle minimal d'une formule de Horn stricte et acyclique :

Corollaire 1.2.3 — modèle minimal d'une formule acyclique. Soit F une formule de Horn stricte et acyclique sur les variables $p_1, \dots p_k$, qu'on peut écrire sous la forme

$$F := \bigwedge_{j=1}^{k} \bigwedge_{\alpha \in A_j} (\alpha \to p_j).$$

Soit I_0 le modèle minimal de F. Soit I_1 l'unique modèle de la conjonction d'équivalences

$$E(F) := \bigwedge_{j=1}^{k} \left(p_j \leftrightarrow \bigvee_{\alpha \in A_j} \alpha \right).$$

(l'unicité vient de l'acyclicité du graphe G_F). Alors $I_0 = I_1$.

Démonstration. En appliquant le point 2 du lemme 1.2.2 à chaque variable p_j, on obtient que le modèle minimal I_0 de $F := \bigwedge_{j=1}^{k} \bigwedge_{\alpha \in A_j} (\alpha \to p_j)$ est modèle des k équivalences $p_j \leftrightarrow \bigvee_{\alpha \in A_j} \alpha$. Par conséquent, I_0 est l'unique modèle I_1 de E. □

La caractérisation d'un modèle unique (construit par une succession d'équivalences) donné par le corollaire 1.2.3 nécessite le critère d'acyclicité. En respectant ce critère, on peut généraliser cette caractérisation au cas où les variables peuvent être niées.

Formule de Horn propositionnelle inductive

Définition 1.2.2 — formule d'équivalence, formule inductive. Une formule propositionnelle E sur les variables $p_1, \dots, p_k$ est une *formule d'équivalence* si elle est de la forme

$$E := \bigwedge_{j=1}^{k} (p_j \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$$

où chaque α est une conjonction, possiblement vide, de littéraux p_i ou $\neg p_i$ et chaque A_j est un ensemble, possiblement vide, de tels α.

On appelle *graphe* de E le graphe orienté G_E dont l'ensemble des sommets est l'ensemble des variables de E et l'ensemble des arcs est constitué des couples (p_i, p_j) tels que p_i ou $\neg p_i$

appartient à au moins une conjonction $\alpha \in A_j$.

Une formule d'équivalence E dont le graphe G_E est acyclique est appelée *formule inductive*.

■ **Exemple 1.3** La formule E_0 suivante (où $\top$ désigne le "vrai" ou la conjonction vide) est une formule d'équivalence :

$$E_0 := (p_1 \leftrightarrow \top) \wedge (p_2 \leftrightarrow \bot) \wedge (p_3 \leftrightarrow ((p_1 \wedge \neg p_2) \vee p_2)) \wedge (p_4 \leftrightarrow \bot) \wedge (p_5 \leftrightarrow (p_1 \wedge p_4)).$$

E_0 est une formule inductive puisque son graphe, G_{E_0}, constitué des arcs (p_1, p_3), (p_2, p_3), (p_1, p_5) et (p_4, p_5), est acyclique. ■

La proposition suivante est une conséquence immédiate de la définition 1.2.2 :

Proposition 1.2.4 Une formule inductive a un unique modèle.

Exemple : La formule inductive E_0 de l'exemple 1.3 a un unique modèle I_0 construit inductivement à partir de son graphe acyclique G_{E_0}, ce qui donne $I_0(p_1) = 1$, $I_0(p_2) = 0$, $I_0(p_3) = 1$, $I_0(p_4) = 0$ et $I_0(p_5) = 0$.

Intuition : Le corollaire 1.2.3, affirme que le modèle minimal (modèle naturel) d'une formule de Horn stricte et acyclique $F := \bigwedge_{j=1}^{k} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow p_j)$ est aussi l'unique modèle de la formule d'équivalence (formule inductive) $E(F) := \bigwedge_{j=1}^{k} (p_j \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$ associée à F.

Intuitivement, le "modèle naturel" de F met à "vrai" une variable p_j s'il existe dans F *au moins une* implication $\alpha \rightarrow p_j$ dont l'hypothèse α a été mise à "vrai" ; sinon, on met p_j à "faux".

En généralisant à une formule "acyclique" de la forme $F := \bigwedge_{j=1}^{k} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow p_j)$ où chaque α est une conjonction de littéraux $(\neg)p_j$ (au lieu d'une conjonction de variables, donc F n'est plus une formule de Horn), à laquelle on associe la formule inductive $E(F) := \bigwedge_{j=1}^{k} (p_j \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$, on appelle "modèle naturel" de F l'interprétation

— qui met à "vrai" une variable p_j s'il existe dans F au moins une implication $\alpha \rightarrow p_j$ dont l'hypothèse α a été mise à "vrai"

— et qui, sinon, met p_j à "faux".

Il s'agit donc de l'unique modèle de $E(F)$.

Par exemple, le "modèle naturel" de la formule "acyclique"

$$F_0 := (\rightarrow p_1) \wedge (p_1 \wedge \neg p_2 \rightarrow p_3) \wedge (p_2 \rightarrow p_3) \wedge (p_1 \wedge p_4 \rightarrow p_5)$$

est l'unique modèle I_0 de la formule inductive associée $E_0 := E(F_0)$ (voir exemple 1.3) :

$$E_0 := (p_1 \leftrightarrow \top) \wedge (p_2 \leftrightarrow \bot) \wedge (p_3 \leftrightarrow ((p_1 \wedge \neg p_2) \vee p_2)) \wedge (p_4 \leftrightarrow \bot) \wedge (p_5 \leftrightarrow (p_1 \wedge p_4)).$$

Dans la sous-section suivante, on généralisera les notions de formules de Horn et de formules inductives et les résultats sur leur sémantique à la logique du premier ordre et à nos logiques (logiques du second ordre existentiel).

1.2.2 Sémantique en logique du premier ordre

La logique du premier ordre ou calcul des prédicats a été introduite par Gottlob Frege à la fin du XIX^e siècle dans le but de formaliser le langage des mathématiques. Cette logique utilise des *variables* prenant leurs valeurs dans un domaine donné. En logique du premier ordre, contrairement au calcul propositionnel, ces variables peuvent être quantifiées et mises en relation notamment par l'utilisation de *prédicats* et de *fonctions*. Ces prédicats (ou ces fonctions) peuvent prendre en argument une ou plusieurs variables, on parle alors d'arité du prédicat : un prédicat d'arité 2 (ou prédicat binaire) prend 2 variables en arguments. Les prédicats et les fonctions sont interprétés sur

des structures ayant chacune un domaine fixé (non vide), parfois appelées structures du premier ordre. Comme il est d'usage [23], on notera souvent par un même symbole un prédicat (une fonction) et son interprétation dans une structure du premier ordre. On fait cet abus de notation en définissant ci-dessous une structure de mot. De façon générale, on utilisera les notions et notations classiques [31] : signature d'une formule, structure, modèle d'une formule, etc.

Structure de mot

On s'intéresse aux langages $L \subseteq \Sigma^+$ formés de mots non vides sur un alphabet Σ. On utilise alors la structure suivante pour représenter un mot $w = w_1 \dots w_n \in \Sigma^+$:

$$\langle w \rangle = ([1,n]; (Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{max}, \texttt{pred}, \texttt{succ})$$

Où :

- l'intervalle d'entiers $[1,n]$ est le *domaine* dans lequel les variables prennent leurs valeurs ;
- $(Q_s)_{s\in\Sigma}$ est un ensemble de prédicats unaires appelés *prédicats d'entrée* tels que $Q_s(x) \iff w_x = s$;
- `min` et `max` sont des prédicats unaires avec leur sens usuel : $\texttt{min}(x) \iff x = 1$ et $\texttt{max}(x) \iff x = n$;
- `pred` est la fonction définie par $\texttt{pred}(1) = 1$ et $\texttt{pred}(x) = x-1$ pour $x \in [2,n]$;
- `succ` est la fonction définie par $\texttt{succ}(n) = n$ et $\texttt{succ}(x) = x+1$ pour $x \in [1,n-1]$.

$\langle w \rangle$ est appelée la *structure d'entrée* associée au mot (d'entrée) w. La structure $\langle w \rangle$ est de signature (ensemble de prédicats et de symboles de fonctions) $\mathscr{S}_\Sigma = \{(Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{pred}, \texttt{succ}\}$.

Notation 1.1 Pour un entier fixé $a \geq 0$, on note $x-a$ (resp. $x+a$) l'opération consistant à appliquer a fois l'opérateur `pred` (resp. `succ`) à x, soit : $x-a = \texttt{pred}^a(x)$ (resp. $x+a = \texttt{succ}^a(x)$). Pour des soucis de clarté, on s'autorisera par la suite à écrire $x = 1$ (resp. $x = n$) à la place de $\texttt{min}(x)$ (resp. $\texttt{max}(x)$). De même, pour un entier fixé $a > 0$, on utilisera librement les abréviations $x > a$ et $x = a$ (resp. $x \leq n-a$ et $x = n-a$) à la place de $\neg\texttt{min}(x-(a-1))$ et $\texttt{min}(x-(a-1)) \wedge \neg\texttt{min}(x-(a-2))$ (resp. $\neg\texttt{max}(x+(a-1))$ et $\texttt{max}(x+a) \wedge \neg\texttt{max}(x+(a-1))$.

Formules de Horn utilisées

Dans cette thèse, on utilisera principalement des formules de Horn du second ordre existentiel. Ces formules portent sur des structures de mots comme définies dans la section précédente et sont de la forme $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :

— $\mathbf{R} = \{R_1, \dots, R_n\}$ est un ensemble de variables de prédicats binaires R_i appelés *prédicats de calcul* prenant leurs valeurs dans l'ensemble des parties de $[1,n]^2$;
— x et y sont des variables prenant leur valeur dans le domaine $[1,n]$;
— $\forall x \forall y \psi(x,y)$ est une formule de Horn, c'est-à-dire une conjonction de clauses de Horn.

Les clauses de Horn utilisées dans la formule $\psi(x,y)$ sont toutes de la forme $\alpha_1 \wedge \dots \wedge \alpha_n \to \beta$ où :

— la conclusion β est soit le faux ($\bot$), soit un atome $R(x,y)$, avec $R \in \mathbf{R}$, appelé *atome de calcul* ;
— chaque hypothèse α_i est :
  — soit un *littéral d'entrée* $(\neg)\eta(x+a)$ ou $(\neg)\eta(y+a)$ avec a un entier (positif, négatif ou nul) et $\eta \in \{(Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{max}\}$;
  — soit un *atome de calcul* $R(x+a, y+b)$ avec a et b deux entiers (positifs, négatifs ou nuls) et $R \in \mathbf{R}$.

Si la conclusion β d'une clause est $\bot$ alors la clause est appelée *clause de Horn négative* ou *clause de contradiction*. Sinon, la clause est appelée *clause de Horn stricte* ou *clause de calcul*.
Une formule de Horn stricte est une formule $\forall x \forall y \psi(x,y)$ où ψ est une conjonction de clauses de Horn strictes.

Relation étroite entre logique de Horn du second ordre existentiel et logique de Horn propositionnelle

Comme on l'a vu en logique propositionnelle, la propriété du *modèle minimal* est la propriété clé des *formules de Horn* : si une formule de Horn est satisfaisable, alors elle a un modèle minimal qu'on appellera aussi *modèle naturel*.

Par souci d'application directe à nos problèmes et pour simplifier les notations, on présente ici la propriété du modèle minimal pour les formules de Horn à 2 variables x, y et de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$, pour $\mathbf{R} := \{R_1, \ldots, R_m\}$, formées de prédicats binaires R_i et s'appliquant à des structures mots $\langle w \rangle$, mais tout ce qu'on en dira est vrai quels que soient la structure sous-jacente, le nombre de variables et la signature de la formule.

On verra comment, à cause de la propriété du modèle minimal, la vérité d'une formule de Horn *du second ordre existentiel* se ramène à la vérité d'une formule de Horn *du premier ordre* dans "le" modèle minimal (proposition 1.2.7 ci-dessous). En particulier, cela s'applique aux logiques `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN` que l'on définira dans la section 2.2.

Toujours comme dans le cas propositionnel, nous verrons une nouvelle caractérisation du modèle minimal d'une formule de nos logiques sous la condition que cette formule satisfait un certain *critère d'acyclicité* : c'est aussi l'unique modèle d'une certaine *conjonction d'équivalences*.

L'intérêt de cette dernière caractérisation (par conjonction d'équivalences) est d'être généralisable aux logiques *inductives* `pred-ESO-IND`, `pred-dio-ESO-IND` et `incl-ESO-IND`, que nous définirons comme des extensions des logiques respectives `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN` quand on autorise la *négation* des atomes de calcul, tout en conservant la condition d'acyclicité.

On verra que ces logiques inductives permettent une programmation naturelle de problèmes intéressants, par exemple les langages de Dyck ou le langage de Culik, sachant que la condition d'acyclicité, elle-même naturelle, est facile à satisfaire.

Modèle minimal d'une formule de Horn :

Soit $\forall x \forall y \psi(x,y)$ une formule de Horn. Décomposons la formule ψ en $\psi := \psi_0 \wedge \psi_{neg}$ où ψ_0 est la conjonction de ses clauses de Horn strictes et ψ_{neg} la conjonction de ses clauses négatives. La formule $\forall x \forall y \psi(x,y)$ peut s'écrire comme la conjonction

$$\forall x \forall y \psi_0(x,y) \wedge \forall x \forall y \psi_{neg}(x,y).$$

La propriété du modèle minimal (corollaire 1.2.6 ci-dessous) est une conséquence directe d'une propriété caractéristique des formules de Horn [24] : l'ensemble des modèles d'une formule de Horn est *clos par intersection*.

Définition 1.2.3 — intersection de structures. Soit un mot $w \in \Sigma^n$ et deux structures $\mathtt{S} := (\langle w \rangle, R_1^{\mathtt{S}}, \ldots, R_m^{\mathtt{S}})$ et $\mathtt{S}' := (\langle w \rangle, R_1^{\mathtt{S}'}, \ldots, R_m^{\mathtt{S}'})$ de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$ qui étendent la structure mot $\langle w \rangle$. On définit l'intersection de $\mathtt{S}$ et $\mathtt{S}'$ comme la structure

$$\mathtt{S} \cap \mathtt{S}' := (\langle w \rangle, R_1^{\mathtt{S}} \cap R_1^{\mathtt{S}'}, \ldots, R_m^{\mathtt{S}} \cap R_m^{\mathtt{S}'}).$$

Autrement dit, pour tout $i \in [1,m]$ et tout $(a,b) \in [1,n]^2$, on a l'équivalence $R_i^{\mathtt{S}\cap\mathtt{S}'}(a,b) \iff R_i^{\mathtt{S}}(a,b) \wedge R_i^{\mathtt{S}'}(a,b)$.

Définition 1.2.4 — inclusion de structures. Soient deux structures $\mathtt{S} := (\langle w \rangle, R_1^{\mathtt{S}}, \ldots, R_m^{\mathtt{S}})$ et $\mathtt{S}' := (\langle w \rangle, R_1^{\mathtt{S}'}, \ldots, R_m^{\mathtt{S}'})$ étendant $\langle w \rangle$ et de même signature $\mathscr{S}_\Sigma \cup \mathbf{R}$. On dit que $\mathtt{S}$ est incluse

dans $\mathtt{S}'$, ce qu'on note $\mathtt{S} \subseteq \mathtt{S}'$, si on a $R_i^{\mathtt{S}} \subseteq R_i^{\mathtt{S}'}$, pour tout $i \in [1,m]$.

Lemme 1.2.5 — clôture par intersection des modèles d'une formule de Horn. Si des structures $\mathtt{S}$ et $\mathtt{S}'$ sont modèles d'une formule de Horn stricte $\forall x \forall y \psi_0(x,y)$ alors leur intersection $\mathtt{S} \cap \mathtt{S}'$ l'est aussi.

Démonstration. On constate d'abord que pour tout atome de calcul $\alpha(x,y) := R(x+a, y+b)$ et tout $(x_0,y_0) \in [1,n]^2$, on a les équivalences : $\mathtt{S} \cap \mathtt{S}' \models \alpha(x_0,y_0) \iff R^{\mathtt{S}\cap\mathtt{S}'}(x_0+a, y_0+b) \iff R^{\mathtt{S}}(x_0+a, y_0+b) \wedge R^{\mathtt{S}'}(x_0+a, y_0+b) \iff \mathtt{S} \models \alpha(x_0,y_0)$ et $\mathtt{S}' \models \alpha(x_0,y_0)$. Au total, on a $\mathtt{S} \cap \mathtt{S}' \models \alpha(x_0,y_0)$ *ssi* on a à la fois $\mathtt{S} \models \alpha(x_0,y_0)$ et $\mathtt{S}' \models \alpha(x_0,y_0)$.

Supposons maintenant que $\mathtt{S}$ et $\mathtt{S}'$ sont modèles de la formule de Horn stricte $\forall x \forall y \psi_0(x,y)$ et soit une clause de Horn quelconque C de ψ_0, donc de la forme $C := \bigwedge_{i=1}^{k} \alpha_i \wedge \beta \to R_h(x,y)$, où chaque α_i est un atome de calcul et β est une conjonction de littéraux d'entrée. Il suffit de montrer que $\mathtt{S} \cap \mathtt{S}'$ est modèle de la formule $\forall x \forall y C(x,y)$.

Soit un couple (x_0,y_0) qui satisfait toutes les hypothèses de l'implication C dans la structure $\mathtt{S} \cap \mathtt{S}'$, c'est-à-dire tel que $\mathtt{S} \cap \mathtt{S}' \models \bigwedge_{i=1}^{k} \alpha_i(x_0,y_0) \wedge \beta(x_0,y_0)$. On a donc à la fois $\mathtt{S} \models \bigwedge_{i=1}^{k} \alpha_i(x_0,y_0) \wedge \beta(x_0,y_0)$ et $\mathtt{S}' \models \bigwedge_{i=1}^{k} \alpha_i(x_0,y_0) \wedge \beta(x_0,y_0)$. Comme on a par hypothèse $\mathtt{S} \models C(x_0,y_0)$ et $\mathtt{S}' \models C(x_0,y_0)$, on en déduit $\mathtt{S} \models R_h(x_0,y_0)$ et $\mathtt{S}' \models R_h(x_0,y_0)$ et donc $\mathtt{S} \cap \mathtt{S}' \models R_h(x_0,y_0)$.

On a ainsi établi $\mathtt{S} \cap \mathtt{S} \models \forall x \forall y C(x,y)$.

□

Puisqu'une formule de Horn stricte a toujours au moins un modèle $\mathtt{S}$, celui où tout est vrai, défini par $R^{\mathtt{S}} := [1,n]^2$, pour tout $R \in \mathbf{R}$, on déduit du lemme 1.2.5 le corollaire suivant :

Corollaire 1.2.6 Pour tout mot $w \in \Sigma^+$, toute formule de Horn stricte $\varphi_0 := \forall x \forall y \psi_0(x,y)$ de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$ a un modèle minimal $\mathtt{S}_0$ étendant $\langle w \rangle$.

C'est l'intersection de tous les modèles $(\langle w \rangle, \mathbf{R})$ de φ_0 qui étendent $\langle w \rangle$.

Définition 1.2.5 Le modèle minimal d'une formule φ_0 étendant $\langle w \rangle$ sera aussi appelé *modèle minimal* ou *modèle naturel* de φ_0 sur $\langle w \rangle$.

Du corollaire 1.2.6, on tire la conséquence suivante :

Proposition 1.2.7 Soit $\mathtt{S}_0 := (\langle w \rangle, \mathbf{R})$ le modèle naturel d'une formule de Horn stricte $\varphi_0 := \forall x \forall y \psi_0$ sur une structure d'entrée $\langle w \rangle$ et soit ψ_{neg} une conjonction de clauses de Horn négatives. On a l'équivalence

$$\langle w \rangle \models \exists \mathbf{R} \forall x \forall y (\psi_0 \wedge \psi_{neg}) \iff \mathtt{S}_0 \models \forall x \forall y \psi_{neg}.$$

Démonstration. L'implication $\Leftarrow$ est immédiate à cause de l'hypothèse $\mathtt{S}_0 \models \forall x \forall y \psi_0$.

Supposons donc $\langle w \rangle \models \exists \mathbf{R} \forall x \forall y (\psi_0 \wedge \psi_{neg})$. Soit $\mathtt{S} := (\langle w \rangle, R_1, \ldots, R_m)$ un modèle (étendant $\langle w \rangle$) de la formule $\forall x \forall y (\psi_0 \wedge \psi_{neg})$.

En particulier, $\mathtt{S}$ est un modèle de la formule de Horn stricte $\forall x \forall y \psi_0$ et donc $\mathtt{S} \supseteq \mathtt{S}_0$. Autrement dit, pour chaque prédicat de calcul $R \in \mathbf{R}$, on a $R^{\mathtt{S}} \supseteq R^{\mathtt{S}_0}$.

$\mathtt{S}$ est modèle de la conjonction de clauses négatives $\forall x \forall y \psi_{neg}$. En conséquence, pour chaque $(x_0,y_0) \in [1,n]^2$ et chaque clause (négative) C_j de ψ_{neg}, donc de la forme $C_j := \bigwedge_{i=1}^{k} \alpha_i \wedge \beta \to \bot$, où chaque α_i est un atome de calcul et β est une conjonction de littéraux d'entrée, on a l'alternative suivante :

— soit $\langle w \rangle \models \neg \beta(x_0,y_0)$,

— soit il existe un atome de calcul α_i tel que $\mathtt{S} \models \neg\alpha_i(x_0, y_0)$, d'où $\mathtt{S}_0 \models \neg\alpha_i(x_0, y_0)$ à cause de l'inclusion $R^{\mathtt{S}} \supseteq R^{\mathtt{S}_0}$, pour chaque $R \in \mathbf{R}$, qui, par contraposition, donne l'implication : $\neg R^{\mathtt{S}}(x_0, y_0) \Rightarrow \neg R^{\mathtt{S}_0}(x_0, y_0)$.

Dans les deux cas, on conclut $\mathtt{S}_0 \models C_j(x_0, y_0)$. On a donc prouvé $\mathtt{S}_0 \models \forall x \forall y \psi_{neg}$. □

1.2.3 Un exemple de formule de Horn pour le langage des palindromes

Par la suite, on appellera simplement formule de Horn une formule de Horn du second ordre existentiel comme on l'a définie précédemment. Dans cette partie, on cherche à construire une formule de Horn $\Phi_{\mathtt{Pal}}$ de sorte à avoir l'équivalence suivante : $\langle w \rangle \models \Phi_{\mathtt{Pal}} \iff w$ est un palindrome. Précisons que la définition d'un langage par une formule de Horn ne pouvant se faire que par contradiction, grâce aux clauses négatives, la définition du langage des palindromes se fera naturellement par contradiction. Ici on va définir un prédicat binaire `nonPal` sur le domaine $[1, n]$ d'une structure $\langle w \rangle$ tel que $\mathtt{nonPal}(x, y)$ est vrai si et seulement si $w_x \dots w_y$ n'est pas un palindrome. Autrement dit, on doit caractériser les mots qui ne sont pas des palindromes. On donne ici les deux conditions possibles pour qu'un mot ne soit pas un palindrome.

1. Si $x < y$ et $w_x \neq w_y$ alors $w_x \dots w_y$ n'est pas un palindrome ;
2. Si $x + 1 < y - 1$ et $w_{x+1} \dots w_{y-1}$ n'est pas un palindrome alors $w_x \dots w_y$ n'en est pas un non plus.

La première chose à faire est de définir l'inégalité $x < y$. On introduit pour cela deux prédicats binaires $R_=$ et $R_<$ avec le sens intuitif $R_=(x, y) \iff x = y$ et $R_<(x, y) \iff x < y$. Voyons comment construire de tels prédicats.

Le prédicat $R_=$ est défini inductivement de la façon suivante :

- Si $x = 1$ et $y = 1$ alors l'atome $R_=(x, y)$ est vrai. Ceci se traduit par la clause : $x = 1 \wedge y = 1 \rightarrow R_=(x, y)$;
- Si $R_=(x-1, y-1)$ est vrai alors $R_=(x, y)$ est lui aussi vrai, ce qui se traduit par la clause : $x > 1 \wedge y > 1 \wedge R_=(x-1, y-1) \rightarrow R_=(x, y)$.

Avec $\psi_1(x, y)$ la conjonction de ces deux clauses, on a, par construction, l'équivalence suivante ; pour tout mot $w = w_1 \dots w_n$ et tout $(a, b) \in [1, n]^2$: $R_=(a, b)$ est vrai dans le modèle minimal $(\langle w \rangle, R_=)$ de la formule $\forall x \forall y \psi_1(x, y)$ si et seulement si $a = b$.

À partir du prédicat $R_=$, on peut maintenant définir le prédicat $R_<$.

- Si $R_=(x+1, y)$ est vrai et donc, par construction de $R_=$, si $x + 1 = y$ alors on a $x < y$. Ce qui se traduit par la clause : $x < n \wedge R_=(x+1, y) \rightarrow R_<(x, y)$;
- Si $R_<(x+1, y)$ est vrai alors $R_<(x, y)$ est aussi vrai. On en déduit donc la clause : $x < n \wedge R_<(x+1, y) \rightarrow R_<(x, y)$.

Soit $\psi_2(x, y)$ la conjonction des deux clauses ci-dessus ; par construction, on a l'équivalence suivante pour tout mot $w = w_1 \dots w_n$ et tout $(a, b) \in [1, n]^2$: $R_<(a, b)$ est vrai dans le modèle minimal $(\langle w \rangle, R_=, R_<)$ de la formule $\forall x \forall y \psi_1(x, y) \wedge \psi_2(x, y)$ si et seulement si $a < b$.

Maintenant que l'inégalité $x < y$ est définie via le prédicat $R_<$, il ne reste plus qu'à traduire les conditions données plus haut pour définir le prédicat `nonPal` avec le sens intuitif : $\mathtt{nonPal}(x, y) \iff w_x \dots w_y$ n'est pas un palindrome.

- La condition (1) se traduit naturellement par la clause : $R_<(x, y) \wedge Q_s(x) \wedge Q_{s'}(y) \rightarrow \mathtt{nonPal}(x, y)$, pour tous $s, s' \in \Sigma$ avec $s \neq s'$;
- La condition (2) se traduit elle par la clause suivante : $R_<(x, y) \wedge \mathtt{nonPal}(x+1, y-1) \rightarrow \mathtt{nonPal}(x, y)$.

On peut maintenant se servir du prédicat `nonPal` pour rejeter les mots qui ne sont pas des

palindromes (et donc accepter ceux qui le sont). En effet, avec $\psi_3(x,y)$ la conjonction des 2 clauses données ci-dessus et $\psi' \coloneqq \psi_1 \wedge \psi_2 \wedge \psi_3$, on a l'équivalence suivante pour tout mot $w = w_1 \dots w_n$: $\texttt{nonPal}(1,n)$ est vrai dans le modèle minimal $(\langle w \rangle, R_=, R_<, \texttt{nonPal})$ de la formule $\forall x \forall y \psi'(x,y)$ si et seulement si w n'est pas un palindrome.
On ajoute alors la clause $x = 1 \wedge y = n \wedge \texttt{nonPal}(x,y) \rightarrow \bot$ à la conjonction ψ' pour obtenir une nouvelle conjonction de clauses ψ. Ainsi, avec $\Phi_{\texttt{Pal}} \coloneqq \exists R_=, R_<, \texttt{nonPal} \forall x \forall y \psi(x,y)$ et en utilisant la proposition 1.2.7, on a bien l'équivalence : $\langle w \rangle \models \Phi_{\texttt{Pal}} \iff w$ est un palindrome.

2 Concepts de base : circuit-grille et normalisation logique

2.1 Le circuit-grille

Le circuit-grille est un modèle de calcul que l'on utilisera pour faire le lien entre les automates cellulaires et la logique. Son mode de fonctionnement et les classes de complexité qu'il permet de définir seront présentés dans cette section avant d'exposer les classes de complexité d'automates cellulaires qui leur sont équivalentes.

2.1.1 Principe et définition d'un circuit-grille

Définition informelle : Pour tout entier $n > 0$ et pour tout mot d'entrée $w = w_1 \dots w_n$ sur un alphabet Σ, le circuit-grille C_n est un réseau de $(n+1) \times (n+1)$ sites indexés par deux coordonnées $(i, j) \in [0, n]^2$. Chacun des sites de C_n dans le domaine $[1, n]^2$ se trouve dans un état q appartenant à un ensemble fini $\mathbf{S}$ d'états internes ; les sites de "bordure", à savoir les sites appartenant aux domaines $\{0\} \times [0, n]$ ou $[0, n] \times \{0\}$, se trouvent dans un état particulier noté #.

Le mot d'entrée w est placé sur la grille suivant une fonction d'entrée $\mathtt{Input}_n$. Celle-ci associe à chaque site (i, j) du domaine $[1, n]^2$ soit un caractère du mot d'entrée, soit un caractère neutre n'appartenant pas à Σ et noté \$. Un exemple d'une telle fonction est donné sur la figure 2.1b.

On note $\langle i, j \rangle$ l'état du site (i, j). Comme montré sur la figure 2.1a, l'état du site (i, j) pour $i, j > 0$ est uniquement déterminé par la fonction de transition $\mathtt{f}$ prenant en arguments l'état des "prédécesseurs" de (i, j), à savoir $(i-1, j)$ et $(j-1, i)$, ainsi que le résultat de la fonction d'entrée sur ce site :

$$\langle i, j \rangle = \mathtt{f}(\langle i-1, j \rangle, \langle i, j-1 \rangle, \mathtt{Input}_n(w, i, j)).$$

Le site (n, n) est désigné comme site de sortie : on dit qu'un mot w est accepté par le circuit-grille C_n si l'état du site (n, n) appartient à un ensemble d'états acceptants $\mathbf{A}$.

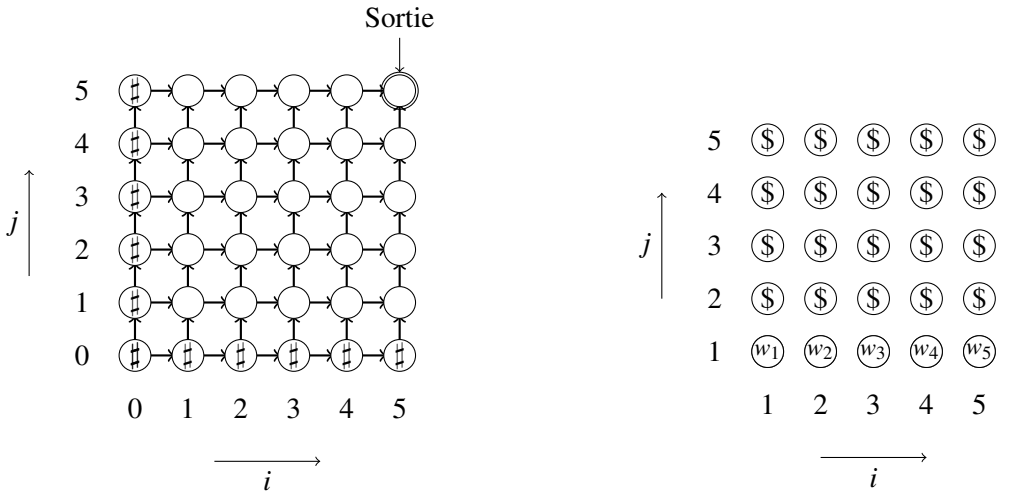
Définissons formellement un circuit-grille et son calcul.

Définition 2.1.1 — circuit-grille. Un *circuit-grille* $(C_n)_{n>0}$ est une famille de quintuplets $C_n \coloneqq (\Sigma, \mathtt{Input}_n, \mathbf{S}, \mathtt{f}, \mathbf{A})$, où :
- Σ est *l'alphabet* des mots d'entrée ;
- $\mathtt{Input}_n : \Sigma^n \times [1, n]^2 \to \Sigma \cup \{\$\}$ est la *fonction d'entrée* ;
- $\mathbf{S} \cup \{\#\}$ est l'ensemble fini des *états* ;
- $\mathtt{f} : (\mathbf{S} \cup \{\#\})^2 \times (\Sigma \cup \{\$\}) \to \mathbf{S}$ est la *fonction de transition* ;
- $\mathbf{A} \subseteq \mathbf{S}$ est l'ensemble des *états acceptants*.

On note que la fonction d'entrée $\mathtt{Input}_n$ est le seul élément de C_n qui dépend de n.

Définition 2.1.2 — calcul d'un circuit-grille. Pour un circuit-grille $C_n \coloneqq (\Sigma, \mathtt{Input}_n, \mathbf{S}, \mathtt{f}, \mathbf{A})$ et un mot $w \in \Sigma^n$, l'état, noté $\langle i, j \rangle$, d'un site $(i, j) \in [0, n]^2$ est défini inductivement par :
- $\langle i, j \rangle \coloneqq \sharp$ si c'est un site de bordure, donc pour $i = 0$ ou $j = 0$;
- $\langle i, j \rangle \coloneqq \mathtt{f}(\langle i-1, j \rangle, \langle i, j-1 \rangle, \mathtt{Input}_n(w, i, j))$ pour $(i, j) \in [1, n]^2$.

Le mot w est *accepté* par le circuit-grille C_n si $\langle n, n \rangle \in \mathbf{A}$.
Le langage reconnu par $(C_n)_{n>0}$ est l'ensemble des mots acceptés par un circuit-grille de $(C_n)_{n>0}$.



(a) Communication dans le circuit grille

(b) $\texttt{Input}(i,j) = w_i$ si $j = 1$ et \$ sinon.

FIGURE 2.1 – Fonction d'entrée et communication du circuit-grille

2.1.2 Trois modes d'entrée pour trois classes de complexité : $\texttt{GRID}_1$, $\texttt{GRID}_2$ et $\texttt{GRID}_3$

La capacité du circuit-grille à pouvoir reconnaître un langage ne dépend que de la façon dont l'entrée est placée sur la grille. Aux symétries près, il existe 4 façons de placer un mot d'entrée $w = w_1 \dots w_n$ sur la grille :

Le long d'une arrête contenant le site de sortie.

La lettre w_h est placée sur le site (n,h) (ou symétriquement, sur le site (h,n)) : $\texttt{Input}_n(w,i,j) = w_j$ si $i = n$ et \$ sinon. Ce cas n'est pas très intéressant car équivalent au calcul d'un automate fini.

Le long d'une arrête opposée au site de sortie.

La lettre w_h est placée sur le site $(1,h)$ (ou symétriquement sur le site $(h,1)$) : $\texttt{Input}_n(w,i,j) = w_j$ si $i = 1$ et \$ sinon. La classe de complexité définie par cette entrée est nommée $\texttt{GRID}_1$.

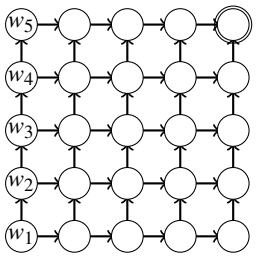


FIGURE 2.2 – $\texttt{GRID}_1$

Le long de la diagonale contenant le site de sortie.

La lettre w_h est placée sur le site (h,h) : $\texttt{Input}_n(w,i,j) = w_j$ si $i = j$ et \$ sinon. La classe de complexité définie par cette entrée est nommée $\texttt{GRID}_2$.

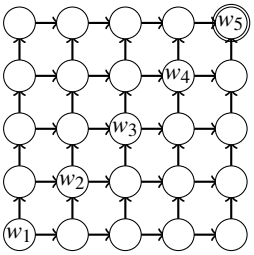


FIGURE 2.3 – $\texttt{GRID}_2$

Le long de la diagonale opposée au site de sortie.
La lettre w_h est placée sur le site $(n+1-h,h)$: $\texttt{Input}_n(w,i,j) = w_i$ si $i = n+1-j$ et \$ sinon. La classe de complexité définie par cette entrée est nommée $\texttt{GRID}_3$. Cette classe de complexité est équivalente au calcul sur la grille avec la sortie sur le site $(1,n)$, une fonction de transition $\langle i,j\rangle = \texttt{f}(\langle i+1,j\rangle, \langle i,j-1\rangle, \texttt{Input}_n(w,i,j))$ et l'entrée placée sur la diagonale : $\texttt{Input}_n(w,i,j) = w_i$ si $i = j$ et \$ sinon. Cette présentation alternative a l'avantage de définir le calcul comme une induction sur des intervalles, le site (i,j) correspondant à l'intervalle $[i,j]$.

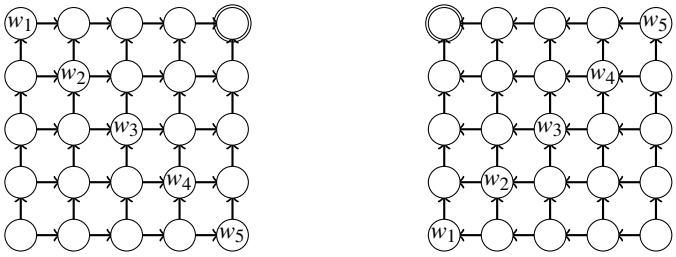


FIGURE 2.4 – $\texttt{GRID}_3$ et sa version alternative

2.2 Trois logiques pour trois modes d'entrée

On définit dans cette section, trois logiques de Horn `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN` ainsi que leurs versions inductives. On montrera dans les sections 3.1, 3.2 et 3.3 que l'ensemble des langages définissables dans les versions inductives de ces logiques `pred-ESO-IND`, `pred-dio-ESO-IND` et `incl-ESO-IND` sont exactement $\texttt{GRID}_1$, $\texttt{GRID}_2$ et $\texttt{GRID}_3$ respectivement.

2.2.1 Les classes `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN`

Définissons nos trois logiques de Horn.

La logique du prédécesseur : `pred-ESO-HORN`

Définition 2.2.1 — logique du prédécesseur. Une *formule de Horn du prédécesseur* est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :
$\mathbf{R}$ est un ensemble de prédicats binaires ;
ψ est une conjonction de clauses de Horn, de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$;
les clauses de ψ sont de la forme $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow \delta_0$ où la conclusion δ_0 est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$ soit $\bot$ (le faux) et où chaque hypothèse δ_i est :

- soit un *littéral d'entrée* de l'une des formes suivantes où a est un entier fixé $a \geq 0$:
  — $Q_s(x-a)$, $Q_s(y-a)$ pour $s \in \Sigma$,
  — $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$,
- soit un *atome de calcul* ou une *conjonction de calcul* :
  1) $S(x,y)$; 2) $S(x-a,y-b) \wedge x > a \wedge y > b$;
  et les mêmes formes avec inversion de x et y :
  1') $S(y,x)$; 2') $S(y-b,x-a) \wedge x > a \wedge y > b$;
  avec $S \in \mathbf{R}$ un *prédicat de calcul* et $a,b \geq 0$ deux entiers fixés tels que $a+b > 0$

On note pred-ESO-HORN la classe des *formules de Horn du prédécesseur* ainsi que la classe de langages qu'elles définissent.

Remarque 2.1 L'ensemble des *conjonctions de calcul* (1-2),(1'-2') de la définition 2.2.1 a été choisi aussi large que possible afin que l'induction puisse être menée à bien. En particulier, un atome $S(x-a,y-b)$ ou $S(y-b,x-a)$, avec $a,b \geq 0$, est acceptable dans une hypothèse δ_i seulement si a ou b est strictement positif (voir conjonctions 2 et 2'). Pour la même raison, l'inégalité $x > a$ (resp. $y > b$) a été ajoutée à chaque occurrence d'un terme $x-a$ (resp. $y-b$) pour $a > 0$ (resp. $b > 0$), dans les *conjonctions de calcul* 2 et 2' pour donner à ce terme son interprétation usuelle. Dans le cas où $a = 0$ (resp. $b = 0$), l'inégalité $x > 0$ (resp. $y > 0$) est toujours vraie et sera donc omise.

■ **Exemple 2.1 — Unbordered.** Le langage Unbordered [44] est l'ensemble des mots $w \in \Sigma^+$ sans bordures de taille supérieure à 1 et inférieure à $|w|$, c'est-à-dire sans préfixe propre de taille supérieure à 1 égal à un suffixe propre : $\texttt{Unbordered} = \{w \in \Sigma^+ \mid \text{il n'existe pas de } u,v,v' \text{ tels que } 1 < |u| < |w| \text{ et } w = uv = v'u\}$. Le langage Unbordered peut être défini par la formule $\Phi_{\texttt{Unbordered}} := \exists \texttt{Border} \forall x \forall y \psi$ de pred-ESO-HORN où ψ est la conjonction des clauses suivantes faisant intervenir le prédicat binaire Border dont le sens intuitif est donné par l'équivalence

$$\texttt{Border}(x,y) \iff 1 < x < y \wedge w_1 \dots w_x = w_{y-x+1} \dots w_y$$

la variable x symbolise alors la taille de la bordure comme exprimé par les clauses 1 et 2 :

1. $x = 2 \wedge y > 2 \wedge Q_s(x-1) \wedge Q_s(y-1) \wedge Q_t(x) \wedge Q_t(y) \rightarrow \texttt{Border}(x,y)$, pour tout $s,t \in \Sigma$, cette clause caractérise les mots $w_1 \dots w_y$ où $w_1 w_2 = w_{y-1} w_y$ (bordure de taille 2) ;
2. $x > 1 \wedge y > 1 \wedge \texttt{Border}(x-1,y-1) \wedge Q_s(x) \wedge Q_s(y) \rightarrow \texttt{Border}(x,y)$, pour $s \in \Sigma$, cette clause signifie que si $w_1 \dots w_{y-1}$ a une bordure de taille $x-1 > 1$ et que $w_x = w_y$ alors $1 \dots w_y$ admet une bordure de taille x ;
3. $y = n \wedge \texttt{Border}(x,y) \rightarrow \bot$, cette dernière clause qui peut se lire $\texttt{Border}(x,n) \rightarrow \bot$ rejette tous les mots admettant une bordure.

Justification : On a l'équivalence :
$\exists x > 1\ \texttt{Border}(x,n) \iff \exists x (1 < x < n \wedge w_1 \dots w_x = w_{n-x+1} \dots w_n) \iff w \notin \texttt{Unbordered}$.

On donne sur la Figure 2.5 une vision géométrique de ce processus où les axes x et y usuels sont implicites.

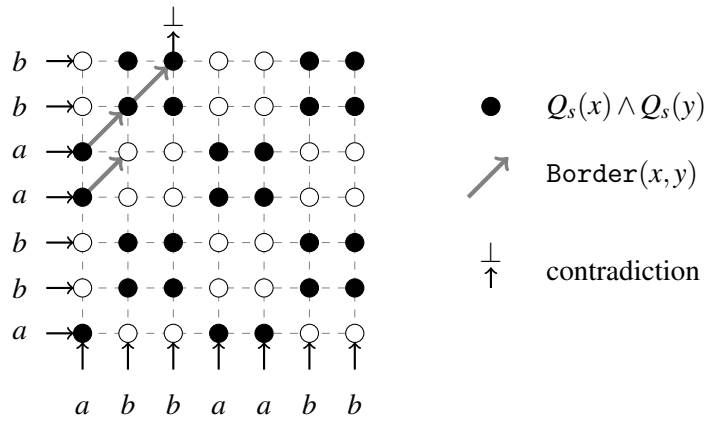


FIGURE 2.5 – Vision géométrique de l'application de $\Phi_{\texttt{Unbordered}}$ sur le mot **abb***a***abb**

■

■ **Exemple 2.2 — Produit d'entiers.** On montre dans cet exemple que la multiplication scolaire est décrite naturellement par une formule de pred-ESO-HORN. Nos logiques ne définissant que des problèmes de décision, le problème de la multiplication est formalisé par l'ensemble $\texttt{Product} := \{(a,b,p) \in (\mathbb{N}_{>0})^3 \mid a \times b = p\}$, où chaque entier est représenté en notation binaire. Soit $p = a \times b$ un produit d'entiers positifs, on note $p = \overline{p_n \dots p_1} = 2^{n-1}p_n + \dots + 2p_2 + p_1$, avec $p_n = 1$, $\texttt{length}(p) = n$, la notation binaire du produit. Dans un souci d'uniformité, on représente aussi les opérandes du produit sur n bits, en ajoutant des zéros au début si nécessaire : $a = \overline{a_n \dots a_1}$ et $b = \overline{b_n \dots b_1}$. On représente alors une entrée $(a,b,p) \in (\mathbb{N}_{>0})^3$, $\texttt{length}(p) = n$, $\texttt{length}(a) \leq n$, et $\texttt{length}(b) \leq n$, par le mot $t = t_1 \dots t_n$ sur l'alphabet $\{0,1\}^3 : t_x = (a_x, b_x, p_x)$, pour $x \in [1,n]$. Ce mot se traduit naturellement par une *structure d'entrée* de domaine $[1,n]$

$$\langle a,b,p \rangle := ([1,n]; A_0, A_1, B_0, B_1, P_0, P_1, \texttt{min}, \texttt{max}, \texttt{pred})$$

ou les prédicats d'entrée A_i, B_i, P_i, pour $i \in \{0,1\}$, sont définis par $A_i(x) \iff a_x = i$, $B_i(x) \iff b_x = i$, et $P_i(x) \iff p_x = i$, pour $x \in [1,n]$.

La multiplication scolaire :

On définit maintenant les sommes partielles successives de la multiplication, pour $1 \leq x \leq n$:

$$c(x) := \Sigma_{z=1}^{x} a \times b_z 2^{z-1}$$

avec $p = c(n)$. On a l'induction $c(1) = a \times b_1$ et $c(x) = c(x-1) + a \times b_x \times 2^{x-1}$, pour $1 < x \leq n$.

Si, pour $x \geq 1$, on définit $e(x) := a \times 2^{x-1}$ et $d(x) := b_x \times e(x)$, on peut alors écrire $c(x) = c(x-1) + d(x)$.

Prédicats de calcul :

On ut lise les 6 prédicats de calcul C_i, D_i, E_i, pour $i \in \{0,1\}$, définis sur $[1,n]$ par l'équivalence : $C_i(x,y)$ (resp. $D_i(x,y)$, $E_i(x,y)$) $\iff$ le bit y de $c(x)$ (resp. $d(x)$, $e(x)$) est i.

Clauses définissant les prédicats E_i en exprimant $e(x) = a \times 2^{x-1}$:

- clauses exprimant $e(1) = a : x = 1 \wedge A_i(y) \to E_i(x,y)$, pour $i \in \{0,1\}$;
- clauses exprimant $e(x) = 2 \times e(x-1)$, pour $x > 1$:
  - $x > 1 \wedge y = 1 \to E_0(x,y)$;
  - $x > 1 \wedge y > 1 \wedge E_i(x-1, y-1) \to E_i(x,y)$, pour $i \in \{0,1\}$.

Clauses définissant les prédicats D_i en exprimant $d(x) = b_x \times e(x)$:

— $B_0(x) \to D_0(x,y)$; $B_1(x) \wedge E_i(x,y) \to D_i(x,y)$, pour $i \in \{0,1\}$

Il reste à implémenter l'induction : $c(1) = d(1)$ et $c(x) = c(x-1) + d(x)$, pour $x > 1$. Dans ce but, on utilise l'algorithme classique d'addition.

L'algorithme d'addition `Add` :

Définissons la somme $w = \overline{w_n \dots w_1} = u + v$ de deux entiers $u = \overline{u_n \dots u_1}$ et $v = \overline{v_n \dots v_1}$, écrits sur n bits, pour $\texttt{length}(u+v) \leq n$. Soit $r_1, \dots, r_n$ la suite des retenues de cette addition, avec $r_i \in \{0,1\}$; on a les égalités

$$\texttt{Add} : u_1 + v_1 = \overline{r_1 w_1}, \text{ et } u_y + v_y + r_{y-1} = \overline{r_y w_y}, \text{ pour } y > 1, \text{ avec } r_n = 0 \text{ (car } w < 2^n).$$

Clauses définissant les prédicats C_i et les prédicats de retenues R_i :

— les clauses $x = 1 \wedge D_i(x,y) \to C_i(x,y)$, pour $i \in \{0,1\}$, expriment $c(1) = d(1)$;

— les clauses suivantes expriment $c(x) = c(x-1) + d(x)$, pour $1 < x \leq n$, par implémentation de l'algorithme d'addition `Add` en utilisant les nouveaux prédicats de calcul R_0, R_1 pour gérer les retenues, avec $h, i, j \in \{0,1\}$:

1. $x > 1 \wedge y = 1 \wedge C_h(x-1,y) \wedge D_i(x,y) \to (R_k(x,y) \wedge C_\ell(x,y))$, où $h + i = \overline{k\ell}$;
2. $x > 1 \wedge y > 1 \wedge C_h(x-1,y) \wedge D_i(x,y) \wedge R_j(x,y-1) \to (R_k(x,y) \wedge C_\ell(x,y))$, où $h + i + j = \overline{k\ell}$;
3. $y = n \wedge R_1(x,y) \to \bot$ (signifiant $c(x) < 2^n$).

Remarque : Pour faciliter la lecture, la conclusion des clauses (1) et (2) est écrite sous la forme d'une conjonction de deux atomes. Une telle clause peut être trivialement réécrite comme la conjonction de deux clauses de Horn.

Clauses vérifiant $c(n) = p$:

— $x = n \wedge C_i(x,y) \wedge \neg P_i(y) \to \bot$, pour $i \in \{0,1\}$ (signifiant $C_i(n,y) \to P_i(y)$).

Soit $\Phi_{\texttt{Product}}$ la formule $\exists C_0, C_1, D_0, D_1, E_0, E_1, R_0, R_1 \; \forall x, y \; \psi$, où ψ est la conjonction des clauses données précédemment. Par construction, on a $\Phi_{\texttt{Product}} \in \texttt{pred-ESO-HORN}$. ∎

La logique du prédécesseur avec entrée diagonale : `pred-dio-ESO-HORN`

La classe `pred-dio-ESO-HORN` représente à la fois l'ensemble des *formules de Horn du prédécesseur avec entrée diagonale* et l'ensemble des langages que ces formules définissent. Comme indiqué dans leur nom, ces formules diffèrent des *formules de Horn du prédécesseur* uniquement sur leur accès à l'entrée. Voici leur définition :

Définition 2.2.2 — logique du prédécesseur avec entrée diagonale. Une *formule de Horn du prédécesseur avec entrée diagonale* est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :

$\mathbf{R}$ est un ensemble de prédicats binaires ;

ψ est une conjonction de clauses de Horn, de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=\}$;

les clauses de ψ sont de la forme $\delta_1 \wedge \dots \wedge \delta_r \to \delta_0$ où la conclusion δ_0 est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$, soit $\bot$, et où chaque hypothèse δ_i est :

- soit une *conjonction d'entrée* : $Q_s(x-a) \wedge x = y$ avec $s \in \Sigma$ et a un entier fixé $a \geq 0$;
- soit un *littéral d'entrée* : $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et a un entier fixé $a \geq 0$;
- soit un *atome de calcul* ou une *conjonction de calcul* :
  1) $S(x,y)$; 2) $S(x-a, y-b) \wedge x > a \wedge y > b$;
  et les mêmes formes avec inversion de x et y :
  1') $S(y,x)$; 2') $S(y-b, x-a) \wedge x > a \wedge y > b$;

avec $S \in \mathbf{R}$ un *prédicat de calcul* et $a, b \geq 0$ deux entiers fixés tels que $a + b > 0$.

■ **Exemple 2.3 —** `Disj`. Le langage `Disj` est l'ensemble des mots $w = w_1 \dots w_n \in \{0,1\}^+$ de taille paire $n = 2k$, $w = x_1 \dots x_k y_1 \dots y_k$, tel que $x_i y_i \neq 11$, pour tout $i \in [1,k]$. Ce langage est un langage de référence en complexité de la communication [29]. C'est aussi un exemple de problème séparant deux classes temps-réel : $\texttt{Disj} \in \texttt{RealTime}_{\texttt{IA}} \setminus \texttt{Trellis}$ [50]. La définition inductive de ce langage en logique est basée sur une construction géométrique dont on donne un exemple sur la Figure 2.6 où les axes x et y usuels sont implicites.

Le langage `Disj` peut être défini par la formule $\Phi_{\texttt{Disj}} := \exists\{D, I^x, I^y, H, T\}\forall x \forall y \psi$ de `pred-dio-ESO-HORN` où D, I^x, I^y, H, T sont des prédicats binaires tels que

— $D(x,y)$ signifie (intuitivement) que $y = 2x$,
— $I^x(x,y)$ signifie $Q_1(x) \wedge y \geq x$,
— $I^y(x,y)$ signifie $Q_1(y) \wedge x \geq y$,
— $H(x,y)$ signifie que y est pair, $y \leq 2x$ et $Q_1(y/2)$,
— $T(x,y)$ signifie que x est pair, $x/2 < y \leq x$ et $Q_1(y - x/2)$,

et ψ est la conjonction des clauses suivantes :

— D est défini inductivement par les clauses :
$x = 1 \wedge y = 2 \to D(x,y)$, et $x > 1 \wedge y > 2 \wedge D(x-1, y-2) \to D(x,y)$;
— I^x est défini par les clauses :
$x = y \wedge Q_1(x) \to I^x(x,y)$, et $y > 1 \wedge I^x(x, y-1) \to I^x(x,y)$, de même pour I^y ;
— H est défini par les clauses :
1) $y = 2x \wedge I^x(x,y) \to H(x,y)$, 2) $x > 1 \wedge H(x-1, y) \to H(x,y)$;
— T est défini par les clauses :
3) $x = y \wedge H(x,y) \to T(x,y)$, et 4) $x > 2 \wedge y > 1 \wedge T(x-2, y-1) \to T(x,y)$.

Justification : Les définitions des prédicats D, I^x et I^y sont triviales, montrons que les définitions de H et T sont correctes. Les hypothèses de la clause 1 signifient que $D(x,y) \wedge Q_1(x) \wedge y \geq x$ ce qui implique que y est pair, $y \leq 2x$, et $Q_1(y/2)$; les hypothèses de la clause 3 signifient que $x = y$, y est pair, $y \leq 2x$, et $Q_1(y/2)$, qui ensemble impliquent que x est pair, $x/2 < y \leq x$, $y - x/2 = y/2$, et aussi $Q_1(y - x/2)$; la clause 4 se justifie par l'implication $(x-2)/2 < y - 1 \leq x - 2 \to x/2 < y \leq x$ et l'identité $(y-1) - (x-2)/2 = y - x/2$.

On donne maintenant les deux dernières clauses de ψ. La première permet de rejeter tous les mots de longueur n avec n impair : $y = n - 1 \wedge D(x,y) \to \bot$. La seconde et dernière clause effectue, pour des mots de longueur n avec n pair, la comparaison entre w_y et $w_{y-n/2}$ pour $y > n/2$ et amène à une contradiction si ces deux caractères sont égaux : $x = n \wedge I^y(x,y) \wedge T(x,y) \to \bot$. En effet si la conjonction $I^y(n,y) \wedge T(n,y)$ est vraie, alors $w_y = 1 \wedge w_{y-n/2} = 1$ par les clauses précédentes ; et donc $w \notin \texttt{Disj}$. Sur la Figure 2.6 un exemple d'induction menant à une contradiction est donné par le signal en gras $I^x \to H \to T \to I^y$.

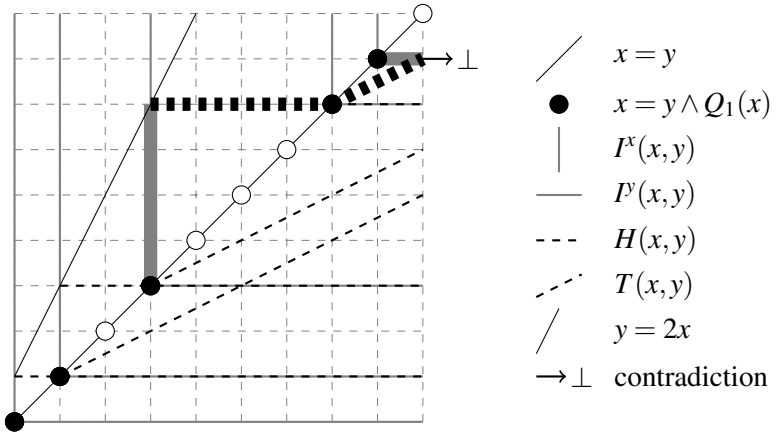


FIGURE 2.6 – Construction géométrique définie par $\Phi_{\texttt{Disj}}$ sur le mot 110**1**000**1**10 $\notin$ `Disj`

■

La logique inclusive : `incl-ESO-HORN`

Définition 2.2.3 — logique inclusive. Une formule de `incl-ESO-HORN` est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :
$\mathbf{R}$ est un ensemble de prédicats binaires ;
ψ est une conjonction de clauses de Horn, de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq, <\}$;
les clauses de ψ sont de la forme $x \leq y \wedge \delta_1 \wedge \cdots \wedge \delta_r \to \delta_0$ où la conclusion δ_0 est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$, soit $\bot$, et où chaque hypothèse δ_i est :

- soit un *littéral d'entrée* de l'une des formes :
  — $Q_s(x+a)$ ou $Q_s(y+a)$, pour $s \in \Sigma$ et $a \in \mathbb{Z}$,
  — $U(x+a)$, $\neg U(x+a)$, $U(y+a)$ ou $\neg U(y+a)$ pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \in \mathbb{Z}$;
- soit une (in)égalité $x = y$ ou $x < y$;
- soit une conjonction $S(x+a, y-b) \wedge x + a \leq y - b$, pour $S \in \mathbf{R}$ et $a, b \geq 0$ deux entiers.

■ **Exemple 2.4 —** `HWB`. Dans la section 6.4, on montrera que la logique inclusive permet de définir les langages générés par des grammaires linéaires conjonctives qui sont une extension des grammaires linéaires algébriques. Ici, on montre comment déduire d'une grammaire générant un langage linéaire algébrique une formule de `incl-ESO-HORN` définissant le complémentaire de ce langage. Notre langage exemple est Hidden Weighted Bit (`HWB`). Un mot binaire $w = w_1 \dots w_n \in \{0,1\}^+$ appartient à ce langage si le bit avec pour indice le nombre de bits à 1 de w a pour valeur 1. Par exemple, le mot $w = 0111010$ appartient à ce langage car $\sum_{i \in [1,7]} w_i = 4$ et $w_4 = 1$.
Formellement, le langage `HWB` est l'ensemble des mots $\{w_1 \dots w_n \in \{0,1\}^n \mid w_{\sum_{i \in [1,n]} w_i} = 1\}$, générés par la grammaire $S \to 1 \mid 1S \mid S0 \mid 0S1$. On déduit directement de cette grammaire les clauses suivantes où le prédicat binaire S a le sens intuitif $S(x,y) \iff w_x \dots w_y \in$ `HWB` :

— $x = y \wedge Q_1(x) \to S(x,y)$;
— $x + 1 \leq y \wedge Q_1(x) \wedge S(x+1,y) \to S(x,y)$;
— $x \leq y - 1 \wedge S(x, y-1) \wedge Q_0(y) \to S(x,y)$;
— $x + 1 \leq y - 1 \wedge Q_0(x) \wedge S(x+1, y-1) \wedge Q_1(y) \to S(x,y)$;
— $x = 1 \wedge y = n \wedge S(x,y) \to \bot$

La formule $\Phi_{\neg\texttt{HWB}} = \exists S \forall x \forall y \psi(x,y)$ où ψ est la conjonction des 5 clauses données ci-dessus satisfait l'équivalence $\langle w \rangle \models \Phi_{\neg\texttt{HWB}} \iff w \notin$ `HWB` et on a donc $\Sigma^+ \setminus$ `HWB` $\in$ `incl-ESO-HORN`. ■

Formules acycliques

Comme dans le cas propositionnel, nous donnons ci-dessous une nouvelle caractérisation du modèle minimal d'une formule de nos logiques sous la condition que cette formule satisfait un certain *critère d'acyclicité* : le modèle minimal de la formule est aussi l'unique modèle d'une certaine *conjonction d'équivalences*.

L'intérêt de cette dernière caractérisation (par conjonction d'équivalences) est d'être généralisable aux logiques *inductives* `pred-ESO-IND`, `pred-dio-ESO-IND` et `incl-ESO-IND`, que nous définirons, dans la prochaine sous-section, comme des extensions des logiques respectives `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN` quand on autorise la *négation* des atomes de calcul dans les hypothèses, tout en conservant la condition d'acyclicité.

On verra que ces logiques inductives permettent une programmation naturelle de problèmes intéressants, par exemple les langages de Dyck ou le langage de Culik, sachant que la condition d'acyclicité, elle-même naturelle, est facile à satisfaire.

On étend ici la définition 1.2.1 d'une formule acyclique dans le cadre propositionnel à nos formules de Horn décrites par les définitions 2.2.1, 2.2.2 et 2.2.3.

Définition 2.2.4 — formule de Horn acyclique. On appelle *graphe* d'une formule de Horn $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ appartenant à `pred-ESO-HORN`, `pred-dio-ESO-HORN` ou `incl-ESO-HORN` le graphe orienté, noté G_Φ ou G_ψ, dont l'ensemble des sommets est l'ensemble $\mathbf{R}$ des prédicats de calcul et l'ensemble des arcs est l'ensemble des (S,R) tels qu'il existe une clause de ψ dont l'une des hypothèses est l'atome $S(x,y)$ ou $S(y,x)$ et la conclusion est l'atome $R(x,y)$.

La formule Φ (resp. ψ) est dite *acyclique* si son graphe G_Φ (resp. G_ψ) est acyclique.

Comme pour la logique propositionnelle, la condition d'acyclicité permet d'obtenir une caractérisation alternative du modèle minimal d'une formule de Horn : c'est le point 1 de la proposition 2.2.2 ci-dessous. Pour ce faire, on se ramène au cas propositionnel à l'aide du lemme 2.2.1 ci-dessous technique mais de preuve évidente.

Lemme 2.2.1 Pour tout mot non vide $w = w_1 \dots w_n \in \Sigma^n$, toute formule de Horn stricte $\forall x \forall y \psi_0(x,y)$ et toute structure $(\langle w \rangle, \mathbf{R})$ qui étend $\langle w \rangle$, on a l'équivalence :

$$(\langle w \rangle, \mathbf{R}) \models \forall x \forall y \psi_0(x,y) \iff I_w^{\mathbf{R}} \models \bigwedge_{(a,b) \in [1,n]^2} \psi_0^w(a,b)$$

où $\psi_0^w(a,b)$ est la formule $\psi_0(a,b)$ simplifiée vis-à-vis de w : on remplace dans $\psi_0(a,b)$ les atomes $a=1$, $b=1$, $a=n$, $b=n$, $Q_s(a)$ et $Q_s(b)$ par leurs valeurs de vérité "vrai" ou "faux" dans la structure $\langle w \rangle$ et où $I_w^{\mathbf{R}}$ est le modèle minimal $(\langle w \rangle, \mathbf{R})$ de $\forall x \forall y \psi_0$ traduit comme interprétation propositionnelle sur les $R_i(a,b)$ alors considérés comme mn^2 variables propositionnelles (à trois indices i, a et b).

La proposition technique suivante généralise les résultats obtenus à la sous-section 1.2.1 pour la logique propositionnelle et s'obtient par réduction à cette logique.

Proposition 2.2.2 Soit $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ une formule acyclique de la logique `pred-ESO-HORN` (resp. `pred-dio-ESO-HORN`, `incl-ESO-HORN`) avec $\psi := \psi_0 \wedge \psi_{neg}$ où $\psi_0 := \bigwedge_{j=1}^m \bigwedge_{\alpha \in A_j} (\alpha \to R_j(x,y))$ est la conjonction des *clauses de calcul* de ψ et ψ_{neg} la conjonction des *clauses de contradiction* de ψ.

Pour un mot $w \in \Sigma^+$, soit $\mathbb{S}_0 := (\langle w \rangle, \mathbf{R})$ le modèle minimal de la formule de Horn stricte $\varphi_0 := \forall x \forall y \psi_0$ sur $\langle w \rangle$. On obtient :

1. S_0 est aussi l'unique modèle sur $\langle w\rangle$ de la conjonction d'équivalences[a] $\varphi_1 := \forall x\forall y E(\psi_0)$ où on définit $E(\psi_0) := \bigwedge_{j=1}^{m}(R_j(x,y) \leftrightarrow \bigvee_{\alpha\in A_j}\alpha)$;
2. $\langle w\rangle \models \Phi \iff S_0 \models \forall x\forall y(E(\psi_0)\wedge\psi_{neg}) \iff \langle w\rangle \models \exists\mathbf{R}\forall x\forall y(E(\psi_0)\wedge\psi_{neg})$.

a. l'unicité est conséquence de l'acyclicité du graphe G_ψ

Preuve du point 1. La preuve se fait par réduction au cas propositionnel en utilisant le corollaire 1.2.3. Pour tout mot non vide $w = w_1\dots w_n \in \Sigma^n$ et toute formule de Horn stricte $\forall x\forall y\psi_0$, on a, par le lemme 2.2.1 et avec les mêmes notations, l'équivalence :

$$\begin{array}{c}(\langle w\rangle,\mathbf{R}) \text{ est le modèle minimal de } \forall x\forall y\psi_0(x,y) \text{ sur } \langle w\rangle\\ \iff I_w^{\mathbf{R}} \text{ est le modèle minimal de } \bigwedge_{(a,b)\in[1,n]^2}\psi_0^w(a,b)(1)\end{array}$$

La formule ψ_0 étant acyclique, la formule propositionnelle ψ_0^w l'est aussi. On peut donc utiliser le corollaire 1.2.3 qui nous dit que le modèle minimal $I_w^{\mathbf{R}}$ de la formule $\bigwedge_{(a,b)\in[1,n]^2}\psi_0^w(a,b)$ est aussi l'unique modèle de la formule $E(\bigwedge_{(a,b)\in[1,n]^2}\psi_0^w(a,b))$. Chaque conjonction $\psi_0^w(a,b)$ pouvant être traitée séparément, cette formule est équivalente à $\bigwedge_{(a,b)\in[1,n]^2}E(\psi_0^w(a,b))$ avec $E(\psi_0^w(a,b)) = \bigwedge_{j=1}^{m}(R_j(a,b)\leftrightarrow\bigvee_{\alpha\in A_j}\alpha)$. Comme $I_w^{\mathbf{R}}$ est le modèle unique de $\bigwedge_{(a,b)\in[1,n]^2}E(\psi_0^w)(a,b)$, par le lemme 2.2.1, on a : $S_0 := (\langle w\rangle,\mathbf{R})$ est le modèle unique de la conjonction d'équivalence $\varphi_1 := \forall x\forall y E(\psi_0)$. □

Preuve du point 2 à partir du point 1. On a les équivalences $\langle w\rangle \models \Phi \iff S_0 \models \forall x\forall y\psi_{neg}$ (par la proposition 1.2.7) $\iff S_0 \models \forall x\forall y(E(\psi_0)\wedge\psi_{neg})$ (par le point 1) $\iff \langle w\rangle \models \exists\mathbf{R}\forall x\forall y(E(\psi_0)\wedge\psi_{neg})$ (à cause de l'unicité du modèle de $\forall x\forall y E(\psi_0)$ sur $\langle w\rangle$). □

Remarque 2.2 — sémantique "inductive". On a la même sémantique intuitive que dans le cas propositionnel : pour tout prédicat de calcul R_j et tout $(x_0,y_0)\in[1,n]^2$, l'atome $R_j(x_0,y_0)$ est vrai dans le "modèle naturel" $(\langle w\rangle,\mathbf{R})$ de la conjonction d'implications $\forall x\forall y\bigwedge_{j=1}^{m}\bigwedge_{\alpha\in A_j}(\alpha\rightarrow R_j(x,y))$ *ssi* pour au moins l'une des implications $\alpha(x,y)\rightarrow R_j(x,y)$ de conclusion $R_j(x,y)$ ($\alpha\in A_j$), l'hypothèse $\alpha(x_0,y_0)$ est "déjà" vraie dans la structure $(\langle w\rangle,\mathbf{R})$.

Ce qui revient à dire que la structure $(\langle w\rangle,\mathbf{R})$ satisfait l'équivalence $\bigwedge_{j=1}^{m}(R_j(x,y)\leftrightarrow\bigvee_{\alpha\in A_j}\alpha)$. Le modèle naturel est ainsi construit "inductivement".

Remarque 2.3 — généralisation. Par ailleurs, on s'aperçoit que cette sémantique inductive où une conjonction d'implications $\forall x\forall y\bigwedge_{j=1}^{m}\bigwedge_{\alpha\in A_j}(\alpha\rightarrow R_j(x,y))$ est interprétée par la conjonction d'équivalences $\bigwedge_{j=1}^{m}(R_j(x,y)\leftrightarrow\bigvee_{\alpha\in A_j}\alpha)$ se généralise immédiatement au cas où les conjonctions d'hypothèses α contiennent des négations d'atomes de calcul, toujours avec la condition d'acyclicité. C'est ce qui justifie les définitions des logiques inductives de la section 2.2.2 suivante.

2.2.2 Logiques inductives

On présente dans cette section, la version inductive des logiques `pred-ESO-HORN`, `pred-dio-ESO-HORN` et `incl-ESO-HORN` définies dans la section précédente. La version inductive présente, par une utilisation contrôlée de la négation, l'avantage de pouvoir définir des langages de façon plus naturelle, ce que l'on illustre au travers de multiples exemples dans les prochaines sections, tout en gardant le même pouvoir d'expression que la version Horn.

Convention 2.1 Nos logiques inductives, étant acycliques, utilisent la sémantique du modèle naturel défini par *équivalences*. C'est pourquoi les définitions 2.2.5 et 2.2.6 ci-dessous présente-

ront ces logiques (logiques du second ordre existentiel) avec, à la place de la conjonction des clauses de calcul

$$\psi_0 = \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \to R_j(x,y))$$

la conjonction des équivalences associées, notée

$$E(\psi_0) = \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$$

déjà définie au point 1 de la définition 2.2.2 mais généralisée au cas où les hypothèses des clauses de la formule peuvent être des *négations* d'atomes de calcul.

• Par la suite, pour respecter l'intuition de l'induction et en cohérence avec l'usage de nos logiques de Horn, on présentera les formules inductives définissant (programmant) des langages spécifiques (langages de Dyck, de Culik, etc.) sous la forme de la conjonction $\psi_0 \wedge \psi_{neg}$ de leurs clauses de calcul et de leurs clauses de contradiction.

• Toutefois, afin de rester dans la sémantique classique, une fois les clauses de ψ_0 et ψ_{neg} données, les formules inductives seront définies formellement en utilisant la conjonction d'équivalence $E(\psi_0)$. Toute formule inductive Φ s'écrira donc :

$$\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$$

Là où l'exemple 2.5 ci-dessous ne donne pas explicitement la conjonction d'équivalence $E(\psi_0)$, l'exemple 2.6 construit explicitement $E(\psi_0)$ à partir des clauses de ψ_0.

Définissons nos logiques inductives.

Notation 2.1 On appelle `pred-ESO-IND` (resp. `pred-dio-ESO-IND`, `incl-ESO-IND`) la classe des langages définie par la logique du même nom.

Logiques prédécesseur inductives : `pred-ESO-IND` et `pred-dio-ESO-IND`

Définition 2.2.5 Une formule $\Phi \in$ `pred-ESO-IND` (resp. $\Phi \in$ `pred-dio-ESO-IND`) est de la forme $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ où :
$\mathbf{R} := \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
$\psi_0 := \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \to R_j(x,y))$;
$E(\psi_0) := \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$;
$\psi_{neg} := \bigwedge_{\alpha \in A_0} (\alpha \to \bot)$;
$\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses sur les variables x, y,
de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$ (resp. $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=\}$).
En outre, chaque α présent dans les conjonctions ψ_0 et ψ_{neg} est une conjonction $\alpha := \delta_1 \wedge \ldots \wedge \delta_q$ où chaque hypothèse δ_i est

- soit un *littéral d'entrée* (resp. *conjonction d'entrée*) de l'une des formes suivantes où $a \geq 0$ est un entier fixé :
  — $Q_s(x-a)$, $Q_s(y-a)$ (resp. $Q_s(x-a) \wedge x = y$), pour $s \in \Sigma$,
  — $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$,
- soit une *conjonction de calcul* de l'une des formes (1-2) et (1'-2'), avec $S \in \mathbf{R}$ et des entiers fixés $a, b \geq 0$ avec $a + b > 0$:
  1) $S(x,y)$; 2) $(\neg)S(x-a, y-b) \wedge x > a \wedge y > b$;

et, en permutant x et y :
1') $S(y,x)$; 2') $(\neg)S(y-b,x-a) \wedge x > a \wedge y > b$.

De plus, la formule Φ doit respecter le critère d'acyclicité de la définition 2.2.4 : le graphe, noté G_Φ ou G_ψ, dont l'ensemble des sommets est $\mathbf{R}$ et les arcs sont les couples (S,R) tels qu'il existe une clause de ψ dont l'atome $S(x,y)$ ou $S(y,x)$ est hypothèse et l'atome $R(x,y)$ est la conclusion doit être acyclique.

■ **Exemple 2.5 — le langage `Prime`.** On cherche à définir en logique l'ensemble des nombres premiers `Prime`. Afin de rester dans le cadre des langages on associe à chaque entier positif n, sa notation unaire 1^n ; ainsi, `Prime` est le langage des mots 1^p de longueur $p \in$ `Prime`.
L'algorithme de Fischer [14] reconnaît en temps-réel sur un automate cellulaire l'ensemble des nombres premiers par un crible d'Ératosthène naïf : ce crible supprime les multiples de tous les nombres supérieurs à 1, là où le crible d'Ératosthène supprime uniquement les multiples des nombres premiers. La figure 2.7 illustre l'algorithme de Fischer sur une grille $[1,n]^2$ où n est le nombre que l'on souhaite tester. Cet exemple est le premier où on ut lise la négation des hypothèses, cet algorithme nécessite en effet de pouvoir arrêter des signaux afin de les contenir à une zone précise, ce qui, comme on le verra dans la suite de cet exemple, s'exprime très bien par un usage contrôlé de la négation.

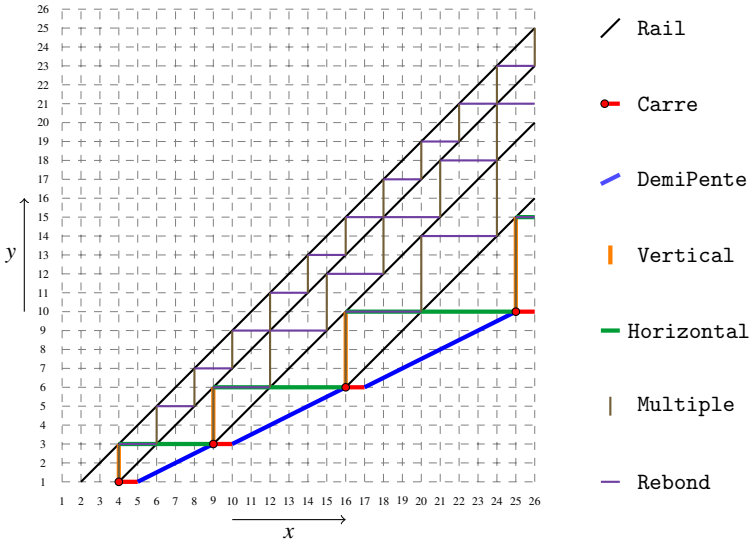


FIGURE 2.7 – Calcul de $\Phi_{\texttt{Prime}}$ représenté géométriquement

Le principe de l'algorithme de Fischer est la mise en place de couloirs (délimités par deux rails, symbolisés par le prédicat `Rail`) de taille croissante : d'abord un couloir de largeur 2, puis de largeur 3, 4 etc. A l'intérieur de ces couloirs, un signal "zig-zag", constitué d'une phase horizontale (`Rebond`) et d'une phase verticale (`Multiple`) qui alternent, permet de caractériser les multiples de la largeur du couloir. Ces signaux "zig-zag" sont le cœur du crible de Fischer : si la colonne $x = n$ contient la phase verticale du signal "zig-zag" du couloir de taille k, alors il existe un entier $k' \geq k$ tel que $n = k' \times k$ et n n'est donc pas premier.
Le rail r_k fermant le couloir de largeur k est initialisé depuis le point (x,y) tel que $x = k^2$ (premier multiple de k supprimé par ce couloir) et $y = \frac{k\times(k-1)}{2} = \Sigma_{i=1}^{k-1} i$. Ce point particulier (symbolisé par le prédicat `Carre`) est aussi le point de départ de la construction du point dont partira le prochain rail r_{k+1}.

Les prédicats utilisés par la formule $\Phi_{\texttt{Prime}}$ sont les suivants :
— le prédicat `Rail` délimitant les couloirs ;

— les prédicats `Rebond` et `Multiple` représentant respectivement les phases horizontales et verticales des signaux "zig-zag" ;
— le prédicat `Carre` caractérisant les points d'initialisation des rails et servant à la construction du prochain ;
— les prédicats `DemiPente`, `Vertical` et `Horizontal` utilisés pour construire le prochain point `Carre` à partir du précédent.

Construction des rails :

Le prédicat `Rail` est composé de "rails" parallèles ; le k-ème rail, noté r_k, a pour équation $x = y + \frac{k\times(k+1)}{2}$, pour $x \geq k^2$. Le point de départ de $rail_k$ est le site (x,y) tel que $\texttt{Carre}(x,y)$ avec $x = k^2$. Le prédicat `Rail` est donc défini par les clauses :
— $x = 2 \wedge y = 1 \rightarrow \texttt{Rail}(x,y)$ (initialisation du rail particulier r_1) ;
— $\texttt{Carre}(x,y) \rightarrow \texttt{Rail}(x,y)$ (initialisation des rails r_k avec $k > 2$) ;
— $\texttt{Rail}(x-1,y-1) \rightarrow \texttt{Rail}(x,y)$ (propagation du rail).

Construction des signaux "zig zag" :

L'ensemble des multiples de l'entier k supérieurs ou égaux à k^2 est défini par des zig zags (vertical-horizontal) entre $rail_{k-1}$ et $rail_k$. Comme on le verra dans les clauses données ci-dessous, l'utilisation de la négation dans les hypothèses permet d'exprimer naturellement le fait qu'un certain signal (ici les signaux "zig" et " zag") s'arrête lorsqu'il croise un autre signal donné (ici le signal "rail").

Les prédicats `Multiple` (le "zig" vertical) et `Rebond` (le "zag" horizontal) sont définis par les clauses suivantes :
— $\texttt{Carre}(x,y) \rightarrow \texttt{Multiple}(x,y)$ (initialisation du zig vertical avec $x = k^2$) ;
— $\texttt{Multiple}(x,y-1) \wedge \neg\texttt{Rail}(x-1,y-1) \rightarrow \texttt{Multiple}(x,y)$ (propagation verticale du zig) ;
— $\texttt{Multiple}(x,y-1) \wedge \texttt{Rail}(x-1,y-1) \rightarrow \texttt{Rebond}(x,y)$ (le zig rencontre un rail) ;
— $\texttt{Rebond}(x-1,y) \wedge \neg\texttt{Rail}(x-1,y-1) \rightarrow \texttt{Rebond}(x,y)$ (propagation horizontale du zag) ;
— $\texttt{Rebond}(x-1,y) \wedge \texttt{Rail}(x-1,y-1) \rightarrow \texttt{Multiple}(x,y)$ (le zag rencontre un rail).
Les équivalences suivantes sont alors obtenues par induction :
— $\texttt{Multiple}(x,y) \iff \exists k \geq 2, \exists k' \geq 0,\ x = k(k+k') \wedge kk' + \frac{k\times(k-1)}{2} \leq y < kk' + \frac{k\times(k+1)}{2}$;
— $\texttt{Rebond}(x,y) \iff \exists k \geq 2, \exists k' \geq 0,\ y = kk' + \frac{k\times(k+1)}{2} \wedge k(k+k') \leq x < k(k+k'+1)$.

Construction des points d'initialisation des rails :

Le premier point `Carre` est le point de départ du rail r_2 : $(2^2, \frac{2\times1}{2}) = (4,1)$. Ce premier point `Carre` est caractérisé par la clause :
— $x = 4 \wedge y = 1 \rightarrow \texttt{Carre}(x,y)$.

Soient `DemiPente`, `Vertical` et `Horizontal` les prédicats suivants (voir Figure 2.7) :
— $\texttt{DemiPente}(x,y) \iff \exists k,k',\ 0 \leq k' < k \wedge x = k^2 + 1 + 2k' \wedge y = k' + \frac{k\times(k-1)}{2}$;
— $\texttt{Vertical}(x,y) \iff \exists k,k',\ 0 < k' < k \wedge x = k^2 \wedge y = k' + \frac{k\times(k-1)}{2}$;
— $\texttt{Horizontal}(x,y) \iff \exists k \geq 2,\ k^2 \leq x \leq k^2 + 2k \wedge y = \frac{k\times(k+1)}{2}$.

Le prédicat `Carre` est défini inductivement par les prédicats auxiliaires `DemiPente`, `Vertical` et `Horizontal` :
— *Définition de* `DemiPente` *:* $\texttt{Carre}(x-1,y) \rightarrow \texttt{DemiPente}(x,y)$;
$\texttt{DemiPente}(x-2,y-1) \wedge \neg\texttt{Horizontal}(x-1,y) \rightarrow \texttt{DemiPente}(x,y)$;
— *Définition de* `Vertical` *:* $\texttt{Carre}(x,y-1) \rightarrow \texttt{Vertical}(x,y)$;
$\texttt{Vertical}(x,y-1) \wedge \neg\texttt{Rail}(x-1,y-1) \rightarrow \texttt{Vertical}(x,y)$;
— *Définition de* `Horizontal` *:* $\texttt{Vertical}(x,y-1) \wedge \texttt{Rail}(x-1,y-1) \rightarrow \texttt{Horizontal}(x,y)$;
$\texttt{Horizontal}(x-1,y) \wedge \neg\texttt{DemiPente}(x-2,y-1) \rightarrow \texttt{Horizontal}(x,y)$;

— *Définition du prochain point* `Carre` :
$\texttt{Horizontal}(x-1,y) \wedge \texttt{DemiPente}(x-2,y-1) \rightarrow \texttt{Carre}(x,y)$.

Définition de `Prime` :

Un entier $x \geq 2$ est premier *ssi* il n'est pas de la forme $x = kk'$ pour tout $k' \geq k \geq 2$, i.e., *ssi* x n'est pas un multiple de k supérieur ou égal à k^2 pour tout $k \geq 2$. On en déduit donc la clause suivante, équivalente à $\forall y\, \neg\texttt{Multiple}(n,y)$, exprimant la primalité de n :

— $x = n \wedge \texttt{Multiple}(x,y) \rightarrow \bot$.

On appelle ψ_{neg} la clause négative ci-dessus et ψ_0 la conjonction de toutes les autres clauses. Soit $\sigma_{\texttt{Prime}}$ l'ensemble des prédicats de calcul utilisés :
$\sigma_{\texttt{Prime}} := \{\texttt{Multiple}, \texttt{Rebond}, \texttt{Carre}, \texttt{Rail}, \texttt{DemiPente}, \texttt{Vertical}, \texttt{Horizontal}\}$. Soit $\Phi_{\texttt{Prime}}$ la formule $\exists\sigma_{\texttt{Prime}}\, \forall x \forall y\, E(\psi_0) \wedge \psi_{neg}$, où $E(\psi_0)$ est la conjonction d'équivalences obtenue à partir de ψ_0, en accord avec la Convention 2.1. Le graphe de la formule $\Phi_{\texttt{Prime}}$ a pour seuls arcs $(\texttt{Carre}, \texttt{Multiple})$ et $(\texttt{Carre}, \texttt{Rail})$, $\Phi_{\texttt{Prime}}$ est donc acyclique. Par construction on a $\Phi_{\texttt{Prime}} \in$ `pred-dio-ESO-IND` et $\langle 1^p \rangle \models \Phi_{\texttt{Prime}} \iff p$ est premier.

■

Logique inclusive inductive : `incl-ESO-IND`

Définition 2.2.6 Une formule $\Phi \in$ `incl-ESO-IND` est de la forme $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ où
$\mathbf{R} := \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
$\psi_0 := \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow R_j(x,y))$;
$E(\psi_0) := \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$;
$\psi_{neg} := \bigwedge_{\alpha \in A_0} (\alpha \rightarrow \bot)$;
$\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses sur les variables x, y,
de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq, <\}$.

En outre, chaque α présent dans les conjonctions ψ_0 et ψ_{neg} est une conjonction $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_q$ où chaque hypothèse δ_i est

- soit un *littéral d'entrée* $(\neg)U(x+a)$ ou $(\neg)U(y+a)$, pour $U \in \{(Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{max}\}$ et $a \in \mathbb{Z}$,
- soit une (in)égalité $x = y$ ou $x < y$,
- soit un atome de calcul $S(x,y)$ ou une conjonction de la forme

$$(\neg)S(x+a, y-b) \wedge x + a \leq y - b,$$

pour $S \in \mathbf{R}$ et $a, b \geq 0$ tels que $a + b > 0$.

De plus, le graphe, noté G_Φ ou G_ψ, dont l'ensemble des sommets est $\mathbf{R}$ et les arcs sont les couples (S,R) tels qu'il existe une clause de ψ dont $S(x,y)$ est une hypothèse et $R(x,y)$ est la conclusion, doit être acyclique.

■ **Exemple 2.6 — le langage `Dyck`.** Le langage `Dyck` est l'ensemble des mots bien parenthésés sur l'alphabet $\Sigma = \{(,)\}$. On donne dans cet exemple, une formule de `incl-ESO-IND` définissant ce langage. Cette formule utilise les deux prédicats binaires `Ouvrante` et `Fermante` dont le sens intuitif, pour tout mot $w = w_1 \ldots w_n$, est le suivant :

— $\texttt{Ouvrante}(x,y) \iff x \leq y$ et w_x est une parenthèse ouvrante qui n'est pas fermée (n'a pas de fermante qui la ferme) dans le facteur $w_x \ldots w_y$;

— $\texttt{Fermante}(x,y) \iff x \leq y$ et w_y est une parenthèse fermante qui n'est pas ouverte dans le facteur $w_x \ldots w_y$.

Définissons maintenant ces prédicats.

Pour un facteur de taille 1 (pour $w_x \dots w_y$ quand $x = y$), si le seul caractère qui compose ce facteur est une parenthèse ouvrante (resp. fermante) alors ce facteur commence (resp. finit) par une parenthèse ouvrante (resp. fermante) non fermée (resp. non ouverte) et donc $\texttt{Ouvrante}(x,y)$ (resp. $\texttt{Fermante}(x,y)$) est vrai. C'est ce qu'exprime les deux clauses :

(1) $x = y \wedge Q_(x) \rightarrow \texttt{Ouvrante}(x,y)$;
(2) $x = y \wedge Q_)(x) \rightarrow \texttt{Fermante}(x,y)$.

Pour un facteur $w_x \dots w_y$ de longueur supérieure à 1, si w_x est une parenthèse ouvrante qui n'est pas fermée dans le facteur $w_x \dots w_{y-1}$ et que w_y n'est pas une parenthèse fermante non ouverte dans $w_{x+1} \dots w_y$ alors w_x ne sera toujours pas fermée dans le facteur $w_x \dots w_y$. En effectuant un raisonnement analogue pour le cas d'une parenthèse fermante, on obtient alors les deux clauses suivantes :

(3) $x < y \wedge \texttt{Ouvrante}(x,y-1) \wedge \neg\texttt{Fermante}(x+1,y) \rightarrow \texttt{Ouvrante}(x,y)$;
(4) $x < y \wedge \neg\texttt{Ouvrante}(x,y-1) \wedge \texttt{Fermante}(x+1,y) \rightarrow \texttt{Fermante}(x,y)$.

On note ψ_0 la conjonction des clauses (1-4) définissant les prédicats `Ouvrante` et `Fermante`. On définit ensuite à partir de ψ_0 la conjonction d'équivalences $E(\psi_0)$:

$$\begin{aligned} E(\psi_0) := \big(\big(x = y \wedge Q_(x)\big) \vee (x < y \wedge \texttt{Ouvrante}(x,y-1) \wedge \neg\texttt{Fermante}(x+1,y)) \iff \texttt{Ouvrante}(x,y)\big) \\ \wedge \big(\big(x = y \wedge Q_)(x)\big) \vee (x < y \wedge \neg\texttt{Ouvrante}(x,y-1) \wedge \texttt{Fermante}(x+1,y)) \iff \texttt{Fermante}(x,y)\big) \end{aligned}$$

Un mot n'est pas bien parenthésé si et seulement si un de ses préfixes finit par une parenthèse fermante non ouverte ou si un de ses suffixes commence par une parenthèse ouvrante non fermée. C'est ce qu'exprime la conjonction ψ_{neg} des deux clauses :

(5) $x \leq y \wedge x = 1 \wedge \mathit{Fermante}(x,y) \rightarrow \bot$;
(6) $x \leq y \wedge y = n \wedge \mathit{Ouvrante}(x,y) \rightarrow \bot$.

Par construction, la formule $\Phi_{\texttt{Dyck}} := \exists\{\texttt{Ouvrante},\texttt{Fermante}\}\forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ satisfait l'équivalence : $\langle w \rangle \models \Phi_{\texttt{Dyck}} \iff w \in \texttt{Dyck}$. ∎

2.3 Programmation par la logique et normalisation : un premier exemple

On donne dans cette section un exemple d'application de notre méthode de normalisation. Cette normalisation est la principale étape pour construire un programme d'AC à partir d'une formule logique. Plus précisément, on montre comment déduire à partir de la formule $\Phi_{\texttt{Unbordered}} \in$ `pred-ESO-HORN`, étudiée en sous-section 2.2.1 (exemple 2.1) pour définir le langage `Unbordered`, un programme d'automate cellulaire qui reconnaît ce langage en temps-réel. Contrairement à ce que nous ferons au chapitre 3, on ne donnera pas ici la fonction de transition explicite de l'automate mais on dessinera son diagramme de calcul espace-temps sur un exemple.

L'idée de la normalisation est de transformer la formule initiale $\Phi_{\texttt{Unbordered}}$ pour qu'elle se rapproche du calcul d'un circuit $\texttt{GRID}_1$. La formule normalisée que l'on veut obtenir est $\exists\mathbf{R}\forall x \forall y \psi$ où $\mathbf{R} = \{R_1, \dots, R_m\}$ est un ensemble des prédicats binaires (prédicats de calcul) et ψ est une

conjonction de clauses des formes suivantes où A et B sont des parties (possiblement vides) de $[1,m]$ et $R_h \in \mathbf{R}$:

— une clause d'initialisation :
  — $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$,
  — $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j\in A} R_j(x,y-1) \rightarrow R_h(x,y)$;

— une clause de calcul :
  — $x > 1 \wedge y = 1 \wedge \bigwedge_{j\in A} R_j(x-1,y) \rightarrow R_h(x,y)$,
  — $x > 1 \wedge y > 1 \wedge \bigwedge_{j\in A} R_j(x-1,y) \wedge \bigwedge_{j\in B} R_j(x,y-1) \rightarrow R_h(x,y)$;

— une clause de contradiction :
  — $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$.

La principale difficulté lors du processus de normalisation est d'éviter l'utilisation d'atomes de calcul $R(x,y)$ dans les hypothèses des clauses de calcul. Dans cet exemple introductif, on évite l'emploi de tels atomes de façon "ad hoc" Une procédure générale pour éliminer les atomes $R(x,y)$ dans les hypothèses des clauses de calcul sera donnée et justifiée dans les prochaines sections.

0 : Avant la normalisation

Rappelons que $\Phi_{\texttt{Unbordered}}$ est la formule $\exists \texttt{Border} \forall x \forall y \psi$ où ψ est la conjonction des clauses suivantes :

0.a : $\neg\texttt{min}(x) \wedge \texttt{min}(x-1) \wedge \neg\texttt{min}(y-1) \wedge Q_s(x-1) \wedge Q_s(y-1) \wedge Q_t(x) \wedge Q_t(y) \rightarrow \texttt{Border}(x,y)$, pour tout $s,t \in \Sigma$;

0.b : $\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{Border}(x-1,y-1) \wedge Q_s(x) \wedge Q_s(y) \rightarrow \texttt{Border}(x,y)$, pour $s \in \Sigma$;

0.c : $\texttt{max}(y) \wedge \texttt{Border}(x,y) \rightarrow \bot$.

1 : Traitement de la clause de contradiction

On s'intéresse ici à la clause de contradiction (0.c) : $\texttt{max}(y) \wedge \texttt{Border}(x,y) \rightarrow \bot$. On note tout d'abord dans les hypothèses de cette clause la présence d'un atome de la forme $R(x,y)$ dont on doit se débarrasser. Comme $\texttt{Border}(x,y)$ implique $2 \leq x < y \leq n$, et donc $x < n$, cette clause peut être remplacée par la clause équivalente :

0.c' : $\texttt{max}(y) \wedge \texttt{Border}(x-1,y) \rightarrow \bot$.

On cherche maintenant à décaler cette nouvelle clause de contradiction au site (n,n). On introduit pour cela le prédicat binaire $R_\bot^{\texttt{max}(y)}$ avec la signification suivante : $R_\bot^{\texttt{max}(y)}(x,y) \iff (\texttt{max}(y) \rightarrow \bot)$. Dans ce but, on remplace la clause (0.c') par les trois clauses suivantes :

1.a : $\neg\texttt{min}(x) \wedge \texttt{Border}(x-1,y) \rightarrow R_\bot^{\texttt{max}(y)}(x,y)$ (avec la signification : $\neg\texttt{min}(x) \wedge \texttt{Border}(x-1,y) \rightarrow (\texttt{max}(y) \rightarrow \bot)$, équivalente à la clause (0.c')) ;

1.b : $\neg\texttt{min}(x) \wedge R_\bot^{\texttt{max}(y)}(x-1,y) \rightarrow R_\bot^{\texttt{max}(y)}(x,y)$ (la clause transportant $R_\bot^{\texttt{max}(y)}$ au côté de la grille où $x = n$) ;

1.c : $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R_\bot^{\texttt{max}(y)}(x,y) \rightarrow \bot$ (clause donnant le sens de $R_\bot^{\texttt{max}(y)}$).

2 : Traitement de l'entrée

Pour chaque $s \in \Sigma$, on introduit deux nouveaux prédicats binaires W_s^x et W_s^y (avec la signification $W_s^x(x,y) \iff Q_s(x)$ et $W_s^y(x,y) \iff Q_s(y)$) afin de remplacer les prédicats unaires Q_s quand x ou y est supérieur à 1. W_s^x et W_s^y sont définis par les clauses suivantes :

2.a : $\texttt{min}(y) \wedge Q_s(x) \rightarrow W_s^x(x,y)$;

2.b : $\neg\texttt{min}(y) \wedge W_s^x(x,y-1) \rightarrow W_s^x(x,y)$;

2.c : $\texttt{min}(x) \wedge Q_s(y) \rightarrow W_s^y(x,y)$;

2.d : $\neg\texttt{min}(x) \wedge W_s^y(x-1,y) \rightarrow W_s^y(x,y)$.

Ceci justifie le remplacement, pour tout $s \in \Sigma$, des atomes $Q_s(x-1)$ et $Q_s(y-1)$ par $W_s^x(x-1,y)$ et $W_s^y(x,y-1)$, respectivement, ainsi que le remplacement de $Q_s(x)$ et $Q_s(y)$ par $W_s^x(x,y-1)$ (équivalent à $W_s^x(x,y)$) et $W_s^y(x-1,y)$ (équivalent à $W_s^y(x-1,y)$), respectivement. Les clauses (0.a) et (0.b) deviennent alors respectivement :

2.e : $\neg\texttt{min}(x) \wedge \texttt{min}(x-1) \wedge \neg\texttt{min}(y-1) \wedge W_s^x(x-1,y) \wedge W_s^y(x,y-1) \wedge W_t^x(x,y-1) \wedge W_t^y(x-1,y) \rightarrow \texttt{Border}(x,y)$, for $s,t \in \Sigma$;

2.f : $\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{Border}(x-1,y-1) \wedge W_s^x(x,y-1) \wedge W_s^y(x-1,y) \rightarrow \texttt{Border}(x,y)$, for $s \in \Sigma$.

Remarque 2.4 Les substitutions effectuées respectent notre contrainte d'éviter les atomes $R(x,y)$ dans les hypothèses des clauses de calcul.

3 : Restriction des atomes de calcul à la forme $R(x-1,y)$ ou $R(x,y-1)$

A cette étape de la normalisation, le seul atome de calcul qui n'est pas d'une des formes désirées est $\texttt{Border}(x-1,y-1)$ présent dans les hypothèses de la clause (2.f). Afin de ramener cet atome à l'une des formes voulues, on introduit un nouveau prédicat binaire $\texttt{Border}^{\texttt{x}-1}$ avec la signification $\texttt{Border}^{\texttt{x}-1}(x,y) \iff (x > 1 \wedge \texttt{Border}(x-1,y))$. Ce prédicat est défini par les clauses suivantes :

3.a : $\neg\texttt{min}(x) \wedge \texttt{Border}(x-1,y) \rightarrow \texttt{Border}^{\texttt{x}-1}(x,y)$.

De cette façon, les clauses (2.f) $\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{Border}(x-1,y-1) \wedge W_s^x(x,y-1) \wedge W_s^y(x-1,y) \rightarrow \texttt{Border}(x,y)$, pour $s \in \Sigma$, sont remplacées par les clauses suivantes :

3.b : $\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{Border}^{\texttt{x}-1}(x,y-1) \wedge W_s^x(x,y-1) \wedge W_s^y(x-1,y) \rightarrow \texttt{Border}(x,y)$, pour $s \in \Sigma$.

4 : Traitement de `min` et `max`

Afin de se débarrasser des atomes $\texttt{min}(x-1)$ et $\neg\texttt{min}(y-1)$ dans les clauses (2.e) $\neg\texttt{min}(x) \wedge \texttt{min}(x-1) \wedge \neg\texttt{min}(y-1) \wedge W_s^x(x-1,y) \wedge W_s^y(x,y-1) \wedge W_t^x(x,y-1) \wedge W_t^y(x-1,y) \rightarrow \texttt{Border}(x,y)$, pour $s,t \in \Sigma$, on introduit deux nouveaux prédicats binaires $R_{\texttt{min}(x)}$ et $R_{\neg\texttt{min}(y)}$ (avec la signification $R_{\texttt{min}(x)}(x,y) \iff \texttt{min}(x)$ et $R_{\neg\texttt{min}(y)}(x,y) \iff \neg\texttt{min}(y)$) définis par les clauses suivantes :

4.a : $\texttt{min}(x) \wedge \texttt{min}(y) \rightarrow R_{\texttt{min}(x)}(x,y)$ équivalente à la conjonction des clauses d'initialisation $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow R_{\texttt{min}(x)}(x,y)$ pour tous $s \in \Sigma$;

4.b : $\neg\texttt{min}(y) \wedge R_{\texttt{min}(x)}(x,y-1) \rightarrow R_{\texttt{min}(x)}(x,y)$;

4.c : $\texttt{min}(x) \wedge \neg\texttt{min}(y) \rightarrow R_{\neg\texttt{min}(y)}(x,y)$ elle aussi équivalente à la conjonction des clauses d'initialisation $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge Q_s(y) \rightarrow R_{\neg\texttt{min}(y)}(x,y)$ pour tous $s \in \Sigma$;

4.d : $\neg\texttt{min}(x) \wedge R_{\neg\texttt{min}(y)}(x-1,y) \rightarrow R_{\neg\texttt{min}(y)}(x,y)$.

Après substitution, les clauses (2.e) deviennent, pour $s,t \in \Sigma$:

4.e : $\neg\texttt{min}(x) \wedge R_{\texttt{min}(x)}(x-1,y) \wedge \neg\texttt{min}(y) \wedge R_{\neg\texttt{min}(y)}(x,y-1) \wedge W_s^x(x-1,y) \wedge W_s^y(x,y-1) \wedge W_t^x(x,y-1) \wedge W_t^y(x-1,y) \rightarrow \texttt{Border}(x,y)$.

5 : Définir l'égalité et les inégalités

Dans la prochaine étape du processus de normalisation, on pliera le domaine $[1,n]^2$ le long de la diagonale $x = y$ sur le triangle supérieur où $x \leq y$. Dans ce but, on a besoin de prédicats exprimant l'égalité $x = y$ ($R_=$) et les inégalités $x < y$ ($R_<$) et $x \leq y$ ($R_\leq$). $R_=$ est définit conjointement avec le prédicat $R_{\texttt{pred}}$, avec la signification $R_{\texttt{pred}}(x,y) \iff x = y-1$, par les clauses suivantes :

5.a : $\texttt{min}(x) \wedge \texttt{min}(y) \rightarrow R_=(x,y)$ équivalente à la conjonction de clauses d'initialisation $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow R_=(x,y)$ pour tous $s \in \Sigma$;

5.b : $\neg\texttt{min}(y) \wedge R_=(x,y-1) \rightarrow R_{\texttt{pred}}(x,y)$;

5.c : $\neg\texttt{min}(x) \wedge R_{\texttt{pred}}(x-1,y) \rightarrow R_=(x,y)$.

Les prédicats $R_<$ et $R_\leq$ sont eux définis par les clauses :

5.d : $\neg\texttt{min}(y) \wedge R_=(x,y-1) \rightarrow R_<(x,y)$;

5.e : $\neg\texttt{min}(y) \wedge R_<(x,y-1) \rightarrow R_<(x,y)$;

5.f : $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow R_\leq(x,y)$ pour tous $s \in \Sigma$;

5.g : $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge Q_s(y) \rightarrow R_\leq(x,y)$ pour tous $s \in \Sigma$;

5.h : $\neg\texttt{min}(x) \wedge R_<(x-1,y) \rightarrow R_\leq(x,y)$.

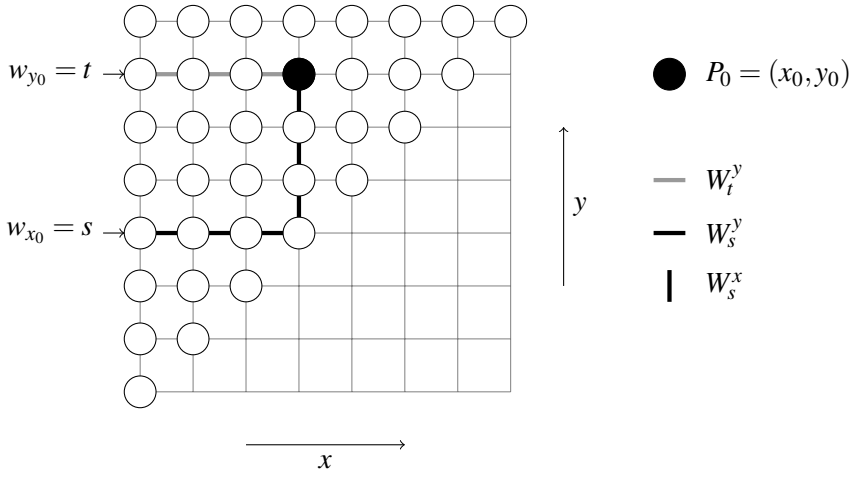


FIGURE 2.8 – Comportement de W_s^x et W_s^y après le pliage

6 : Pliage du domaine

On souhaite restreindre l'accès à l'entrée au seul axe $x = 1$ avec les seules clauses d'entrée (i.e., les seules clauses où interviennent les prédicats unaires Q_s) $\texttt{min}(x) \wedge Q_s(y) \rightarrow W_s^y(x,y)$. En d'autres mots, on cherche à se débarrasser des clauses $\texttt{min}(y) \wedge Q_s(x) \rightarrow W_s^x(x,y)$. Dans ce but, on plie le domaine carré $[1,n]^2$ le long de la diagonale $x = y$ sur le triangle supérieur $T_n = \{(x,y) \in [1,n]^2 \mid 1 \leq x \leq y \leq n\}$ par la transformation qui envoie tout point (y,x) tel que $x \leq y$ sur le point (x,y). Les prédicats d'(in)égalité $R_=, R_{\texttt{pred}}, R_<, R_\leq$ ainsi que les prédicats `Border` et $\texttt{Border}^{\texttt{x}-1}$ étant inclus dans T_n (voir Figure 2.8), les seuls prédicats de calcul agissant réellement sur les sites (x,y) tels que $x > y$ sont les prédicats de transport de l'entrée W_s^x et W_s^y, qui sont le "plié" l'un de l'autre. Les versions "pliées" sur T_n des clauses définissant W_s^x et W_s^y sont, pour tout $s \in \Sigma$, les clauses suivantes définissant W_s^y :

— $\texttt{min}(x) \wedge Q_s(y) \rightarrow W_s^y(x,y)$, cette clause est remplacée de façon équivalente par la conjonction des clauses suivantes (6.a) et (6.b), qui sont des formes voulues pour les clauses d'entrée normalisées :
6.a : $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow W_s^y(x,y)$,
6.b : $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge Q_s(y) \rightarrow W_s^y(x,y)$, et
6.c : $\neg\texttt{min}(x) \wedge W_s^y(x-1,y) \wedge R_<(x-1,y) \rightarrow W_s^y(x,y)$ (où $R_<(x-1,y)$ signifie $x \leq y$),

et les clauses suivantes définissant W_s^x :

— $x = y \wedge W_s^y(x,y) \rightarrow W_s^x(x,y)$ ("rebond" du signal horizontal W_s^y sur la diagonale qui devient le signal vertical W_s^x), cette clause peut être réécrite de manière équivalente comme la conjonction des deux clauses suivantes :
6.d : $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow W_s^x(x,y)$,
6.e : $\neg\texttt{min}(x) \wedge R_{\texttt{pred}}(x-1,y) \wedge W_s^y(x-1,y) \rightarrow W_s^x(x,y)$ (où $R_{\texttt{pred}}(x-1,y)$ signifie $x = y$), et
— $x < y \wedge \neg\texttt{min}(y) \wedge W_s^x(x,y-1) \rightarrow W_s^x(x,y)$, qui peut être réécrite
6.f : $\neg\texttt{min}(y) \wedge R_\leq(x,y-1) \wedge W_s^x(x,y-1) \rightarrow W_s^x(x,y)$.

La Figure 2.8 illustre le nouveau comportement de W_s^x et W_s^y. On notera qu'après le pliage, les prédicats de calcul W_s^x et W_s^y sont bien inclus dans le triangle supérieur T_n.

Une formule normalisée définissant le langage `Unbordered`

On récapitule ici la liste finale des clauses obtenues après les étapes 1 à 6 :

• les clauses des étapes 4 et 5 définissant les prédicats de calcul "arithmétiques" $R_{\texttt{min}(x)}$, $R_{\neg\texttt{min}(y)}$,

$R_=$, $R_{\texttt{pred}}$, $R_<$ et $R_\leq$;

• les clauses définissant les prédicats transportant l'information de l'entrée W^x et W^y_s, pour $s \in \Sigma$:

6.a : $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow W^y_s(x,y)$;
6.b : $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge Q_s(y) \rightarrow W^y_s(x,y)$;
6.c : $\neg\texttt{min}(x) \wedge W^y_s(x-1,y) \wedge R_<(x-1,y) \rightarrow W^y_s(x,y)$;
6.d : $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_s(y) \rightarrow W^x_s(x,y)$;
6.e : $\neg\texttt{min}(x) \wedge R_{\texttt{pred}}(x-1,y) \wedge W^y_s(x-1,y) \rightarrow W^x_s(x,y)$;
6.f : $\neg\texttt{min}(y) \wedge R_\leq(x,y-1) \wedge W^x_s(x,y-1) \rightarrow W^x_s(x,y)$;

• les clauses définissant les prédicats de calcul principaux Border, $\texttt{Border}^{\texttt{x}-1}$ et $R_\perp^{\texttt{max}(y)}$, i.e, décrivant le calcul principal et sa sortie :

4.e : $\neg\texttt{min}(x) \wedge R_{\texttt{min}(x)}(x-1,y) \wedge \neg\texttt{min}(y) \wedge R_{\neg\texttt{min}(y)}(x,y-1) \wedge W^x_s(x-1,y) \wedge W^y_s(x,y-1) \wedge W^x_t(x,y-1) \wedge W^y_t(x-1,y) \rightarrow \texttt{Border}(x,y)$ pour $s,t \in \Sigma$;
3.a : $\neg\texttt{min}(x) \wedge \texttt{Border}(x-1,y) \rightarrow \texttt{Border}^{\texttt{x}-1}(x,y)$;
3.b : $\neg\texttt{min}(y) \wedge \neg\texttt{min}(x) \wedge \texttt{Border}^{\texttt{x}-1}(x,y-1) \wedge W^x_s(x,y-1) \wedge W^y_s(x-1,y) \rightarrow \texttt{Border}(x,y)$, pour $s \in \Sigma$;
1.a : $\neg\texttt{min}(\texttt{x}) \wedge \texttt{Border}(x-1,y) \rightarrow R_\perp^{\texttt{max}(y)}(x,y)$;
1.b : $\neg\texttt{min}(\texttt{x}) \wedge R_\perp^{\texttt{max}(y)}(x-1,y) \rightarrow R_\perp^{\texttt{max}(y)}(x,y)$;
1.c : $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R_\perp^{\texttt{max}(y)}(x,y) \rightarrow \perp$.

On observe que chacune de ces clauses respecte les conditions de la forme normalisée :

— les clauses (4.a), (4.c), (5.a), (5.f), (5,g), (6,a), (6,b) et (6,d) sont des clauses d'entrée ;
— la clause (1.c) $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R_\perp(x,y) \rightarrow \perp$ où $R_\perp := R_\perp^{\texttt{max}(y)}$ est la clause de contradiction ;
— les autres clauses sont des clauses de calcul.

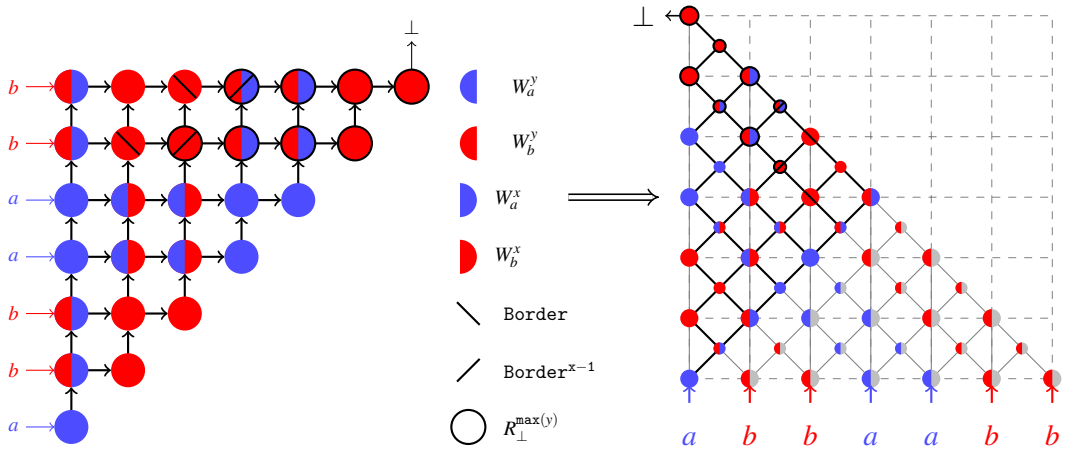
Soit ψ' la conjonction de ces clauses et soit $\mathbf{R}$ l'ensemble des prédicats de calcul, qui sont $R_{\texttt{min}(x)}, R_{\neg\texttt{min}(y)}, R_=, R_{\texttt{pred}}, R_<, R_\leq, W^y_s, W^y_s$, pour $s \in \Sigma$, Border, $\texttt{Border}^{\texttt{x}-1}$ et $R_\perp^{\texttt{max}(y)}$. La formule obtenue $\exists\mathbf{R}\forall x\forall y\psi'$ est une forme normale équivalente de la formule $\Phi_{\texttt{Unbordered}}$ in pred-ESO-HORN.

Bien que nous avons essayé de présenter la liste finale des clauses normalisées de la façon la plus naturelle possible, celle-ci est bien plus longue et moins lisible que la (courte) liste originale des clauses (0,a), (0,b) et (0,c). Alors que l'expression d'un problème dans pred-ESO-HORN (un langage haut niveau) est naturel et synthétique, le seul intérêt de sa forme normale réside dans sa capacité à être traduite aisément, d'abord en programme de circuit-grille puis en automate.

Passage à l'automate

Maintenant que la formule $\Phi_{\texttt{Unbordered}}$ a été normalisée, nous pouvons facilement en déduire un circuit-grille reconnaissant le langage Unbordered. L'ensemble des états du circuit-grille est obtenu à partir des prédicats de calcul de la formule normalisée : l'état du site (x,y) correspond à l'ensemble des prédicats de calcul vérifiés sur le point (x,y). La Figure 2.9a donne un exemple de calcul du circuit-grille sur le mot *abbaabb*. Afin d'alléger la présentation, on ne s'intéresse qu'aux prédicats de transports de l'entrée W^x_a, W^x_b, W^y_a et W^y_b ainsi qu'aux prédicats de calcul Border, $\texttt{Border}^{\texttt{x}-1}$ et $R_\perp^{\texttt{max}(y)}$ (on omet les prédicats d'(in)égalité). Par exemple, sur le mot *abbaabb*, le site $(4,7)$ se trouve dans l'état  signifiant que les atomes $W^x_b(4,7)$, $W^y_a(4,7)$, $\texttt{Border}^{\texttt{x}-1}(4,7)$ et $R_\perp^{\texttt{max}(y)}(4,7)$ sont vrais et donc que les atomes $W^x_a(4,7)$, $W^y_b(4,7)$ et $\texttt{Border}(4,7)$ sont faux.

L'idée de la transformation de ce circuit-grille vers un automate est donnée Figure 2.9.



(a) Sur le circuit grille

(b) Sur un real-time CA

FIGURE 2.9 – Calcul du mot *abbaabb* $\notin$ `Unbordered`

3 Équivalences entre logiques et automates

Ce chapitre, le plus important de la thèse et aussi le plus technique, établit les équivalences annoncées entre les logiques introduites au chapitre précédent et les trois classes de complexité temps-réel des automates cellulaires.
Pour cela, nous mettons en place, pour chacune de nos logiques, un processus de normalisation forte d'une formule : la formule normalisée finale est "assimilée" à un circuit-grille, lui-même "identifié" au diagramme espace-temps d'un automate cellulaire, tout ceci par des transformations inversibles (bijectives).

Dans ce chapitre, on prouve plusieurs équivalences entre nos classes de complexité logiques et les classes de complexité temps-réel des automates cellulaires. Ces équivalences sont données par les trois lemmes suivants :

Théorème 3.0.1 Soit L un langage non vide, alors :

$$L \in \texttt{pred-ESO-HORN} \iff L \in \texttt{pred-ESO-IND} \iff L \in \texttt{RealTime}_{\texttt{CA}}$$

Théorème 3.0.2 Soit L un langage non vide, alors :

$$L \in \texttt{pred-dio-ESO-HORN} \iff L \in \texttt{pred-dio-ESO-IND} \iff L \in \texttt{RealTime}_{\texttt{IA}}$$

Théorème 3.0.3 Soit L un langage non vide, alors :

$$L \in \texttt{incl-ESO-HORN} \iff L \in \texttt{incl-ESO-IND} \iff L \in \texttt{Trellis}$$

La preuve de chacun de ces théorèmes 3.0.1, 3.0.2, 3.0.3 est donnée dans la section de ce chapitre de numéro correspondant 3.1, 3.2, 3.3. Ces trois preuves sont réalisées par la même succession d'inclusions et d'égalités qui est résumée sur la Figure 3.1. Sur cette figure, une flèche unidirectionnelle indique une inclusion, une flèche bidirectionnelle une égalité et l'indice de chaque flèche indique dans quelle sous-section sera prouvée l'inclusion ou l'égalité correspondante.
Chacune des trois prochaines sections suit le schéma suivant :

1. Normalisation de nos formules inductives ;
2. Normalisation de nos formules de Horn, en formules de Horn acycliques, donc inductives ;
3. Seconde normalisation des formules inductives en "formules grille" (formules de Horn particulières), permettant d'établir une équivalence entre chaque logique de Horn et sa version inductive ;
4. Preuve de l'équivalence entre logique, circuit-grille et temps-réel des automates cellulaires à partir de la seconde forme normalisée (en formules grilles) des formules inductives.

3.1 Égalité des classes pred-ESO-HORN, pred-ESO-IND et $\texttt{RealTime}_{\texttt{CA}}$

On prouve dans cette section le théorème 3.0.1. Pour ce faire, on introduit trois nouvelles logiques :

1. normal-pred-ESO-HORN correspondant aux formules de pred-ESO-HORN normalisées ;
2. normal-pred-ESO-IND correspondant aux formules de pred-ESO-IND normalisées ;
3. grid-pred-ESO-HORN dont chaque formule est en en bijection directe avec un circuit-grille.

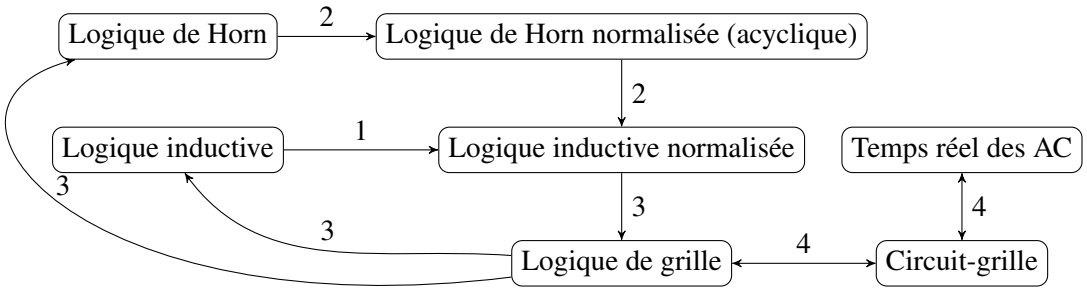


FIGURE 3.1 – Schéma des preuves de ce chapitre

La Figure 3.2 montre l'ordre dans lequel les inclusions entre ces différentes classes de complexité seront prouvées dans cette section.

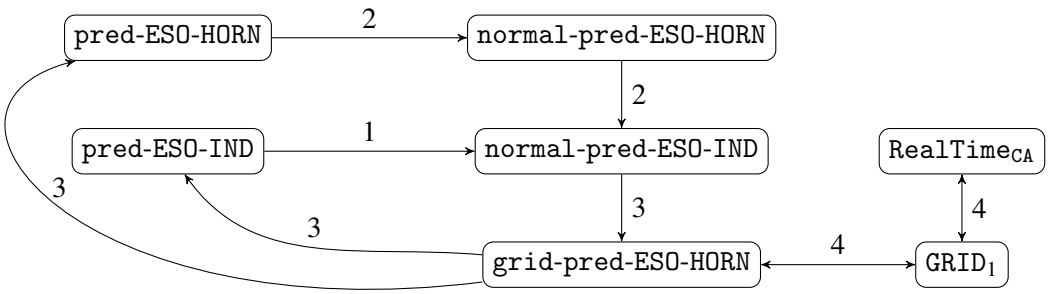


FIGURE 3.2 – Schéma général de la preuve d'égalité des classes `pred-ESO-HORN`, `pred-ESO-IND` et $\texttt{RealTime}_{\texttt{CA}}$

3.1.1 Inclusion de `pred-ESO-IND` dans `normal-pred-ESO-IND`

On définit ici la logique `normal-pred-ESO-IND` ensemble des formules inductives normalisées.

Définition 3.1.1 Une formule de `normal-pred-ESO-IND` est une formule $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ où :
$\mathbf{R} = \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
$\psi_0 := \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow R_j(x,y))$ est une conjonction de clauses de la forme suivante où $s \in \Sigma$, $R_h \in \mathbf{R}$ et A, B sont des parties (possiblement vides) de $[1,m]$:
(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$,
(b) $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j \in A} (\neg) R_j(x, y-1) \rightarrow R_h(x,y)$,
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1, y) \rightarrow R_h(x,y)$,
(d) $x > 1 \wedge y > 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1, y) \wedge \bigwedge_{j \in B} (\neg) R_j(x, y-1) \rightarrow R_h(x,y)$;
$E(\psi_0) := \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$ est la conjonction d'équivalences obtenue à partir de ψ_0 ;
ψ_{neg} est une conjonction de clauses de la forme :
(e) $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$, avec $R_h \in \mathbf{R}$.

On note ψ la conjonction $\psi_0 \wedge \psi_{neg}$ de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$.

Dans cette sous-section, on prouve la proposition suivante :

Proposition 3.1.1 Toute formule $\Phi \in$ pred-ESO-IND est équivalente à une formule $\Phi' \in$ normal-pred-ESO-IND.

Démonstration. Soit $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ une formule de pred-ESO-IND (donc acyclique), alors $\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses de la forme $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow \beta$, où chaque δ_i est :

- soit un littéral d'entrée de l'une des formes :
  - $Q_s(x-a)$, $Q_s(y-a)$, pour $s \in \Sigma$ et $a \geq 0$,
  - $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \geq 0$,
- soit un atome de calcul $S(x,y)$, $S(y,x)$, $(\neg)S(x-a, y-b)$ ou $(\neg)S(y-b, x-a)$ pour $S \in \mathbf{R}$ et $a, b \geq 0$ tels que $a+b > 0$;

et où la conclusion β est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$ soit le symbole $\bot$.

Φ est transformée en une formule normalisée équivalente $\Phi' \in$ normal-pred-ESO-IND par la succession des 8 étapes suivantes :

1. Traitement des clauses de contradiction ;
2. Traitement de l'entrée ;
3. Restriction des atomes de calcul à $(\neg)R(x-1,y)$, $(\neg)R(x,y-1)$, $R(x,y)$ et $R(y,x)$;
4. Élimination des littéraux $(\neg)\texttt{min}(x-a)$, $(\neg)\texttt{min}(y-b)$, $(\neg)\texttt{max}(x-a)$ et $(\neg)\texttt{max}(y-b)$ pour $a, b > 0$;
5. Traitement de min et max ;
6. Pliage du domaine ;
7. Suppression de max dans les clauses d'initialisation ;
8. Élimination des atomes $R(x,y)$ comme hypothèses.

À chacune de ces 8 étapes, on introduit de nouveaux prédicats de calcul qui seront ajoutés à l'ensemble $\mathbf{R}$ ainsi que de nouvelles clauses les définissant. Afin de rendre la progression de la normalisation plus simple à suivre, on indiquera après chaque étape l'ensemble des formes que peuvent prendre les clauses de la formule.
Par ailleurs, le lecteur pourra vérifier à chaque étape que les nouvelles clauses (ou les clauses transformées) introduisant les prédicats de calcul ajoutés à l'ensemble $\mathbf{R}$ préservent l'acyclicité du graphe G_ψ de la conjonction de clauses ψ ainsi modifiée.

1. Traitement des clauses de contradiction

Chaque clause de contradiction $l_1 \wedge \cdots \wedge l_k \rightarrow \bot$ peut être remplacée de manière équivalente par la conjonction des clauses $l_1 \wedge \cdots \wedge l_k \rightarrow R_\bot(x,y)$ et $R_\bot(x,y) \rightarrow \bot$ où $R_\bot$ est un nouveau prédicat de calcul. On va maintenant chercher à "déplacer" la contradiction jusqu'au point de coordonnées (n,n) comme demandé par la forme normale. Pour ce faire, on remplace la clause $R_\bot \rightarrow \bot$ par la conjonction des deux clauses de transport $\neg\texttt{min}(y) \wedge R_\bot(x, y-1) \rightarrow R_\bot(x,y)$ et $\neg\texttt{min}(x) \wedge R_\bot(x-1, y) \rightarrow R_\bot(x,y)$ et de $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R_\bot(x,y) \rightarrow \bot$.

Remarque 3.1 En prenant le prédicat $R_\bot$ ajouté à $\mathbf{R}$ "plus grand" que les autres prédicats de $\mathbf{R}$, on préserve l'acyclicité du graphe G_Φ de la formule Φ.

Après cette étape, chaque clause de la formule ψ ainsi modifiée est :
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y)$ avec $R \in \mathbf{R}$ et où chaque δ_i est :
  - soit un littéral d'entrée de l'une des formes :
    - $Q_s(x-a)$, $Q_s(y-a)$, pour $s \in \Sigma$ et $a \geq 0$,
    - $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \geq 0$,
  - soit un atome de calcul $S(x,y)$, $S(y,x)$, pour $S \in \mathbf{R}$,
  - soit une conjonction $x > a \wedge y > b \wedge (\neg)S(x-a, y-b)$ ou $x > a \wedge y > b \wedge (\neg)S(y-b, x-a)$, pour $S \in \mathbf{R}$ et $a, b \geq 0$ tels que $a+b > 0$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$ pour $R \in \mathbf{R}$.

Convention 3.1 Pour alléger la présentation des encadrés récapitulatifs qu'on donnera par la suite, on omettra de mentionner que les s de Q_s appartiennent à Σ et que les prédicats binaires $R, S, T, \ldots$ appartiennent à $\mathbf{R}$.

2. Traitement de l'entrée

L'idée de cette étape est de rendre les lettres du mot d'entrée uniquement disponibles sur les côtés $x = 1$ et $y = 1$ du carré $\{(x,y) \in [1,n]^2\}$. On introduit dans ce but de nouveaux prédicats W_s^x et W_s^y pour $s \in \Sigma$. Le rôle de ces prédicats étant de transporter l'information du mot d'entrée, on les appelle prédicats de transport. Ils sont définis inductivement par les clauses suivantes :
clauses d'initialisation : $Q_s(x) \wedge \texttt{min}(y) \to W_s^x(x,y)$ et $Q_s(y) \wedge \texttt{min}(x) \to W_s^y(x,y)$;
clauses de transport : $W_s^x(x,y-1) \wedge \neg\texttt{min}(y) \to W_s^x(x,y)$ et $W_s^y(x-1,y) \wedge \neg\texttt{min}(x) \to W_s^y(x,y)$.

Par transitivité, ces clauses impliquent les clauses $Q_s(x) \to W_s^x(x,y)$ et $Q_s(y) \to W_s^y(x,y)$. Ceci justifie le remplacement des atomes d'entrée $Q_s(x-a)$ et $Q_s(y-b)$ par les atomes de calcul respectifs $W_s^x(x-a)$ et $W_s^y(y-b)$ dans toutes les clauses, exceptées les clauses d'initialisation.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $Q_s(x) \wedge \texttt{min}(y) \to R(x,y)$ ou $Q_s(y) \wedge \texttt{min}(x) \to R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \geq 0$,
  - soit un atome de calcul $S(x,y)$, $S(y,x)$,
  - soit une conjonction $x > a \wedge y > b \wedge (\neg)S(x-a, y-b)$ ou $x > a \wedge y > b \wedge (\neg)S(y-b, x-a)$, pour $a, b \geq 0$ tels que $a+b > 0$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

3. Restriction des atomes de calcul à $(\neg)R(x-1,y)$, $(\neg)R(x,y-1)$, $R(x,y)$ et $R(y,x)$

Dans le but de restreindre chaque atome de calcul à l'une des formes demandées, on introduit de nouveaux prédicats de "décalage" R^{x-a}, R^{y-b}, $R^{x-a,y-b}$, $R^{\neg,x-a}$, $R^{\neg,y-b}$ et $R^{\neg,x-a,y-b}$ pour $a, b > 0$ et $R \in \mathbf{R}$. Ces prédicats sont définis de manière inductive de façon que l'atome $R^{x-a,y-b}(x,y)$ (par exemple) remplace la conjonction $x > a \wedge y > b \wedge R(x-a, y-b)$.
Prenons l'exemple de la clause suivante :

$$y > 2 \wedge x > 3 \wedge S(y-2,x) \wedge \neg T(x-3,y-1) \to R(x,y) \tag{3.1}$$

cette clause est remplacée par la clause sans négation sur les atomes de calcul :

$$\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge S^{x-2}(y,x) \wedge T^{\neg,x-2,y-1}(x-1,y) \to R(x,y) \tag{3.2}$$

où les prédicats S^{x-2} et S^{x-1} sont définis par les deux implications
- $x > 1 \wedge S(x-1,y) \to S^{x-1}(x,y)$
- et $x > 1 \wedge S^{x-1}(x-1,y) \to S^{x-2}(x,y)$

dont la conjonction implique $x > 2 \wedge S(x-2,y) \rightarrow S^{x-2}(x,y)$. Et donc, en permutant x, y,

$$y > 2 \wedge S(y-2,x) \rightarrow S^{x-2}(y,x). \tag{3.3}$$

et où les prédicats $T^{\neg,x-1}$, $T^{\neg,x-2}$ et $T^{\neg,x-2,y-1}$ sont définis successivement par les implications $\neg\texttt{min}(x) \wedge \neg T(x-1,y) \rightarrow T^{\neg,x-1}(x,y)$, $\neg\texttt{min}(x) \wedge T^{\neg,x-1}(x-1,y) \rightarrow T^{\neg,x-2}(x,y)$, et $\neg\texttt{min}(y) \wedge T^{\neg,x-2}(x,y-1) \rightarrow T^{\neg,x-2,y-1}(x,y)$, qui impliquent par transitivité $x > 2 \wedge y > 1 \wedge \neg T(x-2,y-1) \rightarrow T^{\neg,x-2,y-1}(x,y)$ et donc :

$$x > 3 \wedge y > 1 \wedge \neg T(x-3,y-1) \rightarrow T^{\neg,x-2,y-1}(x-1,y). \tag{3.4}$$

Justification : Comme les clauses définissant les prédicats de décalage S^{y-1}, S^{y-2}, $T^{\neg,x-1}$, $T^{\neg,x-2}$ et $T^{\neg,x-2,y-1}$ impliquent les clauses 3.3 et 3.4, et comme la conjonction des clauses 3.3, 3.4 et 3.2 implique la clause 3.1 par transitivité, la conjonction de la clause 3.2 avec les clauses définissant les prédicats de décalage implique aussi la clause 3.1 comme demandé.
Après transformation, tous les atomes de calcul dans les hypothèses des clauses sont bien d'une des formes $(\neg)R(x-1,y)$, $(\neg)R(x,y-1)$, $R(y,x)$ ou $R(x,y)$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $Q_s(x) \wedge \texttt{min}(y) \rightarrow R(x,y)$ ou $Q_s(y) \wedge \texttt{min}(x) \rightarrow R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min},\texttt{max}\}$ et $a \geq 0$,
  - soit un atome de calcul $S(x,y)$, $S(y,x)$;
  - soit une conjonction $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y)$, $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

4. Élimination des littéraux $(\neg)\texttt{min}(x-a)$, $(\neg)\texttt{min}(y-b)$, $(\neg)\texttt{max}(x-a)$ et $(\neg)\texttt{max}(y-b)$ pour $a,b > 0$

Les littéraux $\texttt{min}(x-a)$ et $\neg\texttt{min}(x-a)$ sont respectivement équivalents aux inégalités $x \leq a+1$ et $x > a+1$. De plus, pour tout entier $a \geq 1$, l'atome $\texttt{max}(x-a)$ est faux. On peut donc considérer que les seuls littéraux sur x impliquant min ou max sont $\texttt{min}(x)$, $\neg\texttt{min}(x)$, $\texttt{max}(x)$, $\neg\texttt{max}(x)$, $x \leq a$ ou $x > a$, pour un entier $a \geq 1$. On effectue le même raisonnement pour les littéraux sur y.

On définit par récurrence sur l'entier $a \geq 1$ le prédicat binaire $R^{x>a}$ (resp. les prédicats $R^{x\leq a}$, $R^{y>a}$, $R^{y\leq a}$) de façon à avoir l'équivalence intuitive $R^{x>a}(x,y) \iff x > a$ et similairement pour $x \leq a$, $y > a$ et $y \leq a$.
Le prédicat $R^{x>1}$ est défini par la clause $\neg\texttt{min}(x) \rightarrow R^{x>1}(x,y)$.
$R^{x>a}$ pour $a > 1$ est ensuite défini à partir de $R^{x>a-1}$ par la clause suivante : $\neg\texttt{min}(x) \wedge R^{x>a-1}(x-1,y) \rightarrow R^{x>a}(x,y)$. Par récurrence sur l'entier $a > 1$, ces clauses impliquent $x > a \rightarrow R^{x>a}(x,y)$. Ceci justifie le remplacement des atomes $x > a$ et $x \leq a$, pour $a \geq 1$, par les atomes $R^{x>a}(x,y)$ et $R^{x\leq a}(x,y)$, respectivement. On effectue le même traitement pour y à la place de x.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $Q_s(x) \wedge \texttt{min}(y) \rightarrow R(x,y)$ ou $Q_s(y) \wedge \texttt{min}(x) \rightarrow R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)\texttt{min}(x)$, $(\neg)\texttt{max}(x)$, $(\neg)\texttt{min}(y)$ ou $(\neg)\texttt{max}(y)$;
  - soit un atome de calcul $S(x,y)$, $S(y,x)$;
  - soit une conjonction $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y)$, $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

5. Traitement de `min` et `max`

À chaque littéral $\eta(x)$ de la forme $\texttt{min}(x)$, $\neg\texttt{min}(x)$, $\texttt{max}(x)$ ou $\neg\texttt{max}(x)$, on associe le nouveau prédicat binaire $R^{\eta(x)}$ défini par la conjonction de la clause d'initialisation $\eta(x) \wedge \texttt{min}(y) \rightarrow R^{\eta(x)}(x,y)$ et de la clause de transport $R^{\eta(x)}(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R^{\eta(x)}(x,y)$. On définit de la même manière le prédicat binaire $R^{\eta(y)}$ relatif au littéral $\eta(y) \in \{(\neg)\texttt{min}(y), (\neg)\texttt{max}(y)\}$ par la conjonction de la clause d'initialisation $\eta(y) \wedge \texttt{min}(x) \rightarrow R^{\eta(y)}(x,y)$ et de la clause de transport $R^{\eta(y)}(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R^{\eta(y)}(x,y)$.
Ceci justifie le remplacement de chaque littéral $\eta(x)$ (resp. $\eta(y)$) par l'atome $R^{\eta(x)}(x,y)$ (resp. $R^{\eta(y)}(x,y)$) correspondant dans les clauses de calcul sauf si $\eta(x)$ (resp. $\eta(y)$) est $\neg\texttt{min}(x)$ (resp. $\neg\texttt{min}(y)$) et accompagne une hypothèse $R(x-1,y)$ (resp. $R(x,y-1)$).

Une fois cette substitution effectuée, les clauses d'initialisation de la formule sont de l'une des deux formes :

— $\texttt{min}(y) \wedge \eta(x) \rightarrow R(x,y)$ pour $\eta(x) \in \{(Q_s(x))_{s\in\Sigma}, (\neg)\texttt{min}(x), (\neg)\texttt{max}(x)\}$;
— $\texttt{min}(x) \wedge \eta(y) \rightarrow R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\texttt{min}(y), (\neg)\texttt{max}(y)\}$.

Dans la prochaine étape, il sera nécessaire de différencier les clauses d'initialisation suivant l'un des trois domaines où elles agissent : 1) $x = 1$ et $y = 1$; 2) $x = 1$ et $y > 1$; 3) $x > 1$ et $y = 1$. En différenciant donc selon ces trois cas, on transforme de façon équivalente les clauses d'initialisation ci-dessus pour qu'elles soient d'une des trois formes suivantes :

1. $\texttt{min}(x) \wedge \texttt{min}(y) \wedge \eta(y) \rightarrow R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\texttt{max}(y)\}$;
2. $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \eta(y) \rightarrow R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\texttt{max}(y)\}$;
3. $\neg\texttt{min}(x) \wedge \texttt{min}(y) \wedge \eta(x) \rightarrow R(x,y)$ pour $\eta(x) \in \{(Q_s(x))_{s\in\Sigma}, (\neg)\texttt{max}(x)\}$.

Une clause $\texttt{min}(x) \wedge \texttt{max}(y) \rightarrow R(x,y)$ est, par exemple, remplacée de façon équivalente par la conjonction des deux clauses $\texttt{min}(x) \wedge \texttt{min}(y) \wedge \texttt{max}(y) \rightarrow R(x,y)$ et $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{max}(y) \rightarrow R(x,y)$. Une clause $\texttt{min}(x) \wedge \texttt{min}(y) \rightarrow R(x,y)$ peut quant à elle être remplacée de façon équivalente par la conjonction des clauses $\texttt{min}(x) \wedge \texttt{min}(y) \wedge \texttt{max}(y) \rightarrow R(x,y)$ et $\texttt{min}(x) \wedge \texttt{min}(y) \wedge \neg\texttt{max}(y) \rightarrow R(x,y)$.

Après cette étape de la normalisation, les clauses de calcul de la formule sont de la forme $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où les δ_i sont de l'une des trois formes $R(x,y)$, $(\neg)R(x-1,y) \wedge \neg\texttt{min}(x)$ ou $(\neg)R(x,y-1) \wedge \neg\texttt{min}(y)$. On peut alors supposer sans perte de généralité que chaque clause de calcul et de l'une des formes suivantes :

- $(\neg)S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R(x,y)$;
- $(\neg)S(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$;
- $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
- $S(y,x) \rightarrow R(x,y)$.

En effet, chaque clause de calcul peut être réécrite comme une conjonction de clauses de l'une de ces trois formes par l'introduction de nouveaux prédicats intermédiaires. Par exemple, la clause $R_1(x-1,y) \wedge \neg\texttt{min}(x) \wedge \neg R_2(x,y-1) \wedge \neg\texttt{min}(y) \wedge R_3(x,y) \wedge R_4(y,x) \rightarrow R_5(x,y)$ peut être réécrite comme la conjonction des six clauses suivantes en utilisant les cinq nouveaux prédicats $R_6, R_7, R_8, R_9, R_{10}$:

— $R_1(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R_6(x,y)$;
— $\neg R_2(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R_7(x,y)$;
— $R_6(x,y) \wedge R_7(x,y) \rightarrow R_8(x,y)$;
— $R_4(y,x) \rightarrow R_9(x,y)$
— $R_8(x,y) \wedge R_9(x,y) \rightarrow R_{10}(x,y)$

— $R_{10}(x,y) \wedge R_3(x,y) \to R_5(x,y)$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation :
  - $\mathtt{min}(x) \wedge \mathtt{min}(y) \wedge \eta(y) \to R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\mathtt{max}(y)\}$,
  - $\mathtt{min}(x) \wedge \neg\mathtt{min}(y) \wedge \eta(y) \to R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\mathtt{max}(y)\}$,
  - $\neg\mathtt{min}(x) \wedge \mathtt{min}(y) \wedge \eta(x) \to R(x,y)$ pour $\eta(x) \in \{(Q_s(x))_{s\in\Sigma}, (\neg)\mathtt{max}(x)\}$;
- soit une clause de calcul :
  - $(\neg)S(x-1,y) \wedge \neg\mathtt{min}(x) \to R(x,y)$;
  - $(\neg)S(x,y-1) \wedge \neg\mathtt{min}(y) \to R(x,y)$;
  - $S(x,y) \wedge T(x,y) \to R(x,y)$;
  - $S(y,x) \to R(x,y)$.
- soit une clause de contradiction $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R(x,y) \to \bot$.

6. Pliage du domaine

Dans cette étape, on prévoit de plier le domaine carré $\{(x,y) \in [1,n]^2\}$ le long de la diagonale $x = y$ sur le triangle sur-diagonal $T_n = \{(x,y) \in [1,n]^2 |\ x \leq y\}$. On a besoin pour cela de définir l'égalité $x = y$ et les inégalités $x < y$ et $x \leq y$.
On définit conjointement les prédicats $R_=$ et $R_{\mathtt{pred}}$ avec les significations $R_=(x,y) \iff x = y$ et $R_{\mathtt{pred}}(x,y) \iff x = y-1$ par les clauses suivantes :

- $\mathtt{min}(x) \wedge \mathtt{min}(y) \to R_=(x,y)$;
- $\neg\mathtt{min}(y) \wedge R_=(x,y-1) \to R_{\mathtt{pred}}(x,y)$;
- $\neg\mathtt{min}(x) \wedge R_{\mathtt{pred}}(x-1,y) \to R_=(x,y)$.

Le prédicat $R_<$ tel que $R_<(x,y) \iff x < y$ est lui défini par les deux clauses suivantes :

- $\neg\mathtt{min}(y) \wedge R_=(x,y-1) \to R_<(x,y)$;
- $\neg\mathtt{min}(y) \wedge R_<(x,y-1) \to R_<(x,y)$.

On définit de la même façon le prédicat $R_\leq$ tel que $R_\leq(x,y) \iff x \leq y$ à l'aide des deux clauses :

- $\mathtt{min}(x) \wedge \mathtt{min}(y) \to R_\leq(x,y)$;
- $\neg\mathtt{min}(x) \wedge R_<(x-1,y) \to R_\leq(x,y)$.

Notation 3.1 Afin de faciliter la lecture, on s'autorisera à écrire $x = y$, $x < y$ et $x \leq y$ à la place des atomes $R_=(x,y)$, $R_<(x,y)$ et $R_\leq(x,y)$, respectivement.

On va maintenant "plier" le domaine $[1,n]^2$ le long de la diagonale $x = y$ sur le triangle sur-diagonal $T_n = \{(x,y) \in [1,n]^2 \mid x \leq y\}$, de façon que chaque site (x_0,y_0) sous-diagonal, c'est-à-dire tel que $x_0 > y_0$, soit "plié" (= représenté) sur son point symétrique $(y_0,x_0) \in T_n$.

Un point délicat de cette transformation est de maintenir la condition d'acyclicité exigée pour les formules inductives.

Pour ce faire, on associe à chaque prédicat $R \in \mathbf{R}$ un nouveau prédicat dont la signification est la suivante : pour tout $x < y$, on a $R^{\mathtt{inv}}(x,y) \iff R(y,x)$.

Intuitivement, la structure "pliée" a pour domaine "utile" le triangle T_n et satisfait les points suivants :

— sur tout site (x_0,y_0) tel que $x_0 < y_0$, on place, pour tout $R \in \mathbf{R}$, les valeurs $R(x_0,y_0)$ et $R^{\mathtt{inv}}(y_0,x_0)$, autrement dit on superpose au site initial son site symétrique ;
— chaque site diagonal (x_0,y_0), c'est-à-dire tel que $x_0 = y_0$, reste inchangé ;
— sur les sites extérieurs à T_n, tous les autres sites, les valeurs des prédicats R et $R^{\mathtt{inv}}$ seront indifférentes : la formule "pliée" ne parlera plus de ces sites.

On va maintenant "plier" chaque clause.

Pliage des clauses d'initialisation : Les clauses d'initialisation

$\texttt{min}(x) \wedge \texttt{min}(y) \wedge \eta(y) \rightarrow R(x,y)$

et $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \eta(y) \rightarrow R(x,y)$;

pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\texttt{max}(y)\}$, s'appliquent aux points où $x = 1$ qui sont inclus dans le triangle T_n : par conséquent, celles-ci restent inchangées pendant le pliage. À l'inverse, chaque clause d'initialisation $\neg\texttt{min}(x) \wedge \texttt{min}(y) \wedge \eta(x) \rightarrow R(x,y)$ (avec $\eta(x) \in \{(Q_s(x))_{s\in\Sigma}, (\neg)\texttt{max}(x)\}$) s'appliquant à des sites (x,y) tels que $x > y$ est remplacée par sa version pliée :

- $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \eta(y) \rightarrow R^{\texttt{inv}}(x,y)$.

Pliage des clauses de calcul : Elles sont initialement des formes suivantes :

(a) $(\neg)S(x-1,y) \wedge x > 1 \rightarrow R(x,y)$
(b) $(\neg)S(x,y-1) \wedge y > 1 \rightarrow R(x,y)$
(c) $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$
(d) $S(y,x) \rightarrow R(x,y)$

Chaque clause de calcul est pliée (= transformée en une conjonction de clauses qui lui est équivalente) selon sa forme et les sites, dans T_n et/ou hors de T_n, auxquels la clause s'applique :

• Une clause (a) $(\neg)S(x-1,y) \wedge x > 1 \rightarrow R(x,y)$ est équivalente à la conjonction des trois clauses

1. $x \leq y \wedge (\neg)S(x-1,y) \wedge x > 1 \rightarrow R(x,y)$,
2. $x = y+1 \wedge (\neg)S(x-1,y) \wedge x > 1 \rightarrow R(x,y)$,
3. $x > y+1 \wedge (\neg)S(x-1,y) \wedge x > 1 \rightarrow R(x,y)$.

On laisse inchangée la clause 1 puisque les points (x,y) et $(x-1,y)$ appartiennent au domaine T_n, pour $x \leq y$.
En permutant les variables x et y, la clause 2 se réécrit de façon équivalente

$$y = x+1 \wedge (\neg)S(y-1,x) \wedge y > 1 \rightarrow R(y,x)$$

qui, à cause de l'hypothèse $x = y-1$ qui signifie que le site $(y-1,x)$ est sur la diagonale et qui implique $x < y$, donc $(y,x) \notin T_n$, est transformée en la clause équivalente

$$2')\ x = y-1 \wedge (\neg)S(x,y-1) \wedge y > 1 \rightarrow R^{\texttt{inv}}(x,y)$$

Toujours en permutant les variables x et y, la clause 3 se réécrit de façon équivalente

$$y > x+1 \wedge (\neg)S(y-1,x) \wedge y > 1 \rightarrow R(y,x)$$

qui, à cause de l'hypothèse $x < y-1$, est transformée en la clause équivalente

$$3')\ x < y-1 \wedge (\neg)S^{\texttt{inv}}(x,y-1) \wedge y > 1 \rightarrow R^{\texttt{inv}}(x,y)$$

On rappelle que les expressions $x = y-1$ et $x < y-1$ des clauses 2' et 3' sont des notations intuitives pour les atomes de calculs respectifs $R_{=}(x,y-1)$ et $R_{<}(x,y-1)$.
Cela justifie le remplacement de la clause (a) par la conjonction des clauses 1, 2' et 3'.

• Une clause (b) $(\neg)S(x,y-1) \wedge y > 1 \rightarrow R(x,y)$ est équivalente à la conjonction des trois clauses

1. $x < y \wedge (\neg)S(x,y-1) \wedge y > 1 \rightarrow R(x,y)$,
2. $x = y \wedge (\neg)S(x,y-1) \wedge y > 1 \rightarrow R(x,y)$,
3. $x > y \wedge (\neg)S(x,y-1) \wedge y > 1 \rightarrow R(x,y)$.

On laisse inchangée la clause 1 puisque les points (x,y) et $(x,y-1)$ appartiennent au domaine T_n, pour $x < y$.
En permutant les variables x et y, la clause 2 se réécrit de façon équivalente $x = y \wedge (\neg)S(y,x-1) \wedge x > 1 \rightarrow R(x,y)$. Puisqu'on a $(y,x-1) \notin T_n$ pour $x = y$, cette clause est transformée en la clause équivalente

$$2') \; x = y \wedge (\neg)S^{\texttt{inv}}(x-1,y) \wedge x > 1 \rightarrow R(x,y)$$

Toujours en permutant les variables x et y, la clause 3 se réécrit de façon équivalente $x < y \wedge (\neg)S(y,x-1) \wedge x > 1 \rightarrow R(y,x)$. Puisqu'on a $(y,x) \notin T_n$ et $(y,x-1) \notin T_n$ pour $x < y$, cette clause est transformée en la clause équivalente

$$3') \; x < y-1 \wedge (\neg)S^{\texttt{inv}}(x-1,y) \wedge y > 1 \rightarrow R^{\texttt{inv}}(x,y)$$

Cela justifie le remplacement de la clause (b) par la conjonction des clauses 1, 2' et 3' ci-dessus.

• Une clause (c) $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$ est équivalente à la conjonction des deux clauses $x \leq y \wedge S(x,y) \wedge T(x,y) \rightarrow R(x,y)$ et $x > y \wedge S^{\texttt{inv}}(y,x) \wedge T^{\texttt{inv}}(y,x) \rightarrow R^{\texttt{inv}}(y,x)$, celle-ci remplacée (en permutant x et y) par la clause équivalente $x < y \wedge S^{\texttt{inv}}(x,y) \wedge T^{\texttt{inv}}(x,y) \rightarrow R^{\texttt{inv}}(x,y)$.

• Une clause (d) $S(y,x) \rightarrow R(x,y)$ est équivalente à la conjonction des trois clauses $x = y \wedge S(x,y) \rightarrow R(x,y)$, $x < y \wedge S(y,x) \rightarrow R(x,y)$, celle-ci remplacée par $x < y \wedge S^{\texttt{inv}}(x,y) \rightarrow R(x,y)$, et $x > y \wedge S(y,x) \rightarrow R(x,y)$, elle-même remplacée par $x < y \wedge S(x,y) \rightarrow R^{\texttt{inv}}(x,y)$.

Point essentiel, la condition d'acyclicité pour les clauses de calcul initiales (a-d) qui dit que S et T sont inférieurs à R s'étend à l'exigence demandée pour les clauses obtenues que S, $S^{\texttt{inv}}$, T et $T^{\texttt{inv}}$ sont tous inférieurs à R et à $R^{\texttt{inv}}$.

Pliage des clauses de contradiction : Les clauses de contradiction s'appliquant au point (n,n), inclus dans T_n, elles restent inchangées lors du pliage.

En introduisant de nouveaux prédicats intermédiaires, on peut supposer que chaque clause de calcul est de l'une des formes suivantes :

- $(\neg)S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R(x,y)$;
- $(\neg)S(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$;
- $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$.

Par exemple, en introduisant le prédicat intermédiaire U, la clause ci-dessus $R_{\leq}(x,y) \wedge S(x,y) \wedge T(x,y) \rightarrow R(x,y)$ peut être réécrite comme la conjonction des deux clauses :

— $R_{\leq}(x,y) \wedge S(x,y) \rightarrow U(x,y)$;
— $U(x,y) \wedge T(x,y) \rightarrow R(x,y)$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation :
  - $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge \eta(y) \rightarrow R(x,y)$ pour $\eta(y) \in \{(Q_s(y))_{s\in\Sigma}, (\neg)\texttt{max}(y)\}$;
- soit une clause de calcul :
  - $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y) \rightarrow R(x,y)$;
  - $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$;
  - $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
- soit une clause de contradiction :
  - $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

7. Suppression de max dans les clauses d'initialisation

On utilise dans cette étape une méthode de *calcul sous hypothèses*. Cette méthode consiste à considérer en parallèle pour chaque point (x,y) le cas où une hypothèse (ici $\texttt{max}(y)$) est vraie et le cas opposé où c'est sa négation qui est vraie. Dans ce but, chaque prédicat de calcul $R \in \mathbf{R}$ est dupliqué en deux nouveaux prédicats $R^y_{\leftarrow \texttt{max}}$ et $R^y_{\leftarrow \neg\texttt{max}}$. Intuitivement, l'atome $R^y_{\leftarrow \texttt{max}}(x,y)$ (resp. $R^y_{\leftarrow \neg\texttt{max}}(x,y)$) exprime l'implication $\texttt{max}(y) \rightarrow R(x,y)$ (resp. $\neg\texttt{max}(y) \rightarrow R(x,y)$). On modifie les clauses de la formule en utilisant les principes suivant : la clause $A \wedge B \rightarrow C$ est équivalente à la clause $A \rightarrow (B \rightarrow C)$; de même, la clause $A \rightarrow B$ est équivalente, pour tout M, à la conjonction des clauses $A \rightarrow (M \rightarrow B)$ et $A \rightarrow (\neg M \rightarrow B)$ ou encore à la conjonction des clauses $(M \rightarrow A) \rightarrow (M \rightarrow B)$ et $(\neg M \rightarrow A) \rightarrow (\neg M \rightarrow B)$.

Transformation des clauses d'initialisation : La clause d'initialisation

— $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge \texttt{max}(y) \rightarrow R(x,y)$

est équivalente à la clause

— $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \rightarrow (\texttt{max}(y) \rightarrow R(x,y))$

qui se réécrit

— $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \rightarrow R^y_{\leftarrow \texttt{max}}(x,y)$

en accord avec la sémantique voulue pour le prédicat $R^y_{\leftarrow \neg\texttt{max}}$.

De la même façon, chaque clause $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge \neg\texttt{max}(y) \rightarrow R(x,y)$ est réécrite $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \rightarrow R^y_{\leftarrow \neg\texttt{max}}(x,y)$.

Chaque clause d'initialisation

— $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge Q_s(y) \rightarrow R(x,y)$

est remplacée par la conjonction des deux clauses :

$\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge Q_s(y) \rightarrow R^y_{\leftarrow \texttt{max}}(x,y)$

et $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge Q_s(y) \rightarrow R^y_{\leftarrow \neg\texttt{max}}(x,y)$.

Transformation des clauses de calcul sans négation d'atome de calcul : On décrit ici comment transformer chacune des trois formes de clauses de calcul sans négation pour respecter la sémantique des prédicats $R^y_{\leftarrow \texttt{max}}$ et $R^y_{\leftarrow \neg\texttt{max}}$.

- Chaque clause $S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R(x,y)$ est remplacée par la conjonction des deux clauses suivantes :
  1. $S^y_{\leftarrow \texttt{max}}(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R^y_{\leftarrow \texttt{max}}(x,y)$ et
  2. $S^y_{\leftarrow \neg\texttt{max}}(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R^y_{\leftarrow \neg\texttt{max}}(x,y)$.
- Chaque clause $S(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$ est "équivalente" à $S^y_{\leftarrow \neg\texttt{max}}(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$ car l'hypothèse $\neg\texttt{max}(y-1)$ est toujours vraie. Par conséquent, chacune de ces clauses est remplacée par la conjonction des deux clauses suivantes :

  $S^y_{\leftarrow \neg\texttt{max}}(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R^y_{\leftarrow \texttt{max}}(x,y)$, et

  $S^y_{\leftarrow \neg\texttt{max}}(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R^y_{\leftarrow \neg\texttt{max}}(x,y)$.
- Chaque clause $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$ est remplacée par la conjonction des clauses :
  1. $S^y_{\leftarrow \texttt{max}}(x,y) \wedge T^y_{\leftarrow \texttt{max}}(x,y) \rightarrow R^y_{\leftarrow \texttt{max}}(x,y)$ et
  2. $S^y_{\leftarrow \neg\texttt{max}}(x,y) \wedge T^y_{\leftarrow \neg\texttt{max}}(x,y) \rightarrow R^y_{\leftarrow \neg\texttt{max}}(x,y)$.

Transformation des clauses de calcul avec négation d'atome de calcul : Chaque clause de la forme

$$\text{(a)}\ \neg S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R(x,y)$$

est équivalente à la conjonction des deux clauses :

- $\texttt{max}(y) \wedge \neg S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow (\texttt{max}(y) \rightarrow R(x,y))$, et
- $\neg\texttt{max}(y) \wedge \neg S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow (\neg\texttt{max}(y) \rightarrow R(x,y))$,

qui sont équivalentes à

1) $\neg(\texttt{max}(y) \rightarrow S(x-1,y)) \wedge \neg\texttt{min}(x) \rightarrow (\texttt{max}(y) \rightarrow R(x,y))$, et

2) $\neg(\neg\mathtt{max}(y) \to S(x-1,y)) \wedge \neg\mathtt{min}(x) \to (\neg\mathtt{max}(y) \to R(x,y))$, respectivement.

Au vu de la sémantique des prédicats $S^y_{\leftarrow\mathtt{max}}(x,y)$, $R^y_{\leftarrow\mathtt{max}}$, $S^y_{\leftarrow\neg\mathtt{max}}$ et $R^y_{\leftarrow\neg\mathtt{max}}$, on peut transformer les clauses (1) et (2) en leurs formes "équivalentes"

1') $\neg S^y_{\leftarrow\mathtt{max}}(x-1,y) \wedge \neg\mathtt{min}(x) \to R^y_{\leftarrow\mathtt{max}}(x,y)$, et

2') $\neg S^y_{\leftarrow\neg\mathtt{max}}(x-1,y) \wedge \neg\mathtt{min}(x) \to R^y_{\leftarrow\neg\mathtt{max}}(x,y)$, respectivement.

Cela justifie le remplacement de la clause (a) ci-dessus par la conjonction des clauses (1') et (2').

De la même façon, chaque clause $\neg S(x,y-1) \wedge \neg\mathtt{min}(y) \to R(x,y)$ est équivalente à $\neg\mathtt{max}(y-1) \wedge \neg S(x,y-1) \wedge \neg\mathtt{min}(y) \to R(x,y)$ et donc à $\neg S^y_{\leftarrow\neg\mathtt{max}}(x,y-1) \wedge \neg\mathtt{min}(y) \to R(x,y)$. Ce qui justifie le remplacement de cette clause par la conjonction des clauses suivantes

1. $\neg S^y_{\leftarrow\neg\mathtt{max}}(x,y-1) \wedge \neg\mathtt{min}(y) \to R^y_{\leftarrow\mathtt{max}}(x,y)$, et
2. $\neg S^y_{\leftarrow\neg\mathtt{max}}(x,y-1) \wedge \neg\mathtt{min}(y) \to R^y_{\leftarrow\neg\mathtt{max}}(x,y)$.

Traitement des clauses de contradiction : Clairement, une clause de contradiction $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R(x,y) \to \bot$ est équivalente à la clause $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge (\mathtt{max}(y) \to R(x,y)) \to \bot$ qui est réécrite $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R^y_{\leftarrow\mathtt{max}}(x,y) \to \bot$, qui est bien de la forme demandée pour les clauses de contradiction.

Après cette étape, chaque clause de la formule ψ obtenue est :
- soit une clause d'initialisation :
  - $\mathtt{min}(x) \wedge (\neg)\mathtt{min}(y) \wedge Q_s(y) \to R(x,y)$;
- soit une clause de calcul :
  - $\neg\mathtt{min}(x) \wedge (\neg)S(x-1,y) \to R(x,y)$;
  - $\neg\mathtt{min}(y) \wedge (\neg)S(x,y-1) \to R(x,y)$;
  - $S(x,y) \wedge T(x,y) \to R(x,y)$;
- soit une clause de contradiction :
  - $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R(x,y) \to \bot$.

8. Élimination des atomes $R(x,y)$ comme hypothèses

Arrivé à cette étape (encadré précédent, on peut considérer que la formule ψ est une conjonction de clauses, de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$ où $\mathbf{R} := \{R_1,\ldots,R_m\}$, des formes suivantes :

clause de calcul : $\delta_1 \wedge \ldots \wedge \delta_r \to R_j(x,y)$ où chaque hypothèse $\delta_1,\ldots,\delta_r$ est de l'une des quatre formes suivantes :

1. $\mathtt{min}(x) \wedge (\neg)\mathtt{min}(y) \wedge Q_s(y)$;
2. $\neg\mathtt{min}(x) \wedge (\neg)R_h(x-1,y)$;
3. $\neg\mathtt{min}(y) \wedge (\neg)R_h(x,y-1)$;
4. $R_i(x,y)$, pour $i < j$ [1] ;

clause de contradiction : $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R_j(x,y) \to \bot$.

Comme toujours, ψ se décompose en $\psi := \psi_0 \wedge \psi_{neg}$ où
$\psi_0 := \bigwedge_{j=1}^m \bigwedge_{\alpha\in A_j}(\alpha \to R_j(x,y))$ est la conjonction des clauses de calcul de ψ_0 et ψ_{neg} est la conjonction de ses clauses de contradiction.

Enfin, $\Phi := \exists\mathbf{R}\forall x\forall y(E(\psi_0) \wedge \psi_{neg})$ avec $E(\psi_0) := \bigwedge_{j=1}^m (R_j(x,y) \leftrightarrow (\bigvee_{\alpha\in A_j}\alpha))$. Par un abus de définition, on dit que les clauses de Φ sont les clauses de ψ.

Lemme 3.1.2 On peut transformer une formule $\Phi := \exists\mathbf{R}\forall x\forall y(E(\psi_0) \wedge \psi_{neg})$ de la forme ci-dessus en une formule $\Phi' := \exists\mathbf{R}\forall x\forall y(E(\psi_0') \wedge \psi_{neg})$ équivalente et de la même forme mais sans hypothèse $R_i(x,y)$.

1. Il est commode de numéroter les prédicats de $\mathbf{R}$ dans un ordre topologique des sommets du graphe acyclique G_ψ.

Démonstration. La preuve se fait par récurrence sur le nombre h d'atomes distincts $R_i(x,y)$ qui apparaissent comme hypothèses des clauses de ψ_0.

Il n'y a rien à prouver si $h = 0$. Sachant que les prédicats R_i sont numérotés dans un ordre topologique $R_1 < \ldots < R_m$ des sommets du graphe acyclique G_ψ, il suffit de montrer qu'on peut éliminer les hypothèses $R_1(x,y)$. Puis, la suppression des hypothèses $R_2(x,y), \ldots$, enfin $R_m(x,y)$, se fera de la même façon.

Soit $\alpha_1 \to R_1(x,y), \ldots, \alpha_k \to R_1(x,y)$ les clauses de ψ_0 dont la conclusion est $R_1(x,y)$. Notons que, puisque R_1 n'a aucun prédécesseur dans le graphe G_ψ, aucune hypothèse α_i ne contient d'atome $R_j(x,y)$. La conjonction $E(\psi_0)$ contient donc l'équivalence

$$R_1(x,y) \leftrightarrow \bigvee_{i=1}^{k} \alpha_i \tag{3.5}$$

où chaque α_i est une conjonction de littéraux d'entrée et de littéraux de calcul $R_j(x-1,y)$ et $R_j(x,y-1)$. L'équivalence 3.5 justifie le remplacement des atomes hypothèses $R_1(x,y)$ par la disjonction $\bigvee_{i=1}^{k} \alpha_i$ dans les clauses de ψ_0, donc celles de la forme

$$\alpha \wedge R_1(x,y) \to R_j(x,y) \tag{3.6}$$

pour $j \neq 1$. Plus précisément, chaque clause 3.6 étant équivalente par 3.5 à la conjonction de clauses

$$\bigwedge_{i=1}^{k} (\alpha \wedge \alpha_i \to R_j(x,y)), \tag{3.7}$$

on constate que la conjonction $E(\psi_0)$ est équivalente à $E(\psi_0')$ où la formule ψ_0' est la conjonction de clauses ψ_0 où on a substitué à chaque clause 3.6, $\alpha \wedge R_1(x,y) \to R_j(x,y)$, la conjonction de clauses équivalentes 3.7, ce qui fait que, comme demandé, l'atome $R_1(x,y)$ n'est hypothèse d'aucune clause de ψ_0'. Le lemme 3.1.2 est donc démontré. □

Par le lemme 3.1.2, on peut supposer que notre formule est maintenant de la forme $\exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ avec $\psi := \psi_0 \wedge \psi_{neg}$ conjonction de clauses des formes suivantes :

clause de calcul : $\delta_1 \wedge \ldots \wedge \delta_r \to R_j(x,y)$ où chaque hypothèse $\delta_1, \ldots, \delta_r$ est de l'une des trois formes suivantes :

1. $x = 1 \wedge (\neg)\mathtt{min}(y) \wedge Q_s(x)$, où $s \in \Sigma$;
2. $\neg\mathtt{min}(x) \wedge (\neg)R_h(x-1,y)$;
3. $\neg\mathtt{min}(y) \wedge (\neg)R_h(x,y-1)$;

clause de contradiction : $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R_h(x,y) \to \bot$.

Par une séparation en 4 cas selon la partition du domaine $[1,n]^2$ en les 4 sous-domaines (a-d) :

$$\text{a) } x = 1,\ y = 1 \text{ ; b) } x = 1,\ y > 1 \text{ ; c) } x > 1,\ y = 1 \text{ ; d) } x = y,\ x > 1 \text{ ;}$$

chaque clause de calcul ci-dessus est réécrite sous la forme équivalente d'une conjonction de clauses des formes (a-d) suivantes où A, B sont des parties (possiblement vides) de $[1,m]$, tandis que chaque clause de contradiction (e) est inchangée :

(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \to R_h(x,y)$;
(b) $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j \in A} (\neg) R_j(x,y-1) \to R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1,y) \to R_h(x,y)$;
(d) $x > 1 \wedge y > 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1,y) \wedge \bigwedge_{j \in B} (\neg) R_j(x,y-1) \to R_h(x,y)$;
(e) $x = n \wedge y = n \wedge R_h(x,y) \to \bot$.

Ce sont les clauses (a-e) de la forme normale demandée.

On a donc montré que pour toute formule de `pred-ESO-IND`, on peut définir une formule équivalente de `normal-pred-ESO-IND`, ce qui prouve la proposition 3.1.1. □

La proposition 3.1.1 nous permet d'établir la première inclusion de notre schéma comme indiqué sur la Figure 3.3.

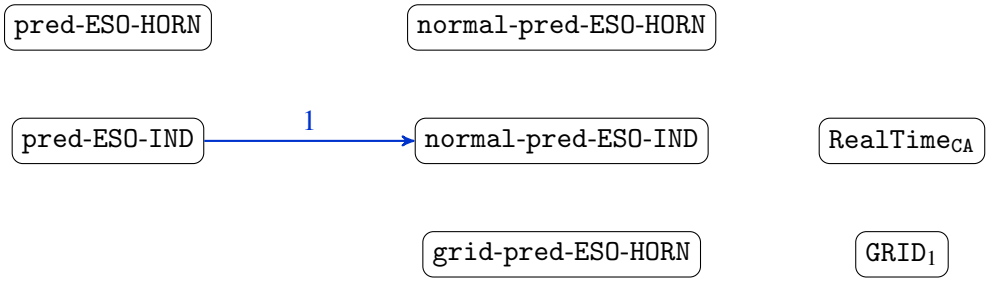


FIGURE 3.3 – Inclusion de pred-ESO-IND dans normal-pred-ESO-IND

Dans la sous-section 3.1.2 suivante, on va établir pour la logique de Horn (pour pred-ESO-HORN) le même résultat de normalisation que pour pred-ESO-IND. Notons que ce n'est pas un cas particulier de cette normalisation puisque la formule de Horn initiale peut ne pas être acyclique.

3.1.2 Inclusion de pred-ESO-HORN dans normal-pred-ESO-HORN

Afin de montrer que pred-ESO-HORN $\subseteq$ normal-pred-ESO-HORN, on va montrer que toute formule Φ de pred-ESO-HORN peut être normalisée en une formule Φ' équivalente appartenant à la classe normal-pred-ESO-HORN, définie ci-dessous, sous-classe de normal-pred-ESO-IND.

Définition 3.1.2 Une formule de normal-pred-ESO-HORN est une formule $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ où $\mathbf{R}$ est un ensemble de prédicats binaires et ψ est une conjonction de clauses de Horn sur les variables x, y, de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$, de l'une des formes suivantes où $s \in \Sigma$, $R_h \in \mathbf{R} = \{R_1, \ldots, R_m\}$, et A, B sont des parties (possiblement vides) de $[1, m]$:

(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j \in A} R_j(x, y-1) \rightarrow R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j \in A} R_j(x-1, y) \rightarrow R_h(x,y)$;
(d) $x > 1 \wedge y > 1 \wedge \bigwedge_{j \in A} R_j(x-1, y) \wedge \bigwedge_{j \in B} R_j(x, y-1) \rightarrow R_h(x,y)$;
(e) $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$.

Remarque 3.2 Toute formule $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ de normal-pred-ESO-HORN étant trivialement acyclique (de graphe vide) est équivalente à la formule $\exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ de normal-pred-ESO-IND où ψ_0 (resp. ψ_{neg}) est la conjonction des clauses de calcul (resp. clauses de contradictions) de ψ.

On cherche donc à prouver la proposition 3.1.3 ci-dessous :

Proposition 3.1.3 Toute formule $\Phi \in$ pred-ESO-HORN est équivalente à une formule $\Phi' \in$ normal-pred-ESO-HORN.

Démonstration. De la même façon que la proposition 3.1.1, cette proposition est prouvée par une normalisation des formules de pred-ESO-HORN. Les 7 premières étapes de la normalisation sont formellement les mêmes que les 7 premières de la sous-section précédente, excepté qu'on utilisera maintenant la sémantique de modèle minimal (d'une formule de Horn) au lieu de la sémantique du

modèle naturel (d'une formule inductive) défini par équivalences. Le lecteur pourra vérifier leur correction pour cette nouvelle sémantique. Après ces 7 étapes, chaque clause de la formule est :

— soit une clause d'entrée de la forme
  — $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge Q_s(y) \to R(x,y)$;
— soit une clause de calcul d'une des formes suivantes :
  — $\neg\texttt{min}(x) \wedge S(x-1,y) \to R(x,y)$;
  — $\neg\texttt{min}(y) \wedge S(x,y-1) \to R(x,y)$;
  — $S(x,y) \wedge T(x,y) \to R(x,y)$;
— soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

L'étape 8 précédemment décrite s'appuyant sur l'acyclicité de la formule, on ne peut pas utiliser la même méthode pour se débarrasser des atomes $R(x,y)$ dans les hypothèses car une formule de `pred-ESO-HORN` n'est en rien garantie d'être acyclique. On s'appuie alors sur le lemme suivant portant sur la logique de Horn propositionnelle :

Lemme 3.1.4 — folklore, voir (4). Soit $F := \bigwedge_{i=1}^{k} \theta_i$ une conjonction de clauses de Horn strictes θ_i sur les variables propositionnelles $p_1, \ldots, p_m$.

Soit $K := \{h \in [1,m] \mid F \to p_h \text{ est une tautologie}\}$. Alors les formules de Horn F et $\bigwedge_{h\in K} p_h$ ont le même modèle minimal I_0 donné par $I_0(p_h) = 1$ si $h \in K$ et $I_0(p_h) = 0$ si $h \notin K$.

Exemple : Pour $F := p_1 \wedge p_2 \wedge (p_1 \wedge p_2 \to p_3) \wedge (p_1 \wedge p_4 \to p_5)$, on obtient $K = \{1,2,3\}$. Les formules F et $p_1 \wedge p_2 \wedge p_3$ ont le même modèle minimal I_0 donné par $I_0(p_1) = I_0(p_2) = I_0(p_3) = 1$ et $I_0(p_4) = I_0(p_5) = 0$.

Preuve du lemme. Pour tout modèle I de F et tout $h \in K$, donc tel que $F \to p_h$ est une tautologie, on a $I(p_h) = 1$.

Il suffit donc de montrer que l'interprétation I_0 définie par $I_0(p_h) = 1$ si $h \in K$ et $I_0(p_h) = 0$ si $h \notin K$ satisfait chaque clause θ_i de F, ce qu'on note $I_0 \models \theta_i$.

Démontrons le par l'absurde. Supposons qu'on a $I_0 \not\models \theta_i$ pour au moins une clause de Horn (stricte) $\theta_i := p_{j_1} \wedge \ldots \wedge p_{j_\ell} \to p_{j_0}$ de F. On a donc $I_0(p_{j_1}) = \ldots = I_0(p_{j_\ell}) = 1$ et $I_0(p_{j_0}) = 0$. Par definition de I_0, on a $j_1, \ldots, j_\ell \in K$ et $j_0 \notin K$. Par définition de K, l'implication $F \to (p_{j_1} \wedge \ldots \wedge p_{j_\ell})$ est une tautologie. Puisque $\theta_i := p_{j_1} \wedge \ldots \wedge p_{j_\ell} \to p_{j_0}$ est une clause de F, on en conclut, par transitivité, que $F \to p_{j_0}$ est une tautologie, ce qui contredit l'assertion $j_0 \notin K$. □

Une fois les 7 premières étapes de normalisation effectuées, la formule Φ peut s'écrire

$$\Phi := \exists \mathbf{R} \forall x \forall y \left[\bigwedge_i C_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right]$$

où les C_i's sont les clauses de contradiction et chaque $\alpha_i(x,y) \to \theta_i(x,y)$ est une clause de calcul ou une clause d'entrée dans laquelle

— $\alpha_i(x,y)$ est de la forme $Q_s(y) \wedge \texttt{min}(x) \wedge (\neg)\texttt{min}(y)$ (resp. $Q_s(x) \wedge x = y \wedge (\neg)\texttt{min}(x)$) ou est une conjonction de formules de la forme $R(x-1,y) \wedge \neg\texttt{min}(x)$ ou $R(x,y-1) \wedge \neg\texttt{min}(y)$
— et $\theta_i(x,y)$ est une clause de Horn (stricte) *sur les seuls atomes* $R(x,y)$.

Numérotons $R_1, \ldots, R_m$ les prédicats de calcul de $\mathbf{R}$. A chaque sous-ensemble $J \subseteq [1,k]$ de la famille d'implications $(\alpha_i(x,y) \to \theta_i(x,y))_{i\in[1,k]}$ on associe l'ensemble de ses atomes "conséquences" :

$$K_J := \{h \in [1,m] \mid (\textstyle\bigwedge_{i\in J} \theta_i(x,y)) \to R_h(x,y) \text{ est une tautologie}\}.$$

Autrement dit, K_J est la partie maximale de $[1,m]$ telle que l'implication

$$\left(\bigwedge_{i\in J}\theta_i(x,y)\right) \to \bigwedge_{h\in K_J} R_h(x,y) \tag{3.8}$$

est une tautologie. Remarquons aussi que la fonction $J \mapsto K_J$ est croissante :

$$J \subseteq J' \subseteq [1,k] \;\Rightarrow\; K_J \subseteq K_{J'} \tag{3.9}$$

Le lemme 3.1.5 suivant donne une procédure constructive pour éliminer les hypothèses $R(x,y)$ dans toutes les clauses :

Lemme 3.1.5 La formule Φ est équivalente à la formule [a]

$$\Phi' := \exists\mathbf{R}\forall x\forall y \left[\bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]} \left(\bigwedge_{i\in J}\alpha_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)\right)\right].$$

a. Évidemment, la partie entre crochets de Φ' peut s'écrire comme conjonction de clauses :
$\Phi' := \exists\mathbf{R}\forall x\forall y \left[\bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]}\bigwedge_{h\in K_J}\left(\bigwedge_{i\in J}\alpha_i(x,y) \to R_h(x,y)\right)\right]$.

Preuve de $\Phi \Rightarrow \Phi'$. Il suffit de prouver, pour tout $J \subseteq [1,k]$, l'implication

$$\left(\bigwedge_{i\in[1,k]}(\alpha_i(x,y)\to\theta_i(x,y))\right) \Rightarrow \left(\bigwedge_{i\in J}\alpha_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)\right)$$

qui s'obtient par la suite d'implications suivantes :

$\left(\bigwedge_{i\in[1,k]}(\alpha_i(x,y)\to\theta_i(x,y))\right) \quad\Rightarrow\quad (\bigwedge_{i\in J}\alpha_i(x,y)\to\bigwedge_{i\in J}\theta_i(x,y)) \quad\Rightarrow$
$(\bigwedge_{i\in J}\alpha_i(x,y)\to\bigwedge_{h\in K_J}R_h(x,y))$,
où la dernière implication $\Rightarrow$ découle de la tautologie (3.8) ci-dessus.

□

Preuve de $\Phi' \Rightarrow \Phi$. Soit $\langle w\rangle$ un modèle de Φ' et soit $(\langle w\rangle,\mathbf{R})$ le modèle minimal (étendant $\langle w\rangle$) de la formule

$$\varphi' := \forall x\forall y \bigwedge_{J\subseteq[1,k]} \left(\bigwedge_{i\in J}\alpha_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)\right).$$

Il est suffisant de démontrer que $(\langle w\rangle,\mathbf{R})$ satisfait aussi la formule

$$\varphi := \forall x\forall y \bigwedge_{i\in[1,k]} (\alpha_i(x,y)\to\theta_i(x,y)).$$

On va prouver, par récurrence sur les entiers $t \in [1,2n]$, que le modèle minimal $(\langle w\rangle,\mathbf{R})$ de φ' satisfait la formule "relativisée" φ_t de la formule φ définie par

$$\varphi_t := \forall x\forall y \left(x+y\le t \to \bigwedge_{i\in[1,k]}(\alpha_i(x,y)\to\theta_i(x,y))\right).$$

Comme l'hypothèse $x+y\le 2n$ est vraie pour tout site $(x,y)\in[1,n]^2$, la formule φ_{2n} est équivalente à φ sur la structure $(\langle w\rangle,\mathbf{R})$. Ce qui donnera le résultat escompté $(\langle w\rangle,\mathbf{R}) \models \varphi$.

Cas de base de la récurrence : Comme l'ensemble $\{(x,y) \in [1,n]^2 \mid x+y \leq t\}$ est vide pour $t = 1$, la formule "relativisée" φ_1 est trivialement vraie dans le modèle minimal $(\langle w \rangle, \mathbf{R})$ de φ'.

Récurrence : Supposons qu'on a $(\langle w \rangle, \mathbf{R}) \models \varphi_{t-1}$, pour un entier $t \in [2,2n]$. Il suffit de montrer que, pour chaque site $(a,b) \in [1,n]^2$ tel que $a+b=t$, on a

$$(\langle w \rangle, \mathbf{R}) \models \bigwedge_{i \in [1,k]} (\alpha_i(a,b) \to \theta_i(a,b)). \tag{3.10}$$

Rappelons que chaque $\alpha_i(a,b)$ est soit une conjonction (possiblement vide) d'atomes [2] $R(a',b')$ où $(a',b') = (a-1,b)$ ou $(a',b') = (a,b-1)$, donc tels que $a'+b' = t-1$, soit une conjonction $Q_s(b) \wedge \mathtt{min}(a) \wedge (\neg)\mathtt{min}(b)$. Dans chaque cas, $\alpha_i(a,b)$ s'évalue à "vrai" ou à "faux" dans la structure $(\langle w \rangle, \mathbf{R})$ où on restreint les prédicats de calcul $R \in \mathbf{R}$ au domaine $\{(x,y) \in [1,n]^2 \mid x+y = t-1\}$.

Soit $J_{a,b}$ l'ensemble des indices $i \in [1,k]$ tels que le site (a,b) satisfait α_i :

$$J_{a,b} := \{i \in [1,k] \mid (\langle w \rangle, \mathbf{R}) \models \alpha_i(a,b)\}.$$

Par la définition de $J_{a,b}$, il est clair que l'assertion (3.10) à démontrer est équivalente à

$$(\langle w \rangle, \mathbf{R}) \models \bigwedge_{i \in J_{a,b}} \theta_i(a,b). \tag{3.11}$$

Examinons maintenant la formule φ' où interviennent les ensembles $J \subseteq [1,k]$. Considérons les deux cas possibles pour J :

— $J \subseteq J_{a,b}$: alors, par définition de $J_{a,b}$, la conjonction $\bigwedge_{i \in J} \alpha_i(a,b)$ est vraie dans la structure $(\langle w \rangle, \mathbf{R})$; par conséquent, dans la structure $(\langle w \rangle, \mathbf{R})$, l'implication $\bigwedge_{i \in J} \alpha_i(a,b) \to \bigwedge_{h \in K_J} R_h(a,b)$ est équivalente à $\bigwedge_{h \in K_J} R_h(a,b)$;

— $J \setminus J_{a,b} \neq \emptyset$: alors, toujours par définition de $J_{a,b}$, la conjonction $\bigwedge_{i \in J} \alpha_i(a,b)$ est fausse dans $(\langle w \rangle, \mathbf{R})$ et donc l'implication $\bigwedge_{i \in J} \alpha_i(a,b) \to \bigwedge_{h \in K_J} R_h(a,b)$ est vraie dans la structure $(\langle w \rangle, \mathbf{R})$.

De ces deux items, on déduit que dans $(\langle w \rangle, \mathbf{R})$ la conjonction

$$\bigwedge_{J \subseteq [1,k]} \left(\bigwedge_{i \in J} \alpha_i(a,b) \to \bigwedge_{h \in K_J} R_h(a,b) \right) \tag{3.12}$$

est équivalente à la conjonction $\bigwedge_{J \subseteq J_{a,b}} \bigwedge_{h \in K_J} R_h(a,b)$. Puisque $J \subseteq J_{a,b}$ implique $K_J \subseteq K_{J_{a,b}}$ par (3.9), cette dernière conjonction se simplifie en

$$\bigwedge_{h \in K_{J_{a,b}}} R_h(a,b). \tag{3.13}$$

Par définition, $K_{J_{a,b}}$ est l'ensemble des $h \in [1,m]$ tels que l'implication "propositionnelle" $\left(\bigwedge_{i \in J_{a,b}} \theta_i(x,y)\right) \to R_h(x,y)$ est une tautologie. De façon équivalente,

$$K_{J_{a,b}} := \{h \in [1,m] \mid (\textstyle\bigwedge_{i \in J_{a,b}} \theta_i(a,b)) \to R_h(a,b) \text{ est une tautologie}\}.$$

2. En effet, chaque conjonction $R(a-1,b) \wedge \neg\mathtt{min}(a)$ s'évalue à "faux" si $a = 1$ et à $R(a-1,b)$ si $a > 1$. Et similairement pour chaque conjonction $R(a,b-1) \wedge \neg\mathtt{min}(b)$.

Or le lemme 3.1.4 établit que la conjonction de clauses de Horn "propositionnelles" $\bigwedge_{i\in J_{a,b}} \theta_i(a,b)$ a le même modèle minimal que $\bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$.

Par ailleurs, la restriction du modèle minimal $(\langle w\rangle, \mathbf{R})$ de φ' aux atomes $R_h(a,b)$ est modèle minimal de $\bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$ et est donc aussi modèle minimal de $\bigwedge_{i\in J_{a,b}} \theta_i(a,b)$.

On a donc prouvé $(\langle w\rangle, \mathbf{R}) \models \bigwedge_{i\in J_{a,b}} \theta_i(a,b)$, qui est l'assertion (3.11) qu'on voulait démontrer. □

En utilisant le Lemme 3.1.5, on transforme notre formule en une formule équivalente dont les clauses ne contiennent pas d'atomes $R(x,y)$ dans leurs hypothèses. On effectue alors la même séparation en cas que dans la sous-section précédente afin d'obtenir uniquement des clauses de l'une des formes suivantes où $s \in \Sigma$, $R_h \in \mathbf{R} = \{R_1, \ldots, R_m\}$, et A, B sont des parties (possiblement vides) de $[1,m]$:

(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j\in A} R_j(x,y-1) \rightarrow R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j\in A} R_j(x-1,y) \rightarrow R_h(x,y)$;
(d) $x > 1 \wedge y > 1 \wedge \bigwedge_{j\in A} R_j(x-1,y) \wedge \bigwedge_{j\in B} R_j(x,y-1) \rightarrow R_h(x,y)$;
(e) $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$.

Ces clauses étant exactement celles demandées par la forme normalisée, la proposition 3.1.3 est prouvée. □

De plus, toute formule de normal-pred-ESO-HORN étant acyclique (le graphe d'une telle formule est vide car aucun atome $R(x,y)$ ne se trouve dans les hypothèses de ses clauses), l'inclusion de normal-pred-ESO-HORN dans normal-pred-ESO-IND est triviale. On peut donc ajouter deux nouvelles inclusions à notre schéma comme le montre la Figure 3.4.

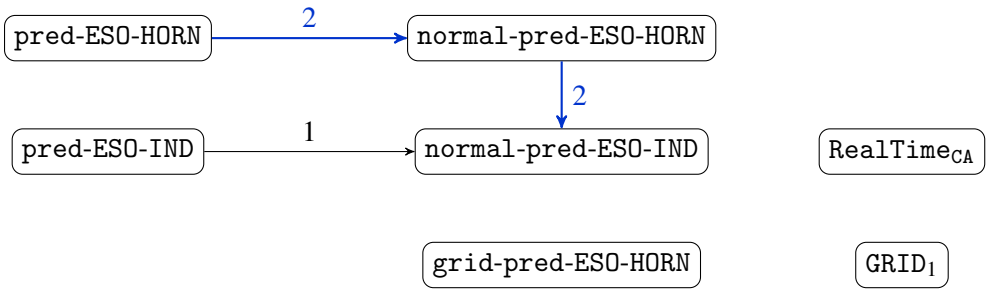


FIGURE 3.4 – Inclusion de pred-ESO-HORN dans normal-pred-ESO-HORN, elle même incluse dans normal-pred-ESO-IND

3.1.3 Inclusion de normal-pred-ESO-IND dans grid-pred-ESO-HORN

On va montrer ici que toute formule de pred-ESO-IND est équivalente à une formule de Horn dite *formule grille* dont les principaux intérêts sont les suivants : il est immédiat de construire, à partir d'une formule grille définissant un langage L, d'une part une nouvelle formule grille qui définit le langage complémentaire $\Sigma^+ \setminus L$, d'autre part un automate grille $\mathtt{GRID}_1$ qui reconnaît le langage L.

L'idée de départ de cette normalisation en formule grille, est de décrire par un même prédicat de calcul la valeur de vérité de chaque prédicat de calcul de la formule d'origine, ceci pour chaque

site (x,y).

Définition 3.1.3 — description. Soit $\mathbf{R} = \{R_1,\dots,R_m\}$ un ensemble de prédicats binaires. On appelle $\mathbf{R}$-*description* ou *description*, quand $\mathbf{R}$ est sous-entendu, une conjonction de la forme $\Delta(x,y) := \bigwedge_{i\in A} R_i(x,y) \wedge \bigwedge_{i\in[1,m]\setminus A} \neg R_i(x,y)$, où A est une partie de l'ensemble $[1,m]$. On dit que $\Delta(x,y)$ *complète* une conjonction $\bigwedge_{i\in B} R_i(x,y) \wedge \bigwedge_{i\in C} \neg R_i(x,y)$ si $B \subseteq A$ et $C \cap A = \emptyset$. L'ensemble des $\mathbf{R}$-descriptions, de cardinal $p = 2^m$, est noté $\texttt{Descr} = \{\Delta_1,\dots,\Delta_p\}$.

On utilisera le fait évident suivant :

Fait 3.1 Toute conjonction de littéraux de calcul $\gamma := \bigwedge_{i\in B} R_i(x,y) \wedge \bigwedge_{i\in C} \neg R_i(x,y)$ est équivalente à la disjonction $\bigvee_\Delta \Delta(x,y)$ des $\mathbf{R}$-descriptions Δ qui complètent γ.

Proposition 3.1.6 — logique de grille. Toute formule $\Phi \in \texttt{normal-pred-ESO-IND}$ est équivalente à une formule de Horn $\Phi' \in \texttt{grid-pred-ESO-HORN}$, dite *formule grille*, de la forme

$$\Phi' := \exists \mathbf{T} \forall x \forall y \left[\psi'(x,y) \wedge \bigwedge_{i\in N} (x = n \wedge y = n \wedge T_i(x,y) \to \bot) \right]$$

où $\mathbf{T} = \{T_1,\dots,T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ' est une conjonction de clauses de Horn des formes (1-4) suivantes :

1. $x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \to T_{f(i,j)}(x,y)$;
2. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \to T_{g(i)}(x,y)$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge T_i(x,y-1) \to T_{u(s,i)}(x,y)$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \to T_{v(s)}(x,y)$.

Ici, f,g,u,v sont des fonctions ; autrement dit, si deux clauses $\theta \to \alpha$ et $\theta \to \alpha'$ ont la même hypothèse θ alors elles ont la *même conclusion* $\alpha = \alpha' = T_{h(\theta)}(x,y)$.

De plus, pour tout mot $w \in \Sigma^n$, $n \geq 1$, la formule $\forall x \forall y \psi'$ a un *unique modèle* $(\langle w \rangle, \mathbf{T})$ et, dans ce modèle, les prédicats $T_1,\dots,T_p$ forment une *partition* de la grille $[1,n]^2$ (certains T_i peuvent être vides) : $(\langle w \rangle, \mathbf{T}) \models \texttt{Partition}(\mathbf{T})$ avec

$$\texttt{Partition}(\mathbf{T}) := \forall x \forall y \left[\bigvee_{1\leq i\leq p} T_i(x,y) \wedge \bigwedge_{1\leq i<j\leq p} (\neg T_i(x,y) \vee \neg T_j(x,y)) \right]$$

Remarque 3.3 — partition des R-descriptions. L'intuition de la preuve de la proposition s'appuie sur la notion de $\mathbf{R}$-description, ou description quand $\mathbf{R}$ est sous-entendu. Les prédicats T_i correspondent aux $\mathbf{R}$-descriptions Δ_i. Par définition, les Δ_i réalisent une partition des couples d'éléments du domaine, autrement dit, les Δ_i satisfont la formule $\texttt{Partition}(\texttt{Descr}) := \forall x \forall y \left[\bigvee_{1\leq i\leq p} \Delta_i(x,y) \wedge \bigwedge_{1\leq i<j\leq p} (\neg\Delta_i(x,y) \vee \neg\Delta_j(x,y)) \right]$.

Preuve de la proposition 3.1.6. Soit Φ une formule de $\texttt{normal-pred-ESO-IND}$, c'est-à-dire $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$, où :

$\mathbf{R} = \{R_1,\dots,R_m\}$ est un ensemble de prédicats binaires ;

ψ_0 est conjonction des clauses (a-d) suivantes où $s \in \Sigma$, $R_h \in \mathbf{R}$, et A,B sont des parties (possiblement vides) de $[1,m]$:

(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \to R_h(x,y)$,

(b) $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{j\in A} (\neg) R_j(x,y-1) \to R_h(x,y)$,

(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j\in A} (\neg) R_j(x-1,y) \to R_h(x,y)$,

(d) $x > 1 \wedge y > 1 \wedge \bigwedge_{j\in A} (\neg) R_j(x-1,y) \wedge \bigwedge_{j\in B} (\neg) R_j(x,y-1) \to R_h(x,y)$;

$E(\psi_0)$ est la conjonction d'équivalences obtenue à partir de ψ_0 (voir Définition 3.1.1) ;

ψ_{neg} est conjonction des clauses (e), où $R_h \in \mathbf{R}$:

(e) $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$

$\psi := \psi_0 \wedge \psi_{neg}$

Dans une première étape, on va remplacer chaque clause (b-d) par une conjonction d'implications, équivalente à cette clause et dont les hypothèses portent exclusivement sur les littéraux d'entrée et les $\mathbf{R}$-descriptions $\Delta_1, \ldots, \Delta_p$.

Chaque clause (b) est remplacée par la conjonction

— $\bigwedge_\Delta (x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \Delta(x, y-1) \rightarrow R_h(x,y))$

où Δ est une description qui complète la conjonction $\bigwedge_{j \in A}(\neg)R_j(x, y-1)$. L'équivalence de la clause initiale (b) avec cette conjonction de clauses est garantie par le fait 3.1.

De même, chaque clause (c) est remplacée par la conjonction

$$\bigwedge_\Delta (x > 1 \wedge y = 1 \wedge \Delta(x-1, y) \rightarrow R_h(x,y))$$

où Δ est une description qui complète la conjonction $\bigwedge_{j \in A}(\neg)R_j(x-1, y)$.

De même encore, chaque clause (d) est remplacée par la conjonction

$$\bigwedge_\Delta \bigwedge_{\Delta'} (x > 1 \wedge y > 1 \wedge \Delta(x-1, y) \wedge \Delta'(x, y-1) \rightarrow R_h(x,y))$$

où Δ (resp. Δ') est une description qui complète la conjonction $\bigwedge_{j \in A}(\neg)R_j(x-1, y)$ (resp. $\bigwedge_{j \in B}(\neg)R_j(x, y-1)$).

Enfin, la clause (e) est remplacée par

$$\bigwedge_\Delta (x = n \wedge y = n \wedge \Delta(x,y) \rightarrow \bot)$$

où Δ est une description qui complète $R_h(x,y)$.

Récapitulation : En examinant les clauses obtenues, on constate qu'elles sont des formes (1-5) suivantes où $\Delta_i, \Delta_j \in \texttt{Descr} = \{\Delta_1, \ldots, \Delta_p\}$:

1. $x > 1 \wedge y > 1 \wedge \Delta_i(x-1, y) \wedge \Delta_j(x, y-1) \rightarrow R_h(x,y)$;
2. $x > 1 \wedge y = 1 \wedge \Delta_i(x-1, y) \rightarrow R_h(x,y)$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \Delta_i(x, y-1) \rightarrow R_h(x,y)$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
5. $x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot$.

Notation 3.2 Soit ψ_1 la conjonction des clauses (1-4) ainsi obtenues (celles dont la conclusion est un atome $R_h(x,y)$) et soit N l'ensemble des indices i des $\mathbf{R}$-descriptions Δ_i des clauses (5) (les clauses de contradiction).

Avec ces notations, on a établi le fait suivant :

Fait 3.2 La formule initiale

$$\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$$

est équivalente à la formule

$$\Phi_1 := \exists \mathbf{R} \forall x \forall y \left[E(\psi_1) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot) \right]$$

Point-clé (incompatibilité des hypothèses) : On constate qu'à cause de la propriété de partition réalisée par les descriptions Δ_i, complétée par la propriété de partition des Q_s, deux membres gauches différents θ, θ' de deux implications $\theta \rightarrow \ldots$ et $\theta' \rightarrow \ldots$ de ψ_1 sont toujours *incompatibles* :

pour tout mot $w \in \Sigma^+$ et toute expansion $(\langle w \rangle, \mathbf{R})$ de $\langle w \rangle$, on a $(\langle w \rangle, \mathbf{R}) \models \forall x \forall y (\theta(x,y) \wedge \theta'(x,y) \rightarrow \bot)$.

On donne maintenant un procédé uniforme qui transforme, pour chacune des quatre formes (1-4), la conjonction des clauses de cet item en une conjonction d'implications dont la conclusion est une **R**-description.

Décrivons cette transformation pour les clauses de l'item (1), sachant que la transformation est absolument similaire pour les clauses des items (2-4).

Notation 3.3 Pour tout couple $(i,j) \in [1,p]^2$ (resp. tout $h \in [1,m]$), on note $A_{i,j}$ l'ensemble des $h \in [1,m]$ (resp. on note A_h^{-1} l'ensemble des couples $(i,j) \in [1,p]^2$) tels qu'il existe une clause (1) $x > 1 \wedge y > 1 \wedge \Delta_i(x-1,y) \wedge \Delta_j(x,y-1) \rightarrow R_h(x,y)$ de ψ_1, avec ces indices i, j, h.

On obtient le fait suivant :

Fait 3.3 Soit un mot $w \in \Sigma^+$ et soit $\mathtt{S}_1 := (\langle w \rangle, \mathbf{R})$ le modèle unique de la formule $\forall x \forall y E(\psi_1)$ sur $\langle w \rangle$. On a

$$\mathtt{S}_1 \models \forall x \forall y \, [x > 1 \wedge y > 1 \wedge \Delta_i(x-1,y) \wedge \Delta_j(x,y-1) \rightarrow \Delta_{f(i,j)}(x,y)]$$

où $\Delta_{f(i,j)}(x,y)$ est la **R**-description $\bigwedge_{h \in A_{i,j}} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus A_{i,j}} \neg R_h(x,y))$.

Démonstration. Notons $\alpha_{i,j} := \Delta_i(x-1,y) \wedge \Delta_j(x,y-1)$. Du fait que $\mathtt{S}_1$ est modèle de $E(\psi_1)$, on déduit en particulier

$$\mathtt{S}_1 \models \forall x \forall y \left[x > 1 \wedge y > 1 \rightarrow \left(R_h(x,y) \leftrightarrow \bigvee_{(i,j) \in A_h^{-1}} \alpha_{i,j}(x,y) \right) \right].$$

Puisque les p^2 ensembles $\{(x,y) \in]1,n]^2 \mid \alpha_{i,j}(x,y)\}$, $i,j \in [1,p]$, forment une partition du domaine $]1,n]^2$, on en déduit

$$\mathtt{S}_1 \models \forall x \forall y \left[x > 1 \wedge y > 1 \rightarrow \left(\neg R_h(x,y) \leftrightarrow \bigvee_{(i,j) \in [1,p]^2 \setminus A_h^{-1}} \alpha_{i,j}(x,y) \right) \right].$$

De ces deux points on conclut, pour tout $(i,j) \in [1,p]^2$,

$$\mathtt{S}_1 \models \forall x \forall y \left[x > 1 \wedge y > 1 \rightarrow \left(\alpha_{i,j}(x,y) \rightarrow \Delta_{f(i,j)} \right) \right]$$

avec la **R**-description $\Delta_{f(i,j)} := \bigwedge_{h \in A_{i,j}} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus A_{i,j}} \neg R_h(x,y)$.

□

On a ainsi décrit et justifié la transformation des clauses de la forme (1). Décrivons maintenant les transformations des clauses des formes (2-4). Pour cela, on introduit les notations suivantes :

Notation 3.4 **Clauses (2) :** pour $i \in [1,p]$, on note B_i l'ensemble des $h \in [1,m]$ tels que ψ_1 contient la clause $x > 1 \wedge y = 1 \wedge \Delta_i(x-1,y) \rightarrow R_h(x,y)$;

Clauses (3) : pour $s \in \Sigma$ et $i \in [1,p]$, on note $C_{s,i}$ l'ensemble des $h \in [1,m]$ tels que ψ_1 contient la clause $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \Delta_i(x,y-1) \rightarrow R_h(x,y)$;

Clauses (4) : pour $s \in \Sigma$, on note D_s l'ensemble des $h \in [1,m]$ tels que ψ_1 contient la clause $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$.

Le fait 3.4 suivant qui reprend le fait 3.3 pour les clauses (1) et le complète pour les clauses (2-4) se prouve de façon absolument similaire.

Fait 3.4 Soit un mot $w \in \Sigma^+$ et soit $\mathtt{S}_1 := (\langle w \rangle, \mathbf{R})$ le modèle unique sur $\langle w \rangle$ de la formule $\forall x \forall y E(\psi_1)$.

Clauses (1) : Pour tout $(i,j) \in [1,p]^2$, on a :
$\mathtt{S}_1 \models \forall x \forall y\, [x > 1 \wedge y > 1 \wedge \Delta_i(x-1,y) \wedge \Delta_j(x,y-1) \rightarrow \Delta_{f(i,j)}(x,y)]$
où $\Delta_{f(i,j)}(x,y)$ est la $\mathbf{R}$-description $\bigwedge_{h \in A_{i,j}} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus A_{i,j}} \neg R_h(x,y))$.

Clauses (2) : Pour tout $i \in [1,p]$, on a :
$\mathtt{S}_1 \models \forall x \forall y\, [x > 1 \wedge y = 1 \wedge \Delta_i(x-1,y) \rightarrow \Delta_{g(i)}(x,y)]$
où $\Delta_{g(i)}(x,y)$ est la $\mathbf{R}$-description $\bigwedge_{h \in B_i} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus B_i} \neg R_h(x,y))$.

Clauses (3) : Pour tout $s \in \Sigma$ et tout $i \in [1,p]$, on a :
$\mathtt{S}_1 \models \forall x \forall y\, [x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \Delta_j(x,y-1) \rightarrow \Delta_{u(s,i)}(x,y)]$
où $\Delta_{u(s,i)}(x,y)$ est la $\mathbf{R}$-description $\bigwedge_{h \in C_{s,i}} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus C_{s,i}} \neg R_h(x,y))$.

Clauses (4) : Pour tout $s \in \Sigma$, on a :
$\mathtt{S}_1 \models \forall x \forall y\, [x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow \Delta_{v(s)}(x,y)]$
où $\Delta_{v(s)}(x,y)$ est la $\mathbf{R}$-description $\bigwedge_{h \in D_s} R_h(x,y) \wedge \bigwedge_{h \in [1,m] \setminus D_s} \neg R_h(x,y))$.

Soit ψ_2 la conjonction des clauses (1-4) devenues les implications des formes (1-4) suivantes où les $R_h \in \mathbf{R}$ n'apparaissent qu'à l'intérieur des $\Delta_j \in \mathtt{Descr}$:

1. $x > 1 \wedge y > 1 \wedge \Delta_i(x-1,y) \wedge \Delta_j(x,y-1) \rightarrow \Delta_{f(i,j)}(x,y)$, pour $i,j \in [1,p]$;
2. $x > 1 \wedge y = 1 \wedge \Delta_i(x-1,y) \rightarrow \Delta_{g(i)}(x,y)$, pour $i \in [1,p]$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \Delta_i(x,y-1) \rightarrow \Delta_{u(s,i)}(x,y)$, pour $s \in \Sigma$ et $i \in [1,p]$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow \Delta_{v(s)}(x,y)$, pour $s \in \Sigma$.

On remarque que la formule ψ_2 considère tous les cas possibles : l'ensemble des parties gauches des clauses de Horn (1-4) est exactement l'ensemble de toutes les situations hypothèses possibles. Par ailleurs, la conclusion de chaque clause (1-4) est une $\mathbf{R}$-description.

Par une induction évidente, on en déduit que, pour tout mot $w \in \Sigma^n$, $n \geq 1$, dans chaque modèle $(\langle w \rangle, \mathbf{R})$ de ψ_2 sur $\langle w \rangle$, la $\mathbf{R}$-description vérifiée par un site $(x_0,y_0) \in [1,n]^2$ ne dépend que de w et de (x_0,y_0). En conséquence, la formule $\forall x \forall y \psi_2$ a un modèle unique sur $\langle w \rangle$. En utilisant le fait 3.4 on en déduit le fait suivant :

Fait 3.5 Pour tout mot $w \in \Sigma^+$, le modèle unique $\mathtt{S}_1 := (\langle w \rangle, \mathbf{R})$ de $\forall x \forall y E(\psi_1)$ sur $\langle w \rangle$ est aussi le modèle unique de $\forall x \forall y \psi_2$ sur $\langle w \rangle$.

En conséquence, les formules suivantes Φ_1 et Φ_2 sont équivalentes :

— $\Phi_1 := \exists \mathbf{R} \forall x \forall y [E(\psi_1) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot)]$;
— $\Phi_2 := \exists \mathbf{R} \forall x,y [\psi_2 \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot)]$;

L'idée de la dernière transformation donnant la formule fortement normalisée est de remplacer chaque $\mathbf{R}$-description $\Delta_j \in \mathtt{Descr} = \{\Delta_1, \ldots, \Delta_p\}$ par un nouveau prédicat binaire $T_j \in \mathbf{T} = \{T_1, \ldots, T_p\}$.

Soit ψ' la transformée de la formule ψ_2 par ces substitutions. ψ' est donc la conjonction des clauses de Horn (1-4) suivantes, calquées sur les implications (1-4) de ψ_2 :

1. $x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{f(i,j)}(x,y)$, pour $i,j \in [1,p]$;
2. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \rightarrow T_{g(i)}(x,y)$, pour $i \in [1,p]$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge T_i(x,y-1) \rightarrow T_{u(s,i)}(x,y)$, pour $s \in \Sigma$ et $i \in [1,p]$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow T_{v(s)}(x,y)$, pour $s \in \Sigma$.

Puisque la formule ψ', portant sur les T_j, calque la formule ψ_2, portant sur les Δ_j, on obtient avec le fait 3.5 le résultat suivant :

Fait 3.6 Pour tout mot $w \in \Sigma^+$, il existe *un et un seul* modèle de la formule $\forall x \forall y \psi'$ qui étend la structure $\langle w \rangle$. Ce modèle $\mathtt{S}' := (\langle w \rangle, \mathbf{T})$ est défini à partir de l'unique modèle $\mathtt{S}_1 := (\langle w \rangle, \mathbf{R})$ de la formule $\forall x \forall y \psi_2$ par l'équivalence

$$\mathtt{S}' \models T_j(x_0,y_0) \Leftrightarrow_{def} \mathtt{S}_1 \models \Delta_j(x_0,y_0),$$

pour tout $j \in [1,p]$ et tout $(x_0,y_0) \in [1,n]^2$. En particulier, on a $\mathtt{S}' \models \mathtt{Partition}(\mathbf{T})$.

On a démontré le fait suivant :

Fait 3.7 Les formules suivantes $\Phi, \Phi_1, \Phi_2, \Phi'$ sont équivalentes :

— $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$;
— $\Phi_1 := \exists \mathbf{R} \forall x \forall y [E(\psi_1) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot)]$;
— $\Phi_2 := \exists \mathbf{R} \forall x \forall y [\psi_2 \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge \Delta_i(x,y) \rightarrow \bot)]$;
— $\Phi' := \exists \mathbf{T} \forall x \forall y [\psi' \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot)]$.

Fin de la preuve de la proposition 3.1.6 : La formule Φ' a la forme demandée et est équivalente à la formule initiale Φ, par le fait 3.7. De plus, par le fait 3.6, pour tout mot $w \in \Sigma^+$, la formule $\forall x \forall y \psi'$ a un unique modèle $\mathcal{S}' := (\langle w \rangle, \mathbf{T})$ et vérifie la propriété $\mathcal{S}' \models \texttt{Partition}(\mathbf{T})$. Tous les points de la proposition 3.1.6 sont donc démontrés. □

Par la proposition 3.1.6, on a l'inclusion suivante `normal-pred-ESO-IND` $\subseteq$ `grid-pred-ESO-IND`. Toute formule de `grid-pred-ESO-HORN` étant une formule de Horn acyclique (pas d'atomes $R(x,y)$ dans les hypothèses des clauses de la formule), l'inclusion de `grid-pred-ESO-HORN` à la fois dans `pred-ESO-IND` et `pred-ESO-HORN` est triviale. On peut donc compléter notre schéma (voir Figure 3.5) et obtenir la première équivalence du Théorème 3.0.1 :

$$L \in \texttt{pred-ESO-HORN} \iff L \in \texttt{pred-ESO-IND}$$

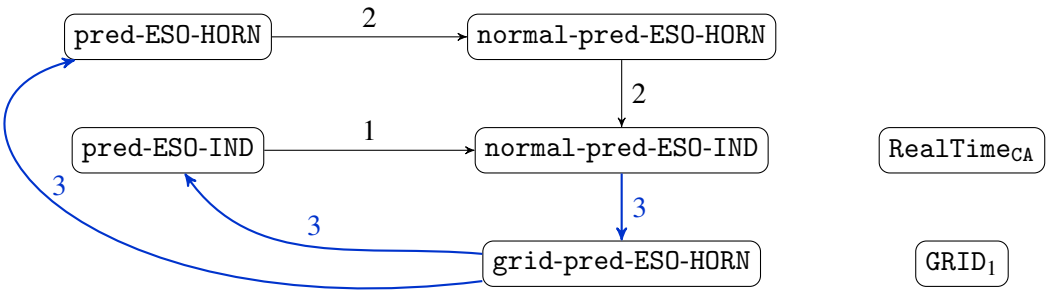


FIGURE 3.5 – Inclusion de `normal-pred-ESO-HORN` dans `grid-pred-ESO-HORN` elle-même incluse dans `pred-ESO-HORN` et `pred-ESO-IND`

Comme annoncé au début de cette sous-section 3.1.3, la logique `grid-pred-ESO-HORN` permet à partir d'une formule grille définissant un langage L sur un alphabet Σ de construire aisément une nouvelle formule grille qui définit le langage complémentaire $\Sigma^+ \setminus L$. Cette formule est donnée par la proposition suivante :

Proposition 3.1.7 Soit un langage $L \subseteq \Sigma^+$ dans `pred-ESO-IND` (resp. `pred-ESO-HORN`) et soit $\Phi := \exists \mathbf{T} \forall x \forall y [\psi(x,y) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot)]$ une formule grille appartenant à `grid-pred-ESO-HORN` qui définit L et où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ est une conjonction de clauses de Horn des formes (1-4) suivantes :

1. $x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{f(i,j)}(x,y)$;
2. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \rightarrow T_{g(i)}(x,y)$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge T_i(x,y-1) \rightarrow T_{u(s,i)}(x,y)$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow T_{v(s)}(x,y)$.

Le langage $\Sigma^+ \setminus L$ est défini par la formule

$$\Phi^c := \exists \mathbf{T} \forall x \forall y \left[\psi(x,y) \wedge \bigwedge_{i \in [1,p] \setminus N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \right]$$

Démonstration. Soit un mot $w \in \Sigma^+$ et soit $\mathtt{S} := (\langle w \rangle, \mathbf{T})$ l'unique modèle sur $\langle w \rangle$ de la formule $\forall x \forall y \psi$. De cette unicité du modèle et de sa propriété de partition on tire les équivalences suivantes : $\langle w \rangle \models \Phi \iff (\langle w \rangle, \mathbf{T}) \models \forall x \forall y \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \iff (\langle w \rangle, \mathbf{T}) \models \bigwedge_{i \in N} \neg T_i(n,n) \iff (\langle w \rangle, \mathbf{T}) \models \bigvee_{i \in [1,p] \setminus N} T_i(n,n)$. On a donc l'équivalence $\langle w \rangle \models \Phi \iff (\langle w \rangle, \mathbf{T}) \models \bigvee_{i \in [1,p] \setminus N} T_i(n,n)$. Par la définition de la formule Φ^c, on obtient de même l'équivalence $\langle w \rangle \models \Phi^c \iff (\langle w \rangle, \mathbf{T}) \models \bigvee_{i \in N} T_i(n,n)$. Par la propriété de partition, les formules $\bigvee_{i \in N} T_i(n,n)$ et $\bigvee_{i \in [1,p] \setminus N} T_i(n,n)$ sont négation l'une de l'autre, d'où l'équivalence $\langle w \rangle \models \Phi^c \iff \langle w \rangle \not\models \Phi \iff w \in \Sigma^+ \setminus L$. □

L'autre propriété essentielle des formules de grid-pred-ESO-HORN est leur bijection avec les automates grilles $\mathtt{GRID}_1$.

3.1.4 Égalité des classes grid-pred-ESO-HORN, $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{CA}}$

On prouve dans cette section l'équivalence donnée par le théorème 3.0.1 entre nos deux logiques du prédécesseur et le temps-réel des automates cellulaire avec entrée parallèle et voisinage bidirectionnel. Par la proposition 3.1.6, on a égalité de ses classes pred-ESO-HORN, pred-ESO-IND et grid-pred-ESO-HORN. Il nous reste donc à montrer l'égalité entre grid-pred-ESO-HORN et $\mathtt{RealTime}_{\mathtt{CA}}$.
On prouve cette égalité en deux temps à l'aide du circuit-grille $\mathtt{GRID}_1$. Dans un premier temps, on montrera que les formules de grid-pred-ESO-HORN sont en bijection directe avec les automates grilles de $\mathtt{GRID}_1$. Dans un second temps, on montrera que les classes $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{CA}}$ sont égales.

Égalité de grid-pred-ESO-HORN et $\mathtt{GRID}_1$

La proposition suivante donne une bijection entre les formules de grid-pred-ESO-HORN et les circuits-grilles de $\mathtt{GRID}_1$.

Proposition 3.1.8 Soit

$$\Phi := \exists \mathbf{T} \forall x \forall y \left[\psi(x,y) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \right]$$

une formule de grid-pred-ESO-HORN définissant un langage L où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ est une conjonction de clauses de Horn des formes (1-4) suivantes, qui vérifie la condition d'*unicité de modèle* et la condition de *partition* de la proposition 3.1.6 :

1. $x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{f(i,j)}(x,y)$;
2. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \rightarrow T_{g(i)}(x,y)$;
3. $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge T_i(x,y-1) \rightarrow T_{u(s,i)}(x,y)$;
4. $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow T_{v(s)}(x,y)$.

Le langage L est reconnu par le circuit grille $(\Sigma, \mathtt{Input}_n, \mathtt{S}, \mathtt{f}, \mathtt{A})$ de $\mathtt{GRID}_1$ dont l'ensemble des états est $\mathtt{S} := \{q_1, \ldots, q_p\}$, l'ensemble des états acceptants est $\mathtt{A} := \{q_i \mid i \in [1,p] \setminus N\}$ et la fonction de transition $\mathtt{f}$ est donnée par les items (1-4) suivants, calqués sur les formes (1-4) des

clauses de ψ :
1. $\mathtt{f}(q_i, q_j, \$) = q_{f(i,j)}$;
2. $\mathtt{f}(q_i, \#, \$) = q_{g(i)}$;
3. $\mathtt{f}(\#, q_i, s) = q_{u(s,i)}$;
4. $\mathtt{f}(\#, \#, s) = q_{v(s)}$.

Démonstration. Pour tout mot d'entrée $w \in \Sigma^n$, $n \geq 1$, l'état $\langle x, y\rangle$ du site $(x, y) \in [1, n]^2$ du circuit-grille $(\Sigma, \mathtt{Input}_n, \mathtt{S}, \mathtt{f}, \mathtt{A})$ de $\mathtt{GRID}_1$ est donné par la formule

$$\mathtt{f}(\langle x-1, y\rangle, \langle x, y-1\rangle, \mathtt{Input}_n(w, x, y)) = \langle x, y\rangle$$

où $\mathtt{Input}_n(w, x, y)$ vaut w_y si $x = 1$ et vaut \$ si $x > 1$. On montre par induction que, pour tout $(x_0, y_0) \in [1, n]^2$ et tout état $q_i \in \mathtt{S}$, on a l'équivalence : $\langle x_0, y_0\rangle = q_i \iff (\langle w\rangle, \mathbf{T}) \models T_i(x_0, y_0)$, où $(\langle w\rangle, \mathbf{T})$ est l'unique modèle de la formule $\forall x \forall y \psi$ pour le mot w. Cette équivalence donne pour $(x_0, y_0) = (n, n)$ l'équivalence $(\langle w\rangle, \mathbf{T}) \models \bigvee_{i \in [1,p] \setminus N} T_i(n, n) \iff \langle n, n\rangle \in [1, p] \setminus N$. D'où la suite d'équivalences : $w \in L \iff (\langle w\rangle, \mathbf{T}) \models \Phi \iff (\langle w\rangle, \mathbf{T}) \models \bigvee_{i \in [1,p] \setminus N} T_i(n, n) \iff \langle n, n\rangle \in \mathtt{A}$. Ce qui signifie que L est le langage reconnu par le circuit-grille $(\Sigma, \mathtt{Input}_n, \mathtt{S}, \mathtt{f}, \mathtt{A})$. □

La bijection donnée dans la proposition 3.1.8 donne l'équivalence entre `grid-pred-ESO-HORN` et $\mathtt{GRID}_1$ que l'on peut ajouter sur notre schéma de preuves Figure 3.1.8.

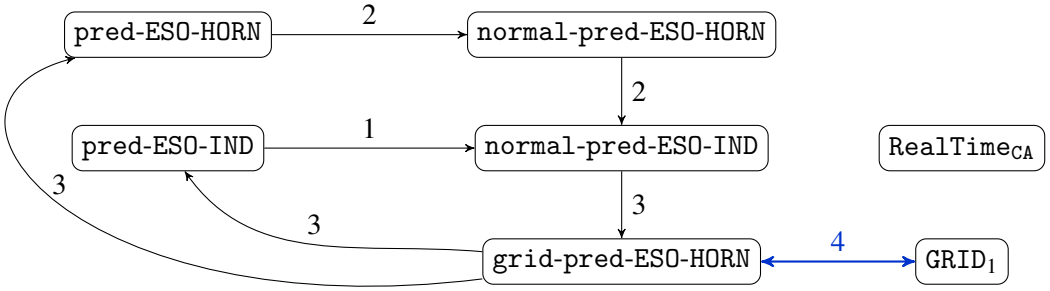


FIGURE 3.6 – Égalité de `grid-pred-ESO-HORN` et $\mathtt{GRID}_1$

Égalité de $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{CA}}$

La preuve de l'équivalence entre $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{CA}}$ utilise l'égalité $\mathtt{RealTime}_{\mathtt{OIA}} = \mathtt{RealTime}_{\mathtt{CA}}$ bien connue en théorie des automates (voir [5, 25] et le survey [48], pp. 136-137). Il existe en effet une bijection immédiate entre les sites (x, y) de la grille et les sites (c, t) du diagramme espace-temps de $\mathtt{RealTime}_{\mathtt{OIA}}$: à chaque site (x, y) de la grille est associé le site $(x, x+y-1)$ du diagramme (et réciproquement, à chaque site (c, t) du diagramme est associé le site $(c, t-c+1)$ de la grille). Cette bijection est montrée sur la Figure 3.7.

À l'aide de cette bijection on peut donc déduire la fonction de transition de l'automate grille à partir de celle de l'automate cellulaire et inversement. L'ensemble des états et l'ensemble des états acceptants restent quant à eux inchangés entre ces deux objets.
On a donc égalité des classes $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{OIA}}$. Cette dernière étant égale à la classe $\mathtt{RealTime}_{\mathtt{CA}}$, on obtient la dernière égalité de notre schéma de preuves (voir Figure 3.8). Ce qui conclut la preuve du théorème 3.0.1.

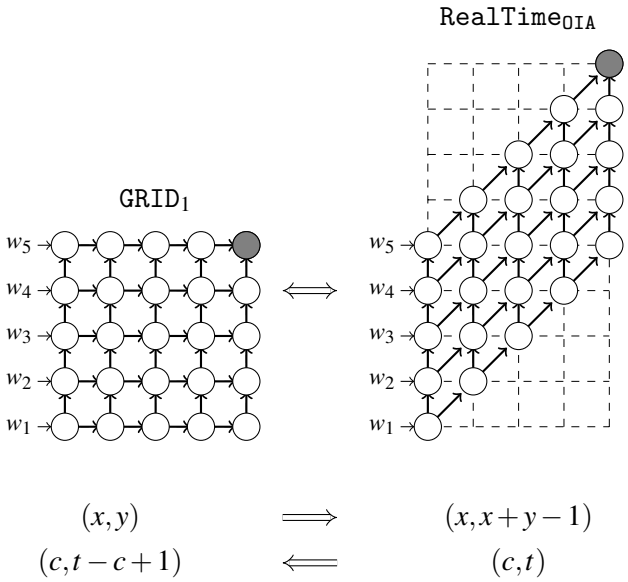


FIGURE 3.7 – Bijection entre $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{OIA}}$

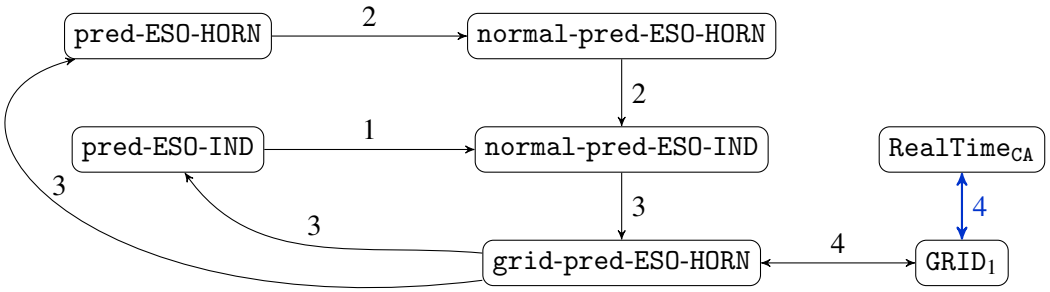


FIGURE 3.8 – Égalité entre $\mathtt{GRID}_1$ et $\mathtt{RealTime}_{\mathtt{CA}}$

3.2 Égalité des classes pred-dio-ESO-HORN, pred-dio-ESO-IN et $\texttt{RealTime}_{\texttt{IA}}$

La preuve du théorème 3.0.2 suit le même schéma que la preuve du théorème 3.0.1 donnée dans la section précédente. De plus, la seule différence entre les classes pred-dio-ESO-IND et pred-ESO-IND étant l'accès à l'entrée, le contenu des étapes de la preuve est lui aussi très similaire. Comme pour la preuve du théorème 3.0.1, on introduit ici aussi trois nouvelles logiques :

1. normal-pred-dio-ESO-HORN correspondant aux formules de pred-dio-ESO-HORN une fois normalisées ;
2. normal-pred-dio-ESO-IND correspondant aux formules de pred-dio-ESO-IND normalisées ;
3. grid-pred-dio-ESO-HORN dont chaque formule est en en directe bijection avec un circuit-grille.

La Figure 3.9 montre la Figure 3.1 adaptée à la preuve des équivalences du théorème 3.0.2.

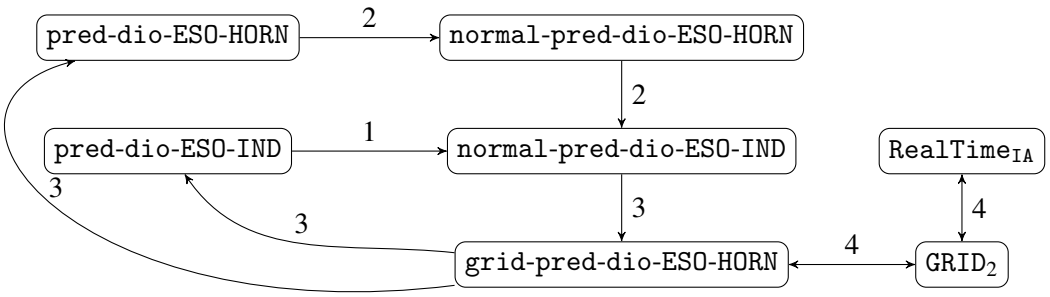


FIGURE 3.9 – Schéma de la preuve d'équivalence entre pred-dio-ESO-HORN, pred-dio-ESO-IND et $\texttt{RealTime}_{\texttt{IA}}$

3.2.1 Égalité de pred-dio-ESO-IND et normal-pred-dio-ESO-IND

On définit ici la logique normal-pred-dio-ESO-IND correspondant aux formules de pred-dio-ESO-IND normalisées.

Définition 3.2.1 Une formule de normal-pred-dio-ESO-IND est une formule
$\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ où :
$\mathbf{R} = \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
$\psi_0 := \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow R_j(x,y))$ est une conjonction de clauses de la forme suivante où $s \in \Sigma$, $R_h \in \mathbf{R}$ et A, B sont des parties (possiblement vides) de $[1,m]$:
(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x > 1 \wedge x = y \wedge Q_s(y) \wedge \bigwedge_{j \in A} (\neg) R_j(x, y-1) \wedge \bigwedge_{j \in B} (\neg) R_j(x-1, y) \rightarrow R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1, y) \rightarrow R_h(x,y)$;
(d) $x = 1 \wedge y > 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x, y-1) \rightarrow R_h(x,y)$;
(e) $x \neq y \wedge x > 1 \wedge y > 1 \wedge \bigwedge_{j \in A} (\neg) R_j(x-1, y) \wedge \bigwedge_{j \in B} (\neg) R_j(x, y-1) \rightarrow R_h(x,y)$;
$E(\psi_0) := \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$ est la conjonction d'équivalences obtenue à partir de ψ_0 ;
ψ_{neg} est une conjonction de clauses de la forme :

(f) $x = n \wedge y = n \wedge R_h(x,y) \to \bot$, avec $R_h \in \mathbf{R}$.
On note ψ la conjonction $\psi_0 \wedge \psi_{neg}$ de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=\}$.

Dans cette sous-section, on prouve la proposition suivante :

Proposition 3.2.1 Toute formule $\Phi \in$ pred-dio-ESO-IND est équivalente à une formule $\Phi' \in$ normal-pred-dio-ESO-IND.

Démonstration. Soit $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ une formule de pred-dio-ESO-IND, alors $\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses de la forme $\delta_1 \wedge \ldots \wedge \delta_r \to \beta$, avec $R \in \mathbf{R}$, où chaque δ_i est :

- soit une conjonction d'entrée : $Q_s(x-a) \wedge x = y$, pour $s \in \Sigma$ et $a \geq 0$,
- soit un littéral d'entrée : $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \geq 0$,
- soit un atome de calcul $S(x,y)$, $S(y,x)$, $(\neg)S(x-a,y-b)$ ou $(\neg)S(y-b,x-a)$ pour $S \in \mathbf{R}$ et $a,b \geq 0$ tels que $a+b>0$;

et où la conclusion β est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$ soit le symbole $\bot$.

Φ est transformée en une formule normalisée équivalente $\Phi' \in$ normal-pred-dio-ESO-IND par une succession de 9 étapes :

1. Traitement des clauses de contradiction ;
2. Traitement de l'entrée ;
3. Restriction des atomes de calcul à $(\neg)R(x-1,y)$, $(\neg)R(x,y-1)$, $R(x,y)$ et $R(y,x)$;
4. Élimination des littéraux $(\neg)\texttt{min}(x-a)$, $(\neg)\texttt{min}(y-b)$, $(\neg)\texttt{max}(x-a)$ et $(\neg)\texttt{max}(y-b)$ pour $a,b>0$;
5. Traitement de min et max ;
6. Pliage du domaine ;
7. Suppression de max dans les clauses d'initialisation ;
8. Transformation des clauses d'initialisation en clauses d'entrée ;
9. Élimination des atomes $R(x,y)$ comme hypothèses.

Les étapes 1, 3, 4, 5, 6, et 9 sont les mêmes que lors de la normalisation des formules de pred-ESO-IND. Par souci de clarté, on rappelle toutefois à la fin de chacune des étapes l'ensemble des formes que peuvent prendre les clauses de la formule. Les étapes 2 et 7 sont adaptées et l'étape 8 ajoutée pour gérer l'accès à l'entrée particulier des formules de pred-dio-ESO-IND.

1. Traitement des clauses de contradiction

Cette étape n'est pas modifiée par rapport à la section précédente.

Après cette étape, chaque clause de la formule est :
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y)$ où chaque δ_i est :
  - soit :
    - une conjonction d'entrée $Q_s(x-a) \wedge x = y$,
    - ou un littéral d'entrée $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \geq 0$;
  - soit un atome de calcul $S(x,y)$, $S(y,x)$ ou une conjonction $x > a \wedge y > b \wedge (\neg)S(x-a,y-b)$, $x > a \wedge y > b \wedge (\neg)S(y-b,x-a)$, pour $a,b \geq 0$ tels que $a+b>0$;
- soit une clause de contradiction : $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

2. Traitement de l'entrée

Cette étape n'est que légèrement modifiée par rapport à la section précédente. On introduit les mêmes prédicats W_s^x et W_s^y pour $s \in \Sigma$ servant à transporter l'information du mot d'entrée. Les clauses de transport restent inchangées et seules sont modifiées les clauses d'initialisation qui sont maintenant :

- $Q_s(y) \wedge x = y \rightarrow W_s^x(x,y)$, ou
- $Q_s(y) \wedge x = y \rightarrow W_s^y(x,y)$ pour $s \in \Sigma$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge Q_s(y) \rightarrow R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min},\texttt{max}\}$ et $a \geq 0$,
  - soit un atome de calcul $S(x,y)$, $S(y,x)$ ou une conjonction $x > a \wedge y > b \wedge (\neg)S(x-a,y-b)$, $x > a \wedge y > b \wedge (\neg)S(y-b,x-a)$ pour $a,b \geq 0$ tels que $a+b > 0$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

3. Restriction des atomes de calcul à $(\neg)R(x-1,y)$, $(\neg)R(x,y-1)$, $R(x,y)$ et $R(y,x)$

Cette étape n'est pas modifiée par rapport à la section précédente.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge Q_s(y) \rightarrow R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{\texttt{min},\texttt{max}\}$ et $a \geq 0$,
  - soit un atome de calcul $S(x,y)$ $S(y,x)$ ou une conjonction $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y)$ ou $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

4. Élimination des littéraux $(\neg)\texttt{min}(x-a)$, $(\neg)\texttt{min}(y-b)$, $(\neg)\texttt{max}(x-a)$ et $(\neg)\texttt{max}(y-b)$ pour $a,b > 0$

Cette étape n'est pas modifiée par rapport à la section précédente.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge Q_s(y) \rightarrow R(x,y)$;
- soit une clause de calcul $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ avec $R \in \mathbf{R}$ et où chaque δ_i est :
  - soit un littéral d'entrée $(\neg)\texttt{min}(x)$, $(\neg)\texttt{max}(x)$, $(\neg)\texttt{min}(y)$ ou $(\neg)\texttt{max}(y)$;
  - soit un atome de calcul $S(x,y)$ ou $S(y,x)$ ou une conjonction $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y)$ ou $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

5. Traitement de min et max

Cette étape n'est pas modifiée par rapport à la section précédente.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation :
  - $x = y \wedge Q_s(x,y) \rightarrow R(x,y)$;
  - $\texttt{min}(x) \wedge \texttt{min}(y) \wedge (\neg)\texttt{max}(y) \rightarrow R(x,y)$;
  - $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge (\neg)\texttt{max}(y) \rightarrow R(x,y)$;
  - $\neg\texttt{min}(x) \wedge \texttt{min}(y) \wedge (\neg)\texttt{max}(x) \rightarrow R(x,y)$
- soit une clause de calcul :
  - $(\neg)S(x-1,y) \wedge \neg\texttt{min}(x) \rightarrow R(x,y)$;
  - $(\neg)S(x,y-1) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$;
  - $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
  - $S(y,x) \rightarrow R(x,y)$.
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

6. Pliage du domaine

Le pliage du domaine est effectué de la même manière que dans la section précédente.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation :
  - $x = y \wedge Q_s(x,y) \rightarrow R(x,y)$
  - $\texttt{min}(x) \wedge (\neg)\texttt{min}(y) \wedge (\neg)\texttt{max}(y) \rightarrow R(x,y)$;
- soit une clause de calcul :
  - $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y) \rightarrow R(x,y)$;
  - $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$;
  - $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

7. Suppression de max dans les clauses d'initialisation

On effectue ici les mêmes manipulations que dans la section précédente. Chaque prédicat de calcul $R \in \mathbf{R}$ est dupliqué en deux nouveaux prédicats $R^{y}_{\leftarrow\texttt{max}}$ et $R^{y}_{\leftarrow\neg\texttt{max}}$. La modification des clauses de calcul reste la même. La modification des clauses d'initialisation est adaptée. Par exemple, la clause $\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \neg\texttt{max}(y) \rightarrow R(x,y)$ est équivalente à $\texttt{min}(x) \wedge \neg\texttt{min}(y) \rightarrow (\neg\texttt{max}(y) \rightarrow R(x,y))$ qui se réécrit $\texttt{min}(x) \wedge \neg\texttt{min}(y) \rightarrow R^{y}_{\leftarrow\neg\texttt{max}}$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation :
  1. $x = y \wedge Q_s(x,y) \rightarrow R(x,y)$;
  2. $\texttt{min}(x) \wedge \texttt{min}(y) \rightarrow R(x,y)$;
  3. $\texttt{min}(x) \wedge \neg\texttt{min}(y) \rightarrow R(x,y)$;
- soit une clause de calcul :
  - $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y) \rightarrow R(x,y)$;
  - $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$;
  - $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
- soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

8. Transformation des clauses d'initialisation en clauses d'entrée

On souhaite rendre l'information sur l'entrée uniquement disponible par des clauses d'entrée d'une des deux formes :
- $x = y \wedge \texttt{min}(x) \wedge Q_s(x,y) \rightarrow R(x,y)$
- et $x = y \wedge \neg\texttt{min}(x) \wedge Q_s(x,y) \rightarrow R(x,y)$, pour $s \in \Sigma$.

Chaque clause (2) peut se réécrire $\texttt{min}(x) \wedge x = y \rightarrow R(x,y)$.
On introduit trois nouveaux prédicats $R^{\texttt{min}(x)}$, $R^{\texttt{min}(y)}$ et $R^{\neg\texttt{min}(y)}$ afin de modifier les clauses (3). Le sens intuitif de ces prédicats est naturel, par exemple $R^{\neg\texttt{min}(y)}(x,y) \iff \neg\texttt{min}(y)$. Ces prédicats sont définis par les clauses d'initialisation

4. $x = y \wedge \texttt{min}(x) \rightarrow R^{\texttt{min}(x)}(x,y)$;
5. $x = y \wedge \texttt{min}(x) \rightarrow R^{\texttt{min}(y)}(x,y)$;

et les clauses de calcul
- $\neg\texttt{min}(y) \wedge R^{\texttt{min}(x)}(x,y-1) \rightarrow R^{\texttt{min}(x)}(x,y)$;
- $\neg\texttt{min}(y) \wedge R^{\texttt{min}(y)}(x,y-1) \rightarrow R^{\neg\texttt{min}(y)}(x,y)$;
- $\neg\texttt{min}(y) \wedge R^{\neg\texttt{min}(y)}(x,y-1) \rightarrow R^{\neg\texttt{min}(y)}(x,y)$.

On peut alors remplacer chaque clause d'initialisation (3) par la clause de calcul $R^{\texttt{min}(x)}(x,y) \wedge R^{\neg\texttt{min}(y)}(x,y) \rightarrow R(x,y)$.

Les clauses d'initialisation ne sont donc plus que de l'une des deux formes :

— $x = y \wedge Q_s(x,y) \rightarrow R(x,y)$ (clauses (1) ci-dessus) ;
— $x = y \wedge \texttt{min}(x) \rightarrow R(x,y)$ (clauses (2), (4) et (5) ci-dessus).

On remplace de façon équivalente chaque clause d'initialisation $x = y \wedge Q_s(x,y) \rightarrow R(x,y)$ par la conjonction des deux clauses d'entrée :

— $x = y \wedge \texttt{min}(x) \wedge Q_s(x,y) \rightarrow R(x,y)$
— et $x = y \wedge \neg\texttt{min}(x) \wedge Q_s(x,y) \rightarrow R(x,y)$.

De la même façon, on remplace chaque clause $x = y \wedge \texttt{min}(x) \rightarrow R(x,y)$ par la conjonction de clauses d'entrée :

— $\bigwedge_{s\in\Sigma} x = y \wedge \texttt{min}(x) \wedge Q_s(x,y) \rightarrow R(x,y)$.

Après cette étape, chaque clause de la formule est :
— soit une clause d'entrée :
  — $x = y \wedge \texttt{min}(x) \wedge Q_s(x) \rightarrow R(x,y)$;
  — $x = y \wedge \neg\texttt{min}(x) \wedge Q_s(x) \rightarrow R(x,y)$;
— soit une clause de calcul :
  — $\neg\texttt{min}(x) \wedge (\neg)S(x-1,y) \rightarrow R(x,y)$;
  — $\neg\texttt{min}(y) \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$;
  — $S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
— soit une clause de contradiction $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

9. Élimination des atomes $R(x,y)$ comme hypothèses

Cette étape utilise la même méthode de substitution que la sous-section 3.1.1 précédente (normalisation de `pred-ESO-IND`), autrement dit, l'analogue du lemme 3.1.2 pour `pred-dio-ESO-IND`. Au final, on obtient une formule de la forme $\exists\mathbf{R}\forall x\forall y(E(\psi_0) \wedge \psi_{neg})$ où $\mathbf{R} = \{R_1,\ldots,R_m\}$ et où $\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses des formes suivantes :

clause de calcul : $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R_j(x,y)$ où chaque hypothèse $\delta_1,\ldots,\delta_r$ est de l'une des trois formes suivantes :
1. $x = y \wedge (\neg)\texttt{min}(x) \wedge Q_s(x)$;
2. $\neg\texttt{min}(x) \wedge (\neg)R_h(x-1,y)$;
3. $\neg\texttt{min}(y) \wedge (\neg)R_h(x,y-1)$;

clause de contradiction : $\texttt{max}(x) \wedge \texttt{max}(y) \wedge R_h(x,y) \rightarrow \bot$.

Par une séparation en 5 cas selon la partition du domaine $[1,n]^2$ en 5 sous-domaines :

a) $x = 1, y = 1$; b) $x = 1, y > 1$; c) $x > 1, y = 1$; d) $x = y, x > 1$; e) $x \neq y, x > 1, y > 1$;

les clauses de calcul ci-dessus sont réécrites sous l'une des 5 formes ci-dessous correspondant aux formes des clauses (a-e) de `normal-pred-dio-ESO-IND`, où A,B sont des parties (possiblement vides) de $[1,m]$:

(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x = 1 \wedge y > 1 \wedge \bigwedge_{j\in A}(\neg)R_j(x,y-1) \rightarrow R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j\in A}(\neg)R_j(x-1,y) \rightarrow R_h(x,y)$;
(d) $x = y \wedge x > 1 \wedge Q_s(x) \wedge \bigwedge_{j\in A}(\neg)R_j(x-1,y) \wedge \bigwedge_{j\in B}(\neg)R_j(x,y-1) \rightarrow R_h(x,y)$;
(e) $x \neq y \wedge x > 1 \wedge y > 1 \wedge \bigwedge_{j\in A}(\neg)R_j(x-1,y) \wedge \bigwedge_{j\in B}(\neg)R_j(x,y-1) \rightarrow R_h(x,y)$.

Les clauses de contradiction, elles, restent inchangées.

L'application de ces 9 étapes prouve la proposition 3.1.1. □

La proposition 3.1.1 nous permet d'établir une première inclusion sur notre schéma (voir figure 3.10).

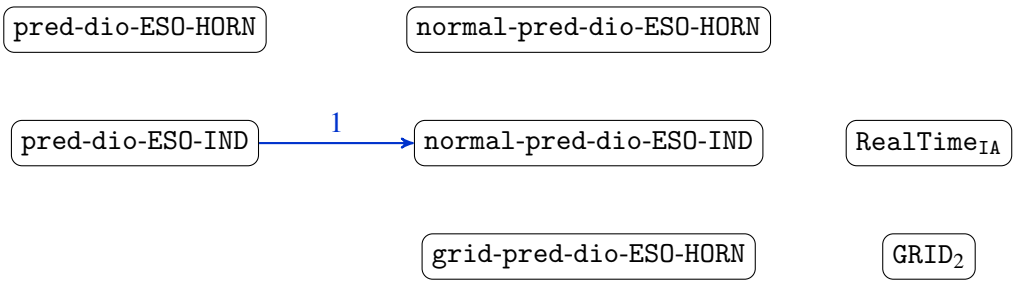


FIGURE 3.10 – Inclusion de pred-dio-ESO-IND dans normal-pred-dio-ESO-IND

3.2.2 Inclusion de pred-dio-ESO-HORN dans normal-pred-dio-ESO-HORN

Notre but est ici de montrer, comme pour la logique inductive, que toute formule Φ de pred-dio-ESO-HORN peut être normalisée en une formule Φ' équivalente appartenant à normal-pred-dio-ESO-HORN.

Définition 3.2.2 Une formule de normal-pred-dio-ESO-HORN est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :
$\mathbf{R} = \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
ψ est une conjonction de clauses de Horn, de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=\}$, des formes suivantes où $s \in \Sigma$, $R_h \in \mathbf{R} = \{R_1, \ldots, R_m\}$, et A, B sont des parties (possiblement vides) de $[1,m]$:
(a) $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x = 1 \wedge y > 1 \wedge \bigwedge_{j \in A} R_j(x, y-1) \rightarrow R_h(x,y)$;
(c) $x > 1 \wedge y = 1 \wedge \bigwedge_{j \in A} R_j(x-1, y) \rightarrow R_h(x,y)$;
(d) $x = y \wedge x > 1 \wedge Q_s(x) \wedge \bigwedge_{j \in A} R_j(x-1, y) \wedge \bigwedge_{j \in B} R_j(x, y-1) \rightarrow R_h(x,y)$;
(e) $x \neq y \wedge x > 1 \wedge y > 1 \wedge \bigwedge_{j \in A} R_j(x-1, y) \wedge \bigwedge_{j \in B} R_j(x, y-1) \rightarrow R_h(x,y)$;
(f) $x = n \wedge y = n \wedge R_h(x,y) \rightarrow \bot$.

Cette normalisation se déroule de la même manière que dans le cas de pred-ESO-HORN (voir proposition 3.1.3) : on commence par les 8 premières étapes de la normalisation de pred-dio-ESO-IND puis on se débarrasse des atomes $R(x,y)$ dans les hypothèses en utilisant le lemme 3.1.5. En effectuant ensuite la séparation en cas de l'étape 9 de la sous-section 3.2.1 précédente, on retrouve la forme des clauses voulues pour les formules de normal-pred-dio-ESO-HORN, on a donc pred-dio-ESO-HORN $\subseteq$ normal-pred-dio-ESO-HORN.
Les formules de normal-pred-dio-ESO-HORN étant acycliques, leur appartenance à normal-pred-dio-ESO-IND est triviale. La figure 3.11 donne notre avancement dans la preuve du théorème 3.0.2.

3.2.3 Inclusion de normal-pred-dio-ESO-IND dans grid-pred-dio-ESO-HORN

De la même façon que toute formule de normal-pred-ESO-IND est équivalente à une formule grille de grid-pred-ESO-HORN, toute formule de normal-pred-dio-ESO-IND est équivalente à une formule grille de grid-pred-dio-ESO-HORN. C'est ce que dit la proposition suivante.

Proposition 3.2.2 — logique de grille avec entrée diagonale. Toute formule $\Phi \in$ normal-

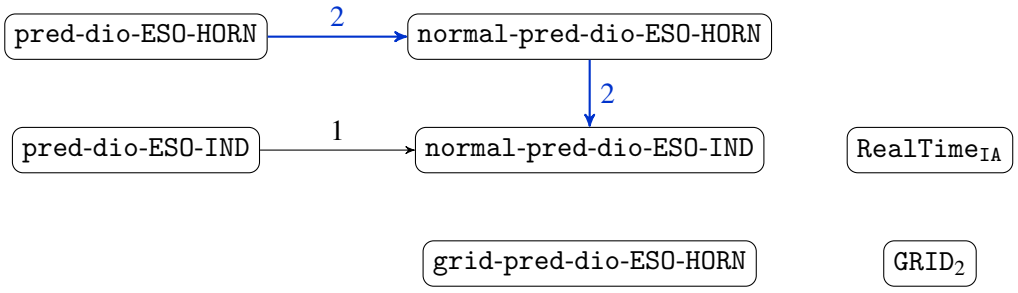


FIGURE 3.11 – Inclusion de pred-dio-ESO-HORN dans normal-pred-dio-ESO-HORN, elle même incluse dans normal-pred-dio-ESO-IND

pred-dio-ESO-IND est équivalente à une formule de Horn $\Phi' \in$ grid-pred-dio-ESO-HORN, dite *formule grille avec entrée diagonale*, de la forme

$$\Phi' := \exists \mathbf{T} \forall x \forall y \left[\psi'(x,y) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \right]$$

où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ' est une conjonction de clauses de Horn des formes (1-5) suivantes :

1. $x \neq y \wedge x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{f(i,j)}(x,y)$;
2. $x = y \wedge x > 1 \wedge Q_s(x) \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{g(s,i,j)}(x,y)$;
3. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \rightarrow T_{t(i)}(x,y)$;
4. $x = 1 \wedge y > 1 \wedge T_i(x,y-1) \rightarrow T_{u(i)}(x,y)$;
5. $x = 1 \wedge y = 1 \wedge Q_s(x) \rightarrow T_{v(s)}(x,y)$.

Ici, f, g, t, u, v sont des fonctions ; autrement dit, si deux clauses $\theta \rightarrow \alpha$ et $\theta \rightarrow \alpha'$ ont la même hypothèse θ alors elles ont la *même conclusion* $\alpha = \alpha' = T_{h(\theta)}(x,y)$.

Démonstration. La transformation d'une formule de normal-pred-dio-ESO-IND en formule équivalente de grid-pred-dio-ESO-HORN est exactement la même que celle utilisée pour la logique du prédécesseur (voir Proposition 3.1.6 et sa preuve). La preuve de la proposition 3.2.2 est donc immédiate. □

De plus, comme pour la logique du prédécesseur, les formules de grid-pred-dio-ESO-HORN sont trivialement des formules de Horn acycliques et donc leur inclusion dans pred-dio-ESO-HORN et pred-dio-ESO-IND est elle aussi immédiate. Notre schéma de preuve avance (voir Figure 3.13) et illustre maintenant la première équivalence du théorème 3.0.2 :

$$L \in \text{pred-dio-ESO-HORN} \iff L \in \text{pred-dio-ESO-IND}$$

3.2.4 Égalité de grid-pred-dio-ESO-HORN, GRID$_2$ et RealTime$_{IA}$

On prouve dans cette section la seconde équivalence donnée par le théorème 3.0.2, entre nos logiques et le temps-réel des automates cellulaires avec entrée séquentielle et communication bidirectionnelle. Par la proposition 3.2.2, on a égalité des classes pred-dio-ESO-HORN, pred-dio-ESO-IND

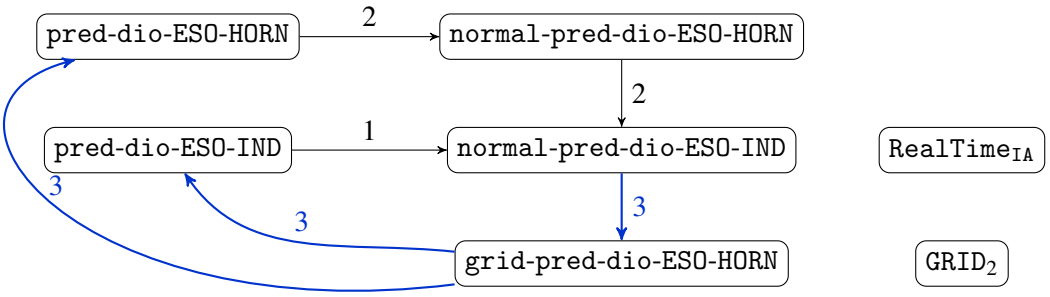


FIGURE 3.12 – Inclusion de normal-pred-dio-ESO-HORN dans grid-pred-dio-ESO-HORN elle même incluse dans pred-dio-ESO-HORN et pred-dio-ESO-IND

et grid-pred-dio-ESO-HORN. Il suffit donc de montrer que grid-pred-dio-ESO-HORN et RealTime$_{IA}$ définissent la même classe de langages.
On prouve cette équivalence en deux temps à l'aide du circuit-grille GRID$_2$. Dans un premier temps, on montrera que les formules de grid-pred-dio-ESO-HORN sont en bijection directe avec les circuits-grilles de GRID$_2$ et dans un second temps, on montrera l'égalité de GRID$_2$ et RealTime$_{IA}$.

Égalité de grid-pred-dio-ESO-HORN et GRID$_2$

La proposition suivante établit une bijection entre les formules de grid-pred-dio-ESO-HORN et les circuits-grilles de GRID$_2$. La preuve de cette proposition est identique à celle de la proposition 3.1.8 et n'est donc pas répétée.

Proposition 3.2.3 Soit

$$\Phi := \exists \mathbf{T} \forall x \forall y \left[\psi(x,y) \wedge \bigwedge_{i \in N} (x = n \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \right]$$

une formule grille de grid-pred-dio-ESO-HORN définissant un langage L où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ est une conjonction de clauses de Horn des formes (1-5) suivantes :

1. $x \neq y \wedge x > 1 \wedge y > 1 \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{f(i,j)}(x,y)$;
2. $x = y \wedge x > 1 \wedge Q_s(x) \wedge T_i(x-1,y) \wedge T_j(x,y-1) \rightarrow T_{g(s,i,j)}(x,y)$;
3. $x > 1 \wedge y = 1 \wedge T_i(x-1,y) \rightarrow T_{t(i)}(x,y)$;
4. $x = 1 \wedge y > 1 \wedge T_i(x,y-1) \rightarrow T_{u(i)}(x,y)$;
5. $x = 1 \wedge y = 1 \wedge Q_s(x) \rightarrow T_{v(s)}(x,y)$.

Le langage L est reconnu par le circuit-grille $(\Sigma, \mathtt{Input}_n, \mathtt{S}, \mathtt{f}, \mathtt{A})$ de GRID$_2$ dont l'ensemble des états $\mathtt{S}$ est $\{q_1, \ldots, q_p\}$, l'ensemble des états acceptants est $\mathtt{A} := \{q_i \mid i \in [1,p] \setminus N\}$ et la fonction de transition $\mathtt{f}$ est donnée par les items (1-5) suivants, calqués sur les formes (1-5) des clauses de ψ :

1. $\mathtt{f}(q_i, q_j, \$) = q_{f(i,j)}$;
2. $\mathtt{f}(q_i, q_j, s) = q_{g(s,i,j)}$;
3. $\mathtt{f}(q_i, \#, \$) = q_{t(i)}$;
4. $\mathtt{f}(\#, q_i, \$) = q_{u(i)}$;
5. $\mathtt{f}(\#, \#, s) = q_{v(s)}$.

La bijection de la proposition 3.2.3 établit une égalité entre grid-pred-dio-ESO-HORN et $\mathtt{GRID}_2$ complétant notre schéma comme montré par la figure 3.13.

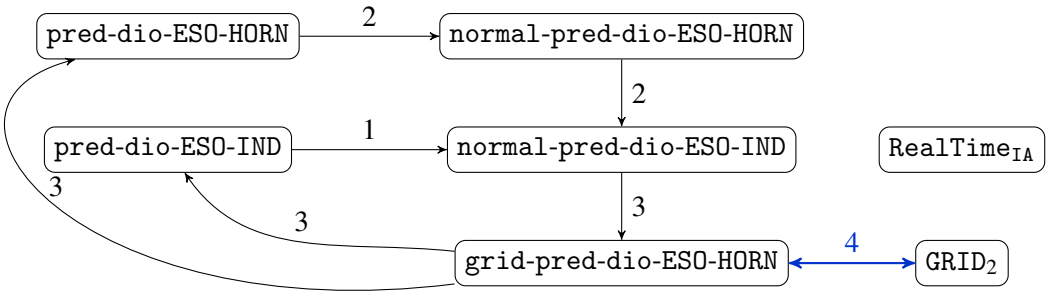


FIGURE 3.13 – Égalité de grid-pred-dio-ESO-HORN et $\mathtt{GRID}_2$

Égalité des classes $\mathtt{GRID}_2$ et $\mathtt{RealTime}_{\mathtt{IA}}$

L'égalité de $\mathtt{GRID}_2$ et $\mathtt{RealTime}_{\mathtt{IA}}$ est prouvée en deux temps. On montrera d'abord comment le calcul d'un circuit grille $\mathtt{GRID}_2$ est simulé par un IA ("Iterative Array" : automate cellulaire avec entrée séquentielle) travaillant en temps-réel. Puis, dans un second temps, on montrera comment le circuit-grille $\mathtt{GRID}_2$ simule un tel automate.

$\mathtt{GRID}_2 \subseteq \mathtt{RealTime}_{\mathtt{IA}}$

La simulation du circuit-grille par l'IA est réalisée en 3 étapes comme le montre la Figure 3.14. La première étape consiste à appliquer aux sites $(x,y) \in [1,n]^2$ de la grille le changement de variables suivant : $(x,y) \mapsto (x-y+1, x+y-1)$. On se retrouve alors avec un circuit divisé par la droite $x = 1$ en deux espaces de travail symétriques.
La deuxième étape consiste alors à "plier" le circuit selon cet axe, méthode que l'on retrouve dans la littérature des automates cellulaires [48].
Enfin, la dernière étape utilise une autre méthode usuelle de la littérature qui consiste à regrouper plusieurs sites en un seul par un ajout de redondance [48]. Chaque site $(c,t) = (\lceil c'/2 \rceil, \lceil t'/2 \rceil)$ (où $\lceil x \rceil$ désigne l'arrondi supérieur du nombre x) du diagramme espace-temps de l'IA contient l'information de l'ensemble des sites $\{(c'-1,t'-1),(c',t'),(c'+1,t'-1)\}$ du circuit après l'étape 2, pour c' et t' impairs et celle de l'ensemble $\{(c',t'),(c'+1,t'-1)\}$ pour $c' = 1$ et $t' > 1$ impair. L'information du site $(1,1)$ n'est pas modifiée par la transformation.

$\mathtt{RealTime}_{\mathtt{IA}} \subseteq \mathtt{GRID}_2$

La figure 3.15 montre comment simuler le temps-réel d'un IA sur le circuit-grille $\mathtt{GRID}_2$. Chaque site (c,t) du diagramme espace-temps de l'IA est envoyé sur le site $(c+t-1, t-c+1)$ du circuit-grille. Les communications sont ensuite segmentées pour rentrer dans le cadre du circuit-grille.

Comme on vient de le voir à travers ces deux inclusions, on a bien égalité entre $\mathtt{GRID}_2$ et $\mathtt{RealTime}_{\mathtt{IA}}$. On peut donc ajouter cette dernière égalité à notre schéma (Figure 3.16) et conclure la preuve du théorème 3.0.2.

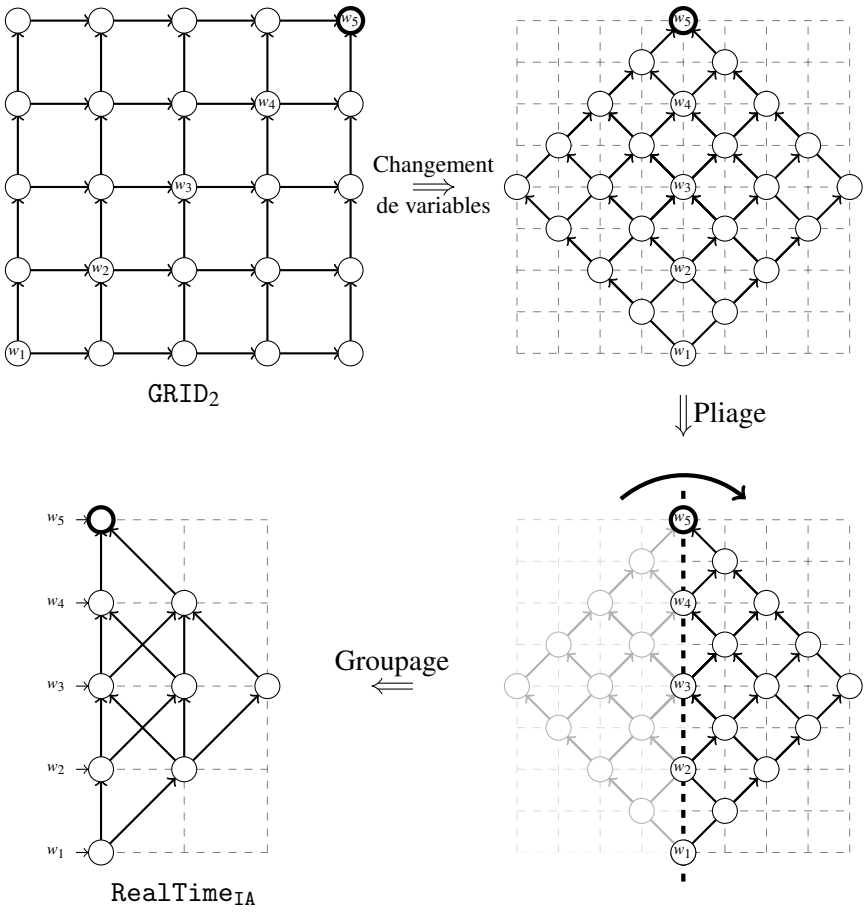


FIGURE 3.14 – Inclusion de $\texttt{GRID}_2$ dans $\texttt{RealTime}_{\texttt{IA}}$

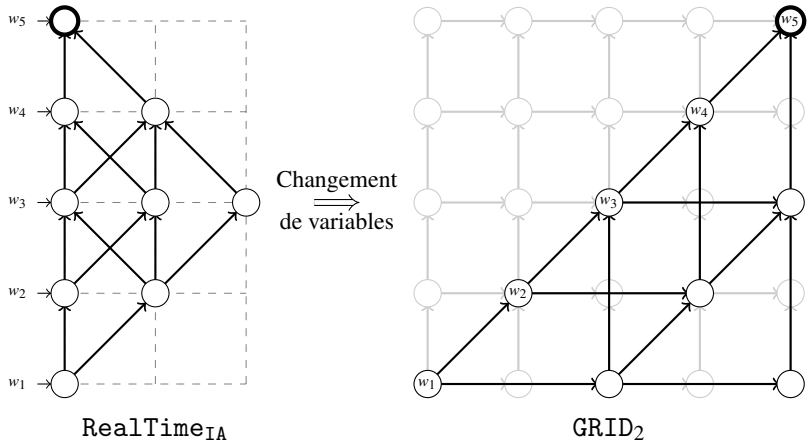


FIGURE 3.15 – Inclusion réciproque de $\texttt{RealTime}_{\texttt{IA}}$ dans $\texttt{GRID}_2$

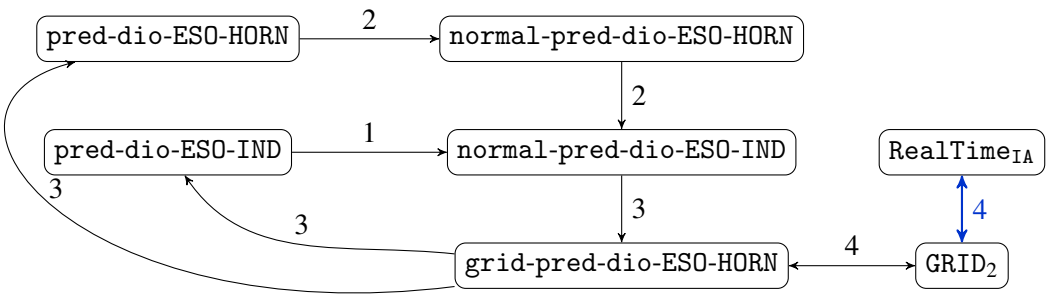


FIGURE 3.16 – Égalité entre $\mathtt{GRID}_2$ et $\mathtt{RealTime}_{\mathtt{IA}}$

3.3 Égalité entre les classes incl-ESO-HORN, incl-ESO-IND et Trellis

On prouve dans cette section le théorème 3.0.3 établissant l'égalité des classes incl-ESO-HORN, incl-ESO-IND et Trellis. Comme lors de la preuve des théorèmes 3.0.1 et 3.0.2, on introduit trois nouvelles logiques :

1. normal-incl-ESO-HORN correspondant aux formules de incl-ESO-HORN une fois normalisées ;
2. normal-incl-ESO-IND correspondant aux formules de incl-ESO-IND normalisées ;
3. grid-incl-ESO-HORN dont chaque formule est en en directe bijection avec un circuit-grille.

La figure 3.17 montre la figure 3.1 adaptée à la preuve du théorème 3.0.3.

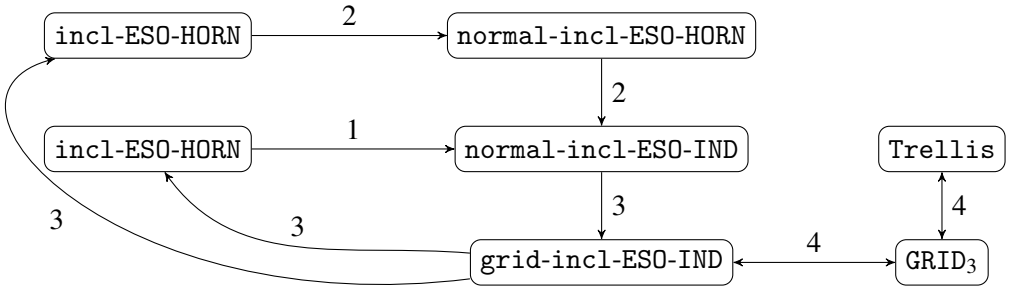


FIGURE 3.17 – Schéma de la preuve d'équivalence entre incl-ESO-HORN, incl-ESO-IND et Trellis

3.3.1 Inclusion de incl-ESO-IND dans normal-incl-ESO-IND

On définit ici la logique normal-incl-ESO-IND correspondant aux formules de incl-ESO-IND normalisées.

Définition 3.3.1 Une formule de normal-incl-ESO-IND est une formule $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ où :
$\mathbf{R} = \{R_1, \ldots, R_m\}$ est un ensemble de prédicats binaires ;
$\psi_0 := \bigwedge_{j=1}^{m} \bigwedge_{\alpha \in A_j} (\alpha \rightarrow R_j(x,y))$ est une conjonction de clauses de la forme suivante où $s \in \Sigma$, $R_h \in \mathbf{R}$ et A, B sont des parties (possiblement vides) de $[1,m]$:
(a) $x = y \wedge Q_s(y) \rightarrow R_h(x,y)$;
(b) $x < y \wedge \bigwedge_{j \in A} (\neg) R_j(x+1,y) \wedge \bigwedge_{j \in B} (\neg) R_j(x,y-1) \rightarrow R_h(x,y)$;
$E(\psi_0) := \bigwedge_{j=1}^{m} (R_j(x,y) \leftrightarrow \bigvee_{\alpha \in A_j} \alpha)$ est la conjonction d'équivalences obtenue à partir de ψ_0 ;
ψ_{neg} est une conjonction de clauses de la forme :
(c) $x = 1 \wedge y = n \wedge R_h(x,y) \rightarrow \bot$, avec $R_h \in \mathbf{R}$.
On note ψ la conjonction de clauses $\psi_0 \wedge \psi_{neg}$ de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq, <\}$.

Le but de cette sous-section est de prouver l'inclusion de incl-ESO-IND dans normal-incl-ESO-IND, ce que dit la proposition suivante :

Proposition 3.3.1 Toute formule $\Phi \in$ incl-ESO-IND est équivalente à une formule $\Phi' \in$ normal-incl-ESO-IND.

Démonstration. Soit $\Phi := \exists \mathbf{R} \forall x \forall y (E(\psi_0) \wedge \psi_{neg})$ une formule de `incl-ESO-IND`, alors $\psi := \psi_0 \wedge \psi_{neg}$ est une conjonction de clauses de la forme $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_r \rightarrow \beta$, avec $R \in \mathbf{R}$, où les δ_i sont :

- soit un littéral d'entrée : $(\neg)U(x-a)$ ou $(\neg)U(y-a)$, pour $U \in \{(Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{max}\}$ et $a \in \mathbb{Z}$,
- soit une (in)égalité $x = y$ ou $x < y$,
- soit un atome de calcul $S(x,y)$ ou une conjonction de la forme $(\neg)S(x+a, y-b) \wedge x+a \leq y-b$ pour $S \in \mathbf{R}$ et $a, b \geq 0$ tels que $a+b > 0$;

et où la conclusion β est soit un atome de calcul $R(x,y)$ avec $R \in \mathbf{R}$ ou une contradiction $\bot$.

La normalisation des formules de `incl-ESO-IND` diffère de celle des formules de `pred-ESO-IND` ou `pred-dio-ESO-IND` en cela qu'elle ne contient que sept étapes. On retrouve toutefois certaines étapes des précédentes normalisations comme le traitement de la contradiction ou de l'entrée, étapes qui sont ici adaptées pour les formules inclusives.

Les 7 étapes de cette normalisation sont les suivantes :

1. Traitement des clauses de contradiction ;
2. Traitement de l'entrée ;
3. Traitement des littéraux `min` et `max` ;
4. Restriction des atomes de calcul à $(\neg)R(x+1,y)$, $(\neg)R(x,y-1)$ et $R(x,y)$;
5. Élimination des égalités $x = a$ ($a \geq 1$) et $y = n-b$ ($b \geq 0$) ;
6. Élimination des descriptions dans les clauses d'initialisation ;
7. Élimination des atomes $R(x,y)$ comme hypothèses.

1. Traitement des clauses de contradiction

Chaque clause négative $x \leq y \wedge l_1 \wedge \cdots \wedge l_k \rightarrow \bot$ est remplacée de manière équivalente par la conjonction de la clause $x \leq y \wedge l_1 \wedge \cdots \wedge l_k \rightarrow R_\bot(x,y)$ avec la clause $R_\bot(x,y) \rightarrow \bot$ où $R_\bot$ est un nouveau prédicat de calcul. L'information de la contradiction est ensuite transportée jusqu'au point $(1,n)$ comme demandé par la forme normale. La clause $R_\bot \rightarrow \bot$ est remplacée par la conjonction des deux clauses de transport :

- $x < y \wedge R_\bot(x, y-1) \rightarrow R_\bot(x,y)$
- et $x < y \wedge R_\bot(x+1, y) \rightarrow R_\bot(x,y)$;

avec la clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R_\bot(x,y) \rightarrow \bot$.

Après cette étape, chaque clause de la formule est :
- soit une clause de calcul $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée de l'une des formes :
    - $Q_s(x+a)$ ou $Q_s(y+a)$, pour $a \in \mathbb{Z}$,
    - $(\neg)U(x+a)$ ou $(\neg)U(y+a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \in \mathtt{Z}$,
  - soit une (in)égalité $x = y$ ou $x < y$;
  - soit une conjonction de la forme $S(x,y) \wedge x \leq y$ ou $(\neg)S(x+a, y-b) \wedge x+a \leq y-b$, pour $a, b \geq 0$ deux entiers tels que $a+b > 0$.
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

2. Traitement de l'entrée

Le but de cette étape est de rendre les lettres du mot d'entrée uniquement disponibles sur la diagonale $x = y$ et seulement via les littéraux $Q_s(x-a)$ et $Q_s(y+b)$ avec $a, b \geq 0$. Dans cette optique, on introduit de nouveaux prédicats de calcul W_s^{x+a} et W_s^{y+a}, pour tout $s \in \Sigma$ et $a \in \mathbb{Z}$, avec le sens intuitif : $W_s^{x+a}(x,y) \iff Q_s(x+a) \wedge 1 \leq x+a \leq n$ (resp. $W_s^{y+a}(x,y) \iff Q_s(y+a) \wedge 1 \leq y+a \leq n$).

Ces prédicats sont définis inductivement par :

les clauses d'initialisation (le long de la diagonale) : pour $s \in \Sigma$ et $a \geq 1$,

- $x = y \wedge Q_s(x) \rightarrow W_s^x(x,y)$; $x = y \wedge Q_s(x) \rightarrow W_s^y(x,y)$,
- $x = y \wedge Q_s(x-a) \wedge x > a \rightarrow W_s^{x-a}(x,y)$,
- $x = y \wedge Q_s(x-a) \wedge x > a \rightarrow W_s^{y-a}(x,y)$,
- $x = y \wedge Q_s(y+a) \wedge y \leq n-a \rightarrow W_s^{x+a}(x,y)$,
- $x = y \wedge Q_s(y+a) \wedge y \leq n-a \rightarrow W_s^{y+a}(x,y)$;

les clauses de transport : pour $a \in \mathbb{Z}$,

- $x < y \wedge W_s^{x+a}(x,y-1) \rightarrow W_s^{x+a}(x,y)$ et
- $x < y \wedge W_s^{y+a}(x+1,y) \rightarrow W_s^{y+a}(x,y)$.

Ces nouvelles clauses permettent de remplacer chaque atome $Q_s(x+a)$ (resp. $Q_s(y+a)$) où $a \in \mathbb{Z}$, par l'atome de calcul $W_s^{x+a}(x,y)$ (resp. $W_s^{y+a}(x,y)$) dans toutes les clauses, exceptées les clauses d'initialisation.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation : $x = y \wedge Q_s(x) \rightarrow R(x,y)$, $x = y \wedge Q_s(x-a) \wedge x > a \rightarrow R(x,y)$ ou $x = y \wedge Q_s(y+a) \wedge y \leq n-a \rightarrow R(x,y)$ pour $a \geq 0$;
- soit une clause de calcul $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ où chaque δ_i est :
  - soit un littéral d'entrée : $(\neg)U(x+a)$ ou $(\neg)U(y+a)$, pour $U \in \{\texttt{min}, \texttt{max}\}$ et $a \in \mathtt{Z}$,
  - soit une (in)égalité $x = y$ ou $x < y$;
  - soit une conjonction de la forme $S(x,y) \wedge x \leq y$ ou $(\neg)S(x+a,y-b) \wedge x+a \leq y-b$, pour $a,b \geq 0$ deux entiers tels que $a+b > 0$.
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

3. Traitement des littéraux `min` et `max`

A ce stade, on peut considérer que les seuls littéraux sur x faisant intervenir `min` ou `max` sont des égalités $x = a$ ou $x = n-b$ et des inégalités $x > a$ ou $x < n-b$, pour a et b des entiers fixés avec $a \geq 1$ et $b \geq 0$; de même pour y. Par un processus similaire à la partie précédente qui traitait les littéraux d'entrée Q_s, on va ici rendre l'information à propos de `min` et `max` uniquement disponible sur la diagonale $x = y$ et uniquement à travers les inégalités et égalités suivantes : $x = a$ ou $x > a$, avec $a \geq 1$, ou $y = n-b$ ou $y < n-b$, avec $b \geq 0$. On introduit dans ce but, de nouveaux prédicats binaires définis inductivement : $R^{x=a}$, $R^{x>a}$, pour $a \geq 1$, et $R^{x=n-a}$, $R^{x<n-a}$, pour $a \geq 0$, de même pour y. Le sens intuitif de ces prédicats est évident : par exemple, $R^{x>a}(x,y) \iff x > a$. La définition de ces prédicats suit le même schéma initialisation-transport que les prédicats W_s^{x+a} et W_s^{y+a}.

Le prédicat $R^{x=a}$ est, par exemple, défini par les deux clauses :
Initialisation : $x = y \wedge x = a \rightarrow R^{x=a}(x,y)$;
Transport : $x < y \wedge R^{x=a}(x,y-1) \rightarrow R^{x=a}(x,y)$.

Le prédicat $R^{x<n-a}$ est lui défini par les deux clauses :
Initialisation : $x = y \wedge y < n-a \rightarrow R^{x<n-a}(x,y)$;
Transport : $x < y \wedge R^{x<n-a}(x,y-1) \rightarrow R^{x<n-a}(x,y)$.

Ceci nous permet de remplacer les atomes $x = a$, $x > a$, $x = n-a$, $x < n-a$ par leurs atomes de calculs respectifs $R^{x=a}(x,y)$, $R^{x>a}(x,y)$, $R^{x=n-a}(x,y)$, $R^{x<n-a}(x,y)$, dans toutes les clauses de calcul. On effectue le même traitement pour y. Afin de réduire les différents cas à traiter, on suppose sans perte de généralité que les inégalités $x > a$ avec $a \geq 1$ (resp. $y < n-b$ avec $b \geq 0$) dans les clauses d'initialisation sont toutes accompagnées d'un atome d'entrée $Q_s(x-a)$ (resp. $Q_s(y+b)$). En effet, l'atome $x > a$ est équivalent à la disjonction $\bigvee_{s \in \Sigma} x > a \wedge Q_s(x-a)$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge \delta \to R(x,y)$, où δ est
  - soit un atome d'entrée $Q_s(x)$;
  - soit une égalité $x = a$ ou $y = n - b$ avec $a \geq 1$ et $b \geq 0$;
  - soit une conjonction $Q_s(x-a) \wedge x > a$ ou $Q_s(y+a) \wedge y \leq n-a$, pour $a \geq 1$;
- soit une clause de calcul $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_r \to R(x,y)$ avec $R \in \mathbf{R}$ et où les δ_i sont une conjonction de l'une des formes $S(x,y) \wedge x \leq y$ ou $(\neg)S(x+a,y-b) \wedge x+a \leq y-b$, pour $a,b \geq 0$ deux entiers tels que $a+b > 0$.
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

4. Restriction des atomes de calcul à $(\neg)R(x+1,y)$, $(\neg)R(x,y-1)$ et $R(x,y)$

Cette étape est une variante de la même étape dans la normalisation de la logique du prédécesseur (étape 3). On introduit de nouveaux prédicats de "décalage" R^{x+a}, R^{y-b}, $R^{x+a,y-b}$, $R^{\neg,x+a}$, $R^{\neg,y-b}$ et $R^{\neg,x+a,y-b}$ pour $a,b > 0$ et $R \in \mathbf{R}$ dont le sens intuitif est trivial : par exemple, $R^{\neg,x+a,y+b} \iff x+a \leq y-b \wedge \neg R(x+a,y-b)$. La définition de ces prédicats est simple car similaire au cas de la logique du prédécesseur. Par exemple, la conjonction des deux clauses :
- $x < y \wedge \neg S(x,y-1) \to S^{\neg,y-1}(x,y)$
- et $x < y \wedge S^{\neg,y-1}(x+1,y) \to S^{\neg,x+1,y-1}(x,y)$;

définit le prédicat $S^{\neg,x+1,y-1}$ à partir des prédicats S et $S^{\neg,y-1}$.

Comme dans la normalisation de la logique du prédécesseur (fin de l'étape 5), on introduit de nouveaux prédicats intermédiaires afin de réduire chaque clause de calcul à l'une des formes suivantes :
- $x < y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$ ou $x = y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$;
- $x < y \wedge (\neg)S(x+1,y) \to R(x,y)$;
- $x < y \wedge (\neg)S(x,y-1) \to R(x,y)$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge \delta \to R(x,y)$ où δ est
  - soit un atome d'entrée $Q_s(x)$;
  - soit une égalité $x = a$ ou $y = n - b$ avec $a \geq 1$ et $b \geq 0$;
  - soit une conjonction $Q_s(x-a) \wedge x > a$ ou $Q_s(y+a) \wedge y \leq n-a$ pour $a \geq 1$;
- soit une clause de calcul de l'une des formes :
  - $x < y \wedge (\neg)S(x+1,y) \to R(x,y)$;
  - $x < y \wedge (\neg)S(x,y-1) \to R(x,y)$;
  - $x < y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$ ou $x = y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$;
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

Les étapes 5 et 6 utilisent la notion de description définie ci-dessous.

Définition 3.3.2 — x-description et y-description. Soit A (resp. B) le maximum des entiers a (resp. b) qui apparaissent dans les égalités $x = a$ (resp. $y = n-b$) et les conjonctions $Q_s(x-a) \wedge x > a$ (resp. $Q_s(y+b) \wedge y \leq n-b$) des clauses d'initialisation. On appelle une *x-description* toute conjonction de la forme
- $x = a \wedge \bigwedge_{i=1}^{a-1} Q_{s_i}(x-i)$, pour $a \in [1,A]$ et $(s_1,\ldots,s_a) \in \Sigma^a$,
- ou $x > A \wedge \bigwedge_{i=1}^{A} Q_{s_i}(x-i)$, pour $(s_1,\ldots,s_A) \in \Sigma^A$.

La première forme de conjonction est appelée une *x-égal-description* et la deuxième une *x-sup-description*. De la même façon, une *y-description* est une conjonction
- $y = n-b \wedge \bigwedge_{j=1}^{b} Q_{t_j}(y+j)$, pour $b \in [0,B-1]$ et $(t_1,\ldots,t_b) \in \Sigma^b$ (*y-égal-description*),
- ou $y \leq n-B \wedge \bigwedge_{j=1}^{B} Q_{t_j}(y+j)$, pour $(t_1,\ldots,t_B) \in \Sigma^B$ (*y-inf-description*).

Soit H l'ensemble des x-descriptions et Θ l'ensemble des y-descriptions. On notera deux cas

spéciaux qui sont la x-description où $x = 1$ et la y-description où $y = n$. On donne ici les caractéristiques importantes des x-descriptions et y-descriptions :
- $x = a$ (resp. $Q_s(x-a) \wedge x > a$), pour $a \in [1,A]$, est équivalent à la disjonction $H^{x=a}$ (resp. H_s^{x-a}) des x-descriptions qui contiennent $x = a$ (resp. $Q_s(x-a)$) ;
- similairement, $y = n-b$, pour $b \in [0,B]$ (resp. $Q_t(y+b) \wedge y \leq n-b$, pour $b \in [1,B]$), est équivalent à la disjonction $\Theta^{y=n-b}$ (resp. Θ_t^{y+b}) des y-descriptions qui contiennent $y = n-b$ (resp. $Q_t(y+b)$) ;
- les x-descriptions (resp. y-descriptions) couvrent tous les cas possibles, i.e., les disjonctions $\bigvee_{\eta \in H} \eta$ et $\bigvee_{\theta \in \Theta} \theta$ sont des tautologies.

5. Élimination des égalités $x = a$ ($a \geq 1$) et $y = n-b$ ($b \geq 0$)

On remplace de façon équivalente chaque clause d'initialisation $x = y \wedge \delta \rightarrow R(x,y)$ où δ est l'égalité $x = a$ (resp. la conjonction $Q_s(x-a) \wedge x > a$) par la conjonction des clauses $x = y \wedge \eta \wedge \theta \rightarrow R(x,y)$ où $\eta \in H^{x=a}$ (resp. $\eta \in H_s^{x-a}$) et $\theta \in \Theta$.
De la même manière, on remplace chaque clause d'initialisation $x = y \wedge \delta \rightarrow R(x,y)$ où δ est $y = n-b$ (resp. $y \leq n-b \wedge Q_t(y+b)$) par la conjonctions des clauses $x = y \wedge \eta \wedge \theta \rightarrow R(x,y)$ où $\eta \in H$ et $\theta \in \Theta^{y=n-b}$ (resp. $\theta \in \Theta_t^{y+b}$).
Ainsi, chaque clause d'initialisation est maintenant de la forme $x = y \wedge Q_s(x) \rightarrow R(x,y)$ ou $x = y \wedge \eta \wedge \theta \rightarrow R(x,y)$ avec $\eta \in H$ et $\theta \in \Theta$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge \delta \rightarrow R(x,y)$, où δ est
  - soit un atome d'entrée $Q_s(x)$;
  - soit une conjonction $\eta \wedge \theta$ où η est une x-description et θ une y-description ;
- soit une clause de calcul de l'une des formes :
  - $x < y \wedge (\neg)S(x+1,y) \rightarrow R(x,y)$;
  - $x < y \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$;
  - $x < y \wedge S(x,y) \wedge T(x,y) \rightarrow R(x,y)$ ou $x = y \wedge S(x,y) \wedge T(x,y) \rightarrow R(x,y)$;
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \rightarrow \bot$.

6. Élimination des descriptions dans les clauses d'initialisation

Le but de cette étape est d'éliminer les x-descriptions et y-descriptions dans les clauses d'initialisation (introduites à l'étape 5) afin que celles-ci soient toutes de la forme

$$x = y \wedge Q_s(x) \rightarrow R(x,y).$$

La méthode que l'on va mettre en oeuvre ici généralise la méthode de *calcul sous hypothèses* utilisée à l'étape 7 de la normalisation des formules de pred-ESO-IND pour la suppression des littéraux $(\neg)\texttt{max}(y)$ dans les clauses d'initialisation. Pour cela on avait introduit deux prédicats $R_{\leftarrow}^{\texttt{max}}$ et $R_{\leftarrow}^{\neg\texttt{max}}$ pour chaque prédicat de calcul R, de signification intuitive suivante : $R_{\leftarrow}^{(\neg)\texttt{max}}(x,y) \iff ((\neg)\texttt{max}(y) \rightarrow R(x,y))$.

Similairement, on introduit ici de nouveaux prédicats de calcul $R_{\leftarrow}^{\eta,\theta}$, pour chaque $R \in \mathbf{R}$, chaque x-description $\eta \in H$ et chaque y-description $\theta \in \Theta$. Ces nouveaux prédicats ont la signification intuitive suivante :

$$R_{\leftarrow}^{\eta,\theta}(x,y) \iff (\eta(x) \wedge \theta(y) \rightarrow R(x,y)).$$

On va maintenant transformer toutes les clauses en utilisant les prédicats $R_{\leftarrow}^{\eta,\theta}$ à la place des prédicats de calcul initiaux R.

Transformation des clauses d'initialisation : On remplace chaque clause
$x = y \wedge Q_s(x) \to R(x,y)$ par la conjonction de clauses
$\bigwedge_{(\eta,\theta)\in H\times\Theta}(x = y \wedge Q_s(x) \to R^{\eta,\theta}_{\leftarrow}(x,y))$, et chaque clause
$x = y \wedge \eta(x) \wedge \theta(y) \to R(x,y)$ qui est "équivalente" à la clause $x = y \to R^{\eta,\theta}_{\leftarrow}(x,y)$ par la conjonction de clauses $\bigwedge_{s\in\Sigma}(x = y \wedge Q_s(x) \to R^{\eta,\theta}_{\leftarrow}(x,y))$.

Transformation des clauses de calcul : On transforme les clauses de calcul suivant la forme de leurs hypothèses.

`Clauses` $x < y \wedge \neg S(x+1,y) \to R(x,y)$:
On utilise les trois faits suivants :
- $\neg S(x+1,y)$ est équivalent à $\bigvee_{\eta\in H,\theta\in\Theta}(\eta(x+1) \wedge \theta(y) \wedge \neg S(x+1,y))$;
- $\eta(x+1) \wedge \theta(y) \wedge \neg S(x+1,y)$ est équivalent à $\neg(\eta(x+1) \wedge \theta(y) \to S(x+1,y))$, qui peut se réécrire $\neg S^{\eta,\theta}_{\leftarrow}(x+1,y)$;
- $R(x,y)$ est équivalent à la conjonction $\bigwedge_{\eta\in H,\theta\in\Theta}(\eta(x) \wedge \theta(y) \to R(x,y))$, qui peut se réécrire $\bigwedge_{\eta\in H,\theta\in\Theta} R^{\eta,\theta}_{\leftarrow}(x,y)$.

Ces faits justifient le remplacement de la clause $x < y \wedge \neg S(x+1,y) \to R(x,y)$ par la conjonction des clauses $x < y \wedge \neg S^{\eta,\theta}_{\leftarrow}(x+1,y) \to R^{\eta',\theta'}_{\leftarrow}(x,y)$, pour tous $\eta,\eta' \in H$ et tous $\theta,\theta' \in \Theta$.

`Clauses` $x < y \wedge S(x+1,y) \to R(x,y)$:
Ces clauses sont les plus difficiles à traiter, surtout à cause des notations. On note $\eta^{x=a,x-1,\ldots,x-(a-1)}_{s_1,\ldots,s_{a-1}}(x)$ la *x-égal-description*

$$x = a \wedge \textstyle\bigwedge_{i=1}^{a-1} Q_{s_i}(x-i), \text{ avec } a \in [1,A] \text{ et } (s_1,\ldots,s_{a-1}) \in \Sigma^a.$$

De même, on note $\eta^{x>A,x-1,\ldots,x-A}_{s_1,\ldots,s_A}(x)$ la *x-sup-description*

$$x > A \wedge \textstyle\bigwedge_{i=1}^{A} Q_{s_i}(x-i), \text{ avec } (s_1,\ldots,s_A) \in \Sigma^A.$$

Les équivalences suivantes sont obtenues par décalage de l'information de $x+1$ à x :

$$\eta^{x=a,x-1,\ldots,x-(a-1)}_{s_1,\ldots,s_{a-1}}(x+1) \iff (Q_{s_1}(x) \wedge \eta^{x=a-1,x-1,\ldots,x-(a-2)}_{s_2,\ldots,s_{a-1}}(x)) \tag{3.14}$$

$$\eta^{x>A,x-1,\ldots,x-A}_{s_1,\ldots,s_A}(x+1) \iff$$

$$Q_{s_1}(x) \wedge \left(\eta^{x=A,x-1,\ldots,x-(A-1)}_{s_2,\ldots,s_A}(x) \vee \bigvee_{s\in\Sigma} \eta^{x>A,x-1,\ldots,x-(A-1),x-A}_{s_2,\ldots,s_A,s}(x)\right) \tag{3.15}$$

Justifions précisément l'équivalence 3.15 (l'équivalence 3.14 est plus facile à justifier). La notation $\eta^{x>A,x-1,\ldots,x-A}_{s_1,\ldots,s_A}(x+1)$ désigne la conjonction

$$x+1 > A \wedge \textstyle\bigwedge_{i=1}^{A} Q_{s_i}(x+1-i)$$

qu'on peut écrire encore : $x > A-1 \wedge Q_{s_1}(x) \wedge \bigwedge_{j=1}^{A-1} Q_{s_{j+1}}(x-j)$. Cette conjonction est équivalente à la disjonction de conjonctions

$$\left(x = A \wedge Q_{s_1}(x) \wedge \bigwedge_{j=1}^{A-1} Q_{s_{j+1}}(x-j)\right) \vee \bigvee_{s\in\Sigma}\left(x > A \wedge Q_{s_1}(x) \wedge \bigwedge_{j=1}^{A-1} Q_{s_{j+1}}(x-j) \wedge Q_s(x-A)\right)$$

qu'on peut réécrire sous la forme voulue (avec les *x-descriptions*) :

$$Q_{s_1}(x) \wedge \left(\eta^{x=A,x-1,\ldots,x-(A-1)}_{s_2,\ldots,s_A}(x) \vee \bigvee_{s\in\Sigma} \eta^{x>A,x-1,\ldots,x-(A-1),x-A}_{s_2,\ldots,s_A,s}(x)\right), CQFD.$$

Notation 3.5 • Pour toute *x-égal-description* $\eta := \eta^{x=a,x-1,\ldots,x-(a-1)}_{s_1,\ldots,s_{a-1}}$, on note $\eta_\triangleright$ la *x-égal-description* $\eta^{x=a-1,x-1,\ldots,x-(a-2)}_{s_2,\ldots,s_{a-1}}$, "décalée" de η, présente en partie droite de l'équivalence 3.14. Avec ces notations, l'équivalence 3.14 est réécrite ainsi :

$$1)\ \eta(x+1) \iff (Q_{s_1}(x) \wedge \eta_\triangleright(x)).$$

• Pour toute *x-sup-description* $\eta := \eta_{s_1,\ldots,s_A}^{x>A,x-1,\ldots,x-A}$, on a deux *x-descriptions* possibles,"décalées" de η, correspondant aux deux formes des opérandes (disjuncts) de la disjonction présente en partie droite de l'équivalence 3.15 :
— on note $\eta_{\triangleright}^{x=A}$ la *x-égal-description* $\eta_{s_2,\ldots,s_A}^{x=A,x-1,\ldots,x-(A-1)}$
— et, pour $s \in \Sigma$, on note $\eta_{\triangleright s}^{x>A}$ la *x-sup-description* $\eta_{s_2,\ldots,s_A,s}^{x>A,x-1,\ldots,x-A}$,
Avec ces notations, l'équivalence 3.15 se réécrit :

$$2)\ \eta(x+1) \iff \big(Q_{s_1}(x) \wedge (\eta_{\triangleright}^{x=A}(x) \vee \bigvee_{s\in\Sigma} \eta_{\triangleright s}^{x>A}(x))\big).$$

Comme on va le vérifier, les équivalences 3.14 et 3.15 justifient que chaque clause

$$x < y \wedge S(x+1,y) \rightarrow R(x,y)$$

soit remplacée par la conjonction des clauses (a) et (b) suivantes :

a) $\bigwedge_{\theta\in\Theta} \big(x < y \wedge S_{\leftarrow}^{\eta,\theta}(x+1,y) \wedge W_{s_1}^x(x,y) \rightarrow R_{\leftarrow}^{\eta_{\triangleright},\theta}(x,y)\big)$, pour tout $s_1 \in \Sigma$ et toute *x-égal-description* η qui contient $Q_{s_1}(x-1)$ (voir équivalence 3.14),

b) pour tout $s_1 \in \Sigma$ et toute *x-sup-description* η qui contient $Q_{s_1}(x-1)$ (voir équivalence 3.15), les clauses (b1) et (b2) :

b1) $\bigwedge_{\theta\in\Theta} \Big(x < y \wedge W_{s_1}^x(x,y) \wedge S_{\leftarrow}^{\eta,\theta}(x+1,y) \rightarrow R_{\leftarrow}^{\eta_{\triangleright}^{x=A},\theta}(x,y)\Big)$;

b2) $\bigwedge_{\theta\in\Theta,s\in\Sigma} \Big(x < y \wedge W_{s_1}^x(x,y) \wedge S_{\leftarrow}^{\eta,\theta}(x+1,y) \rightarrow R_{\leftarrow}^{\eta_{\triangleright s}^{x>A},\theta}(x,y)\Big)$.

Rappel : L'atome $W_{s_1}^x(x,y)$ signifie $Q_{s_1}(x)$.

Vérification : Expliquons la signification des clauses (a) et (b) de façon simplifiée, c'est-à-dire sans mentionner l'hypothèse $\theta(y)$, *y-description* qui reste inchangée car "non décalée".
La clause (a) pour une *x-égal-description* η contenant $Q_{s_1}(x-1)$ signifie
$x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow (\eta_{\triangleright}(x) \rightarrow R(x,y))$,
qui, par une substitution justifiée par l'équivalence 3.14, est équivalente à l'implication
$x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow (\eta(x+1) \rightarrow R(x,y))$ comme attendu.
Les clauses (b1) et (b2) pour une *x-sup-description* η contenant $Q_{s_1}(x-1)$ signifient
b1) $x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow (\eta_{\triangleright}^{x=A}(x) \rightarrow R(x,y))$ et
b2) $x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow (\eta_{\triangleright s}^{x>A}(x) \rightarrow R(x,y))$, pour $s \in \Sigma$. La conjonction des clauses (b1) et (b2) pour η équivaut à l'implication
$x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow ((\eta_{\triangleright}^{x=A}(x) \vee \bigvee_{s\in\Sigma} \eta_{\triangleright s}^{x>A}(x)) \rightarrow R(x,y))$, qui, par une substitution justifiée par l'équivalence 3.15, est équivalente à l'implication attendue
$x < y \wedge (\eta(x+1) \rightarrow S(x+1,y)) \wedge Q_{s_1}(x) \rightarrow (\eta(x+1) \rightarrow R(x,y))$.

`Clauses` $x < y \wedge (\neg)S(x,y-1) \rightarrow R(x,y)$: par symétrie de y par rapport à x, ces clauses sont transformées par une adaptation de la méthode ci-dessus.

`Clauses` $x \preceq y \wedge S(x,y) \wedge T(x,y) \rightarrow R(x,y)$, `où` $\preceq \in \{<,=\}$: une telle clause est remplacée par la conjonction des clauses :
$x \preceq y \wedge S_{\leftarrow}^{\eta,\theta}(x,y) \wedge T_{\leftarrow}^{\eta,\theta}(x,y) \rightarrow R_{\leftarrow}^{\eta,\theta}(x,y)$, pour $(\eta,\theta) \in H \times \Theta$.

Transformation des clauses de contradiction : Chaque clause
$x = 1 \wedge y = n \wedge R(x,y) \rightarrow \bot$ avec $R \in \mathbf{R}$ est équivalente à la clause
$x = 1 \wedge y = n \wedge (x = 1 \wedge y = n \rightarrow R(x,y)) \rightarrow \bot$. Cette clause est donc remplacée par $x = 1 \wedge y = n \wedge R_{\leftarrow}^{x=1,y=n}(x,y) \rightarrow \bot$.

Après cette étape, chaque clause de la formule est :
- soit une clause d'initialisation $x = y \wedge Q_s(x) \to R(x,y)$;
- soit une clause de calcul de l'une des formes :
  - $x < y \wedge (\neg)S(x+1,y) \to R(x,y)$;
  - $x < y \wedge (\neg)S(x,y-1) \to R(x,y)$;
  - $x < y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$ ou $x = y \wedge S(x,y) \wedge T(x,y) \to R(x,y)$;
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R(x,y) \to \bot$.

7. Élimination des atomes $R(x,y)$ comme hypothèses

Cette étape est la même que l'étape similaire de chacune des deux sections précédentes : étape (8) pour `pred-ESO-IND` (ou `pred-ESO-HORN`), étape (9) pour `pred-dio-ESO-IND` (ou `pred-dio-ESO-HORN`). On utilise la même méthode de substitution utilisant l'acyclicité de notre formule pour supprimer les atomes $R(x,y)$ dans les hypothèses des clauses de calcul. Après cette étape chaque clause de la formule ψ de signature $\mathbf{R} \cup S_\Sigma$ où $\mathbf{R} = \{R_1, \dots, R_n\}$ est :
- soit une clause d'initialisation $x = y \wedge Q_s(x) \to R_h(x,y)$;
- soit une clause de calcul $x < y \wedge \bigwedge_{j \in A}(\neg)R_j(x+1,y) \wedge \bigwedge_{j \in B}(\neg)R_j(x,y-1) \to R_h(x,y)$, avec A et B des parties (possiblement vides) de $[1,m]$;
- soit une clause de contradiction $\texttt{min}(x) \wedge \texttt{max}(y) \wedge R_h(x,y) \to \bot$.

Ces trois types de clauses correspondent bien aux formes demandées par la définition de `normal-incl-ESO-IND`, ce qui conclut la preuve de la proposition 3.3.1. □

On peut donc établir la première inclusion de notre schéma comme montré Figure 3.18

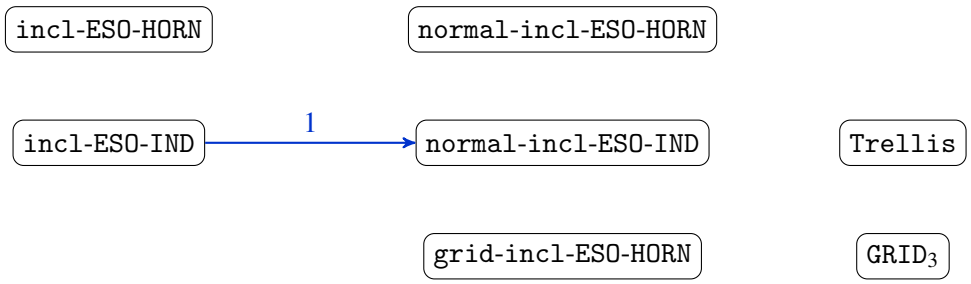


FIGURE 3.18 – Inclusion de `incl-ESO-IND` dans `normal-incl-ESO-IND`

3.3.2 Inclusion de `incl-ESO-HORN` dans `normal-incl-ESO-HORN`

On commence par définir formellement la logique `normal-incl-ESO-HORN`.

Définition 3.3.3 Une formule de `normal-incl-ESO-HORN` est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \psi(x,y)$ où :
$\mathbf{R} = \{\mathbf{R_1}, \dots, \mathbf{R_m}\}$ est un ensemble de prédicats binaires ;
ψ est une conjonction de clauses des formes suivantes, où $s \in \Sigma$, $R_h \in \mathbf{R}$, et A, B sont des parties (possiblement vides) de $[1,m]$:
(a) $x = y \wedge Q_s(y) \to R_h(x,y)$;
(b) $x < y \wedge \bigwedge_{j \in A} R_j(x+1,y) \wedge \bigwedge_{j \in B} R_j(x,y-1) \to R_h(x,y)$, avec A et B des parties (possiblement vides) de $[1,m]$;
(c) $x = 1 \wedge y = n \wedge R_h(x,y) \to \bot$.

Proposition 3.3.2 Toute formule Φ de incl-ESO-HORN est équivalente à une formule Φ' de normal-incl-ESO-HORN.

Démonstration. Comme dans les deux sections précédentes, afin de normaliser les formules de incl-ESO-HORN on commence par appliquer les 6 premières étapes de la normalisation de la version inductive de ces formules. La suppression des hypothèses $R(x,y)$ est, quant à elle, réalisée de la même façon que pour les logiques pred-ESO-HORN et pred-dio-ESO-HORN avec la seule différence suivante. Dans la preuve de l'implication $\Phi' \Rightarrow \Phi$ du lemme 3.1.5, on démontre par récurrence sur l'entier $t \in [-1, n-1]$ que le modèle minimal $(\langle w \rangle, \mathbf{R})$ de la formule φ' satisfait la "relativisé" de φ qui est maintenant :

$$\varphi_t := \forall x \forall y \left(y - x \leq t \rightarrow \bigwedge_{i \in [1,k]} (\alpha_i(x,y) \rightarrow \theta_i(x,y)) \right).$$

Autrement dit, au lieu de faire la preuve par induction sur la somme $x+y=t \in [1,2n]$, on fait maintenant la preuve par induction sur la différence $y-x=t \in [-1,n-1]$.

On montre ainsi que toute formule de $\Phi \in$ incl-ESO-HORN peut être normalisée en une formule $\Phi' \in$ normal-incl-ESO-HORN équivalente. □

Par ailleurs, les formules de normal-incl-ESO-HORN ne contenant pas d'atomes $R(x,y)$ dans leurs hypothèses, leur appartenance à normal-incl-ESO-IND est évidente. Ces deux nouvelles inclusions sont ajoutées sur le schéma de preuve du théorème 3.0.3 donné par la figure 3.19.

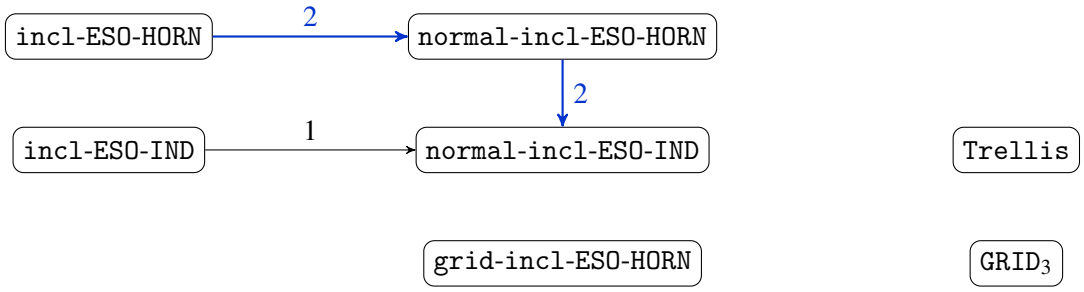


FIGURE 3.19 – Inclusion de incl-ESO-HORN dans normal-incl-ESO-HORN, elle même incluse dans normal-incl-ESO-IND

3.3.3 Inclusion de normal-incl-ESO-IND dans grid-incl-ESO-HORN

Comme pour les formules de normal-pred-ESO-IND et normal-pred-dio-ESO-IND, on montre dans cette section que chaque formule de normal-incl-ESO-IND est équivalente à une formule grille.

Proposition 3.3.3 — logique de grille inclusive. Toute formule $\Phi \in$ normal-incl-ESO-IND est équivalente à une formule de Horn $\Phi' \in$ grid-incl-ESO-HORN, dite *formule grille inclusive*, de la forme

$$\Phi' := \exists \mathbf{T} \forall x \forall y \left[\psi'(x,y) \wedge \bigwedge_{i \in N} (x = 1 \wedge y = n \wedge T_i(x,y) \rightarrow \bot) \right]$$

où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un ensemble de prédicats binaires, N est une partie de $[1, p]$ et ψ' est une conjonction de clauses de Horn des deux formes suivantes :

1. $x = y \wedge Q_s(x) \rightarrow T_{f(s)}(x, y)$;
2. $x < y \wedge T_i(x, y-1) \wedge T_j(x+1, y) \rightarrow T_{g(i,j)}(x, y)$.

Où f et g sont des fonctions ; autrement dit, si deux clauses $\theta \rightarrow \alpha$ et $\theta \rightarrow \alpha'$ ont la même hypothèse θ alors elles ont la *même conclusion* $\alpha = \alpha' = T_{h(\theta)}(x, y)$.

Démonstration. Le passage d'une formule de `normal-incl-ESO-IND` à une formule de `grid-incl-ESO-HORN` se fait de la même manière que pour passer d'une formule de `normal-pred-ESO-IND` à une formule de `grid-pred-ESO-HORN` ou d'une formule de `normal-pred-dio-ESO-IND` à une formule de `grid-pred-dio-ESO-HORN`. La preuve de la proposition 3.3.3 est alors immédiate. □

Par ailleurs, les formules de `grid-incl-ESO-HORN` étant des formules de Horn acycliques, leur inclusion dans `incl-ESO-HORN` et `incl-ESO-IND` est triviale. Ces trois inclusions nous donnent l'équivalence $L \in$ `incl-ESO-HORN` $\iff L \in$ `incl-ESO-IND` comme le montre la Figure 3.21

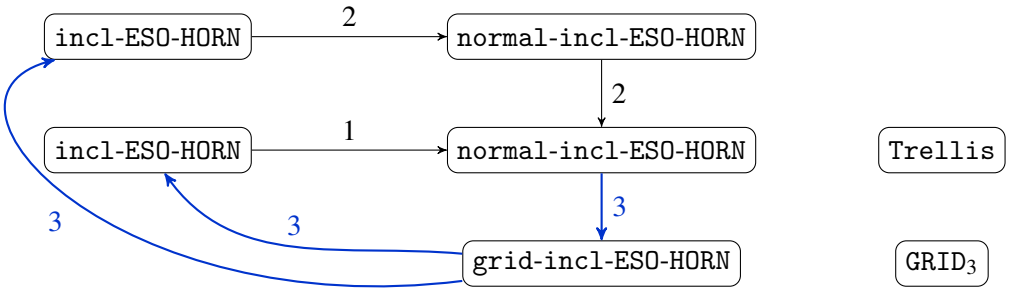


FIGURE 3.20 – Inclusion de `normal-incl-ESO-HORN` dans `grid-incl-ESO-HORN` elle même incluse dans `incl-ESO-HORN` et `incl-ESO-IND`

3.3.4 Égalité de `grid-incl-ESO-HORN`, $\mathtt{GRID}_3$ et `Trellis`

On montre dans cette section deux égalités. La première entre la classe `grid-incl-ESO-HORN` et la classe de complexité du circuit-grille $\mathtt{GRID}_3$ et la seconde entre cette classe et la classe `Trellis`. L'égalité des classes `grid-incl-ESO-HORN`, `incl-ESO-HORN` et `incl-ESO-IND` ayant déjà été montrée dans la sous-section précédente, ces deux nouvelles égalités suffisent pour prouver le théorème 3.0.3.

Égalité de `grid-incl-ESO-HORN` et $\mathtt{GRID}_3$

La proposition 3.3.4 donne une bijection entre les formules de `grid-incl-ESO-HORN` et les automates grilles de $\mathtt{GRID}_3$. La preuve de celle-ci est identique à la preuve de la proposition 3.1.8 et n'est donc pas répétée.

Proposition 3.3.4 Soit

$$\Phi := \exists \mathbf{T} \forall x \forall y \left[\psi(x, y) \wedge \bigwedge_{i \in N} (x = 1 \wedge y = n \wedge T_i(x, y) \rightarrow \bot) \right]$$

une formule grille de `grid-incl-ESO-HORN` définissant un langage L où $\mathbf{T} = \{T_1, \ldots, T_p\}$ est un

ensemble de prédicats binaires, N est une partie de $[1,p]$ et ψ est une conjonction de clauses de Horn des deux formes suivantes :

1. $x = y \wedge Q_s(x) \rightarrow T_{f(s)}(x,y)$;
2. $x < y \wedge T_i(x,y-1) \wedge T_j(x+1,y) \rightarrow T_{g(i,j)}(x,y)$.

Le langage L est reconnu par le circuit-grille $(\Sigma, \mathtt{Input}_n, \mathtt{S}, \mathtt{f}, \mathtt{A})$ de $\mathtt{GRID}_3$ dont l'ensemble des états est $\mathtt{S} := \{q_1, \ldots, q_p\}$, l'ensemble des états acceptants est $\mathtt{A} := \{q_i \mid i \in [1,p] \setminus N\}$ et la fonction de transition $\mathtt{f}$ est donnée par les trois items suivants où la transition (0) s'applique aux sites $(x,y) \in [1,n]^2$ tels que $y < x$. :

0) $\mathtt{f}(\#,\#,\$) = \#$;
1) $\mathtt{f}(\#,\#,s) = q_{f(s)}$;
2) $\mathtt{f}(q_i,q_j,\$) = q_{g(i,j)}$;

Cette bijection établit l'égalité de grid-incl-ESO-HORN et $\mathtt{GRID}_3$ que l'on ajoute à notre schéma (voir Figure 3.21).

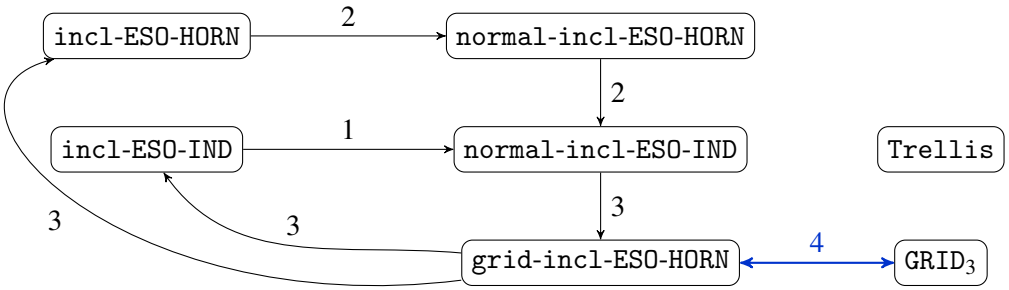


FIGURE 3.21 – Égalité de grid-incl-ESO-HORN et $\mathtt{GRID}_3$

Égalité de $\mathtt{GRID}_3$ et Trellis

L'égalité de $\mathtt{GRID}_3$ et Trellis est triviale. En effet, la bijection entre les deux objets que sont le circuit-grille de $\mathtt{GRID}_3$ et l'automate treillis est immédiate. Pour un mot d'entrée $w = w_1 \dots w_n$, chaque site (x,y) de $\mathtt{GRID}_3$ avec $x \leq y$ (les sites (x,y) avec $y < x$ sont inactifs) est en bijection avec la cellule de l'automate treillis correspondant au facteur $w_x \dots w_y$ de w. Chaque cellule (x,y) de $\mathtt{GRID}_3$ avec $x \leq y$ est aussi en bijection avec la cellule $(c,t) = (y, y-x+1)$ du diagramme espace-temps d'un automate cellulaire avec entrée parallèle et voisinage unidirectionnel travaillant en temps-réel. Ces bijections, illustrées par la Figure 3.22, complètent le schéma de preuve donné par la figure 3.23. Ce qui conclut la preuve du théorème 3.0.3.

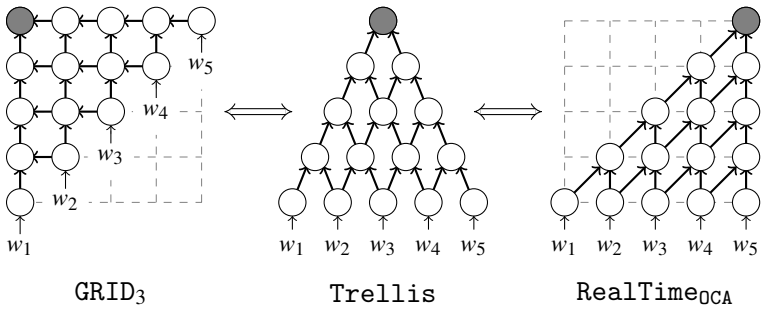


FIGURE 3.22 – Bijections entre $\mathtt{GRID}_3$, Trellis et $\mathtt{RealTime}_{\mathtt{OCA}}$

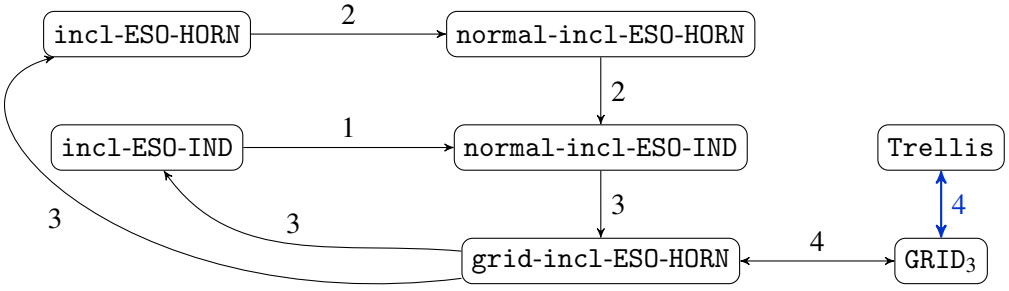
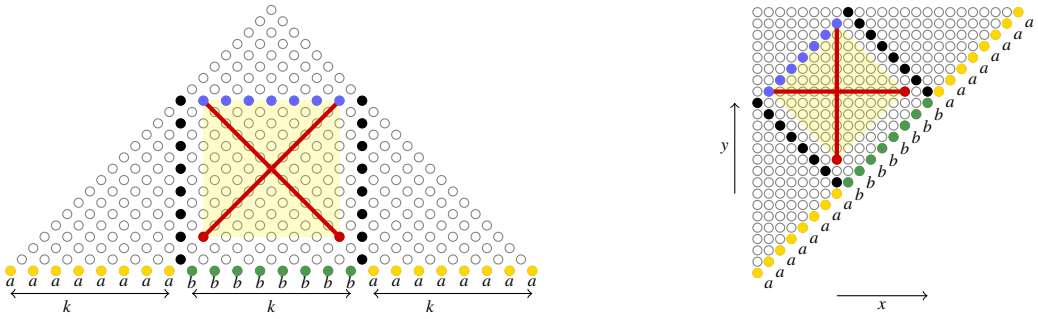


FIGURE 3.23 – Égalité de $\mathtt{GRID}_3$ et Trellis

4 Un problème de référence : la synchronisation



(a) La version treillis

(b) La version $\texttt{GRID}_3$

FIGURE 4.1 – Les versions treillis et $\texttt{GRID}_3$ de l'automate de Čulik reconnaissant le langage $\{a^i b^k a^{k-i} \mid 0 < i < k \text{ et } k \geq 3\}$

La synchronisation est le problème de référence en programmation parallèle. Lors d'une exécution de programme sur plusieurs processus, on veut être capable de faire se rejoindre ces processus à un instant donné : que ce soit dans le but d'établir un consensus ou de démarrer en même temps une nouvelle série d'actions. Sur un automate cellulaire la synchronisation se définit comme suit : toutes les cellules de l'automate atteignent en même temps un nouvel état encore jamais atteint dans le calcul. Le problème de synchronisation sur CA aussi appelé *Firing Squad Synchronization Problem* (ou FSSP) est un sujet largement traité dans la littérature[7, 25, 48] où l'objectif recherché est le plus souvent la minimisation du nombre d'états. On propose dans cette section une formule de `incl-ESO-IND` qui définit le langage $\{a^i b^k a^{k-i} \mid 0 < i < k\}$. Définir ce langage en logique est particulièrement intéressant par son lien avec le FSSP, explicité dans la première sous section de cette partie, on décrira ensuite l'algorithme utilisé sur la grille pour aboutir à une synchronisation avant de pouvoir en déduire une formule définissant le langage $\{a^i b^k a^{k-i} \mid 0 < i < k\}$.

4.1 Lien entre la synchronisation et le langage Culik

En 1989 dans son article *Variations of the firing squad problem and applications* [7], Karel Čulik exhiba un automate treillis reconnaissant le langage $\{a^i b^k a^{k-i} \mid 0 < i < k\}$ que l'on notera `Culik`. Il remarqua que dans le diagramme espace-temps d'un automate treillis travaillant sur un mot $a^k b^k a^k$, les sites du diagramme correspondant aux mots $a^i b^k a^{k-i}$ avec $0 < i < k$ étaient les sites résultant d'une synchronisation (sites représentés par ● sur la Figure 4.1a). La Figure 4.1a montre l'automate treillis tel qu'exhibé par Čulik. L'idée de la reconnaissance est d'utiliser un firing squad à deux généraux établis sur les sites correspondant aux facteurs ab^3 et b^3a (représentés par ● sur la Figure 4.1a). Ainsi, on supposera que k est toujours supérieur ou égal à 3, le cas où $k = 2$ sera traité séparément. La Figure 4.1b présente l'équivalent $\texttt{GRID}_3$ de cet automate sur une grille $3k \times 3k$.

Convention 4.1 Ici comme dans toute la suite, on confond tout site (x, y) de la grille $\texttt{GRID}_3$ tel que $x \leq y$ avec le facteur $w_x \dots w_y$ du mot $w_1 \dots w_{3k} = a^k b^k a^k$.

4.2 L'algorithme de synchronisation

L'idée principale de la reconnaissance du langage `Culik` est la construction récursive d'un arbre dont les nœuds sont des facteurs du mot $a^k b^k a^k$ et la racine est le facteur $ab^k a$. Plus précisément, ce processus récursif est basé sur la notion de *potentiel*.

Définition 4.2.1 — Potentiel d'un facteur. Le *potentiel* d'un facteur $a^i b^k a^j$ tel que $i, j > 0$ et $i + j < k$ est égal à la différence entre son nombre k de b et son nombre $i + j$ de a : $p(a^i b^k a^j) = k - i - j$. Par définition, pour $w \in a^+ bb^+ a^+$ on a : $p(w) = 0 \iff w \in \texttt{Culik}$.

Le but de l'algorithme va être de caractériser les sites des facteurs de potentiel 1 comme feuilles d'un arbre. Cet algorithme consiste à ajouter récursivement à la droite ou à la gauche d'un facteur, un bloc de a de taille égale à la moitié de son potentiel. Chaque facteur correspondant à une étape de ce processus récursif est appelé un *facteur marqué*.

De par la nature discrète des objets utilisés, la récurrence utilise une méthode de division par 2, elle aussi discrète.

Le schéma de récurrence définissant les *facteurs marqués* est le suivant :

— Le facteur $ab^k a$ est un *facteur marqué* ;

— si $a^i b^k a^j$ est un *facteur marqué* avec $i, j \geq 0$ et $k - (i + j) \geq 2$ alors $a^l a^i b^k a^j$ et $a^i b^k a^j a^r$ avec $l = \left\lfloor \frac{k-(i+j)}{2} \right\rfloor$ et $r = \left\lceil \frac{k-(i+j)}{2} \right\rceil$ sont des *facteurs marqués*.

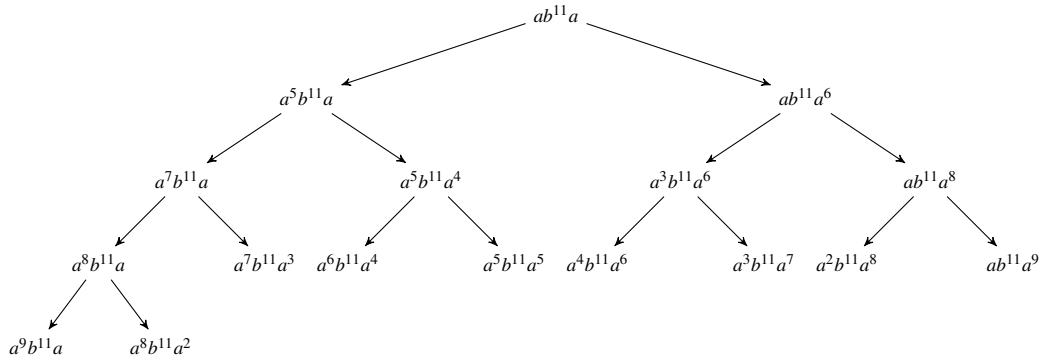
Un exemple d'*arbre de facteurs marqués* obtenu par cette récurrence est donné par la Figure 4.2a : le fils gauche (resp. droit) d'un facteur marqué est le facteur marqué obtenu lorsque l'opération plancher (resp. plafond) est appliquée après la division par 2 de son potentiel. L'*arbre de potentiel* est une vision synthétique de l'arbre des facteurs marqués qui ne prend en compte que le potentiel de chacun des facteurs marqués. Sur cet arbre, la hauteur d'un nœud est donnée par son potentiel. Ainsi, la distance entre un nœud de potentiel p et son fils gauche (resp. droit) est $\left\lfloor \frac{p}{2} \right\rfloor$ (resp. $\left\lceil \frac{p}{2} \right\rceil$) puisque le fils gauche (resp. droit) d'un nœud de potentiel p est $\left\lceil \frac{p}{2} \right\rceil$ (resp. $\left\lfloor \frac{p}{2} \right\rfloor$) comme indiqué sur la Figure 4.2b. La Figure 4.2c, quant à elle, montre le plongement de l'arbre de potentiel dans la grille.

Notation 4.1 Par abus de langage, on appelle l'*arbre de potentiel*, noté T, à la fois l'arbre des facteurs marqués, l'arbre de potentiel correspondant, et son plongement dans la grille.

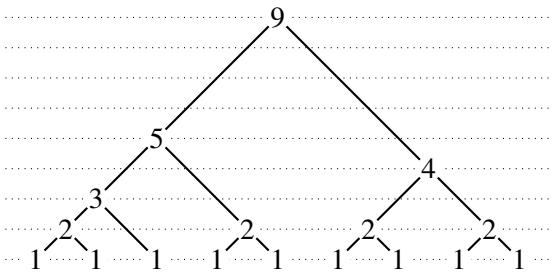
Fait 4.1 Les facteurs de la forme $a^i b^k a^{k-i-1}$, avec $1 \leq i \leq k-2$, correspondent aux $k-2$ feuilles de l'arbre de potentiel : ce sont les nœuds de T de potentiel 1. Les mots de `Culik` de longueur $2k$, donc de la forme $a^i b^k a^{k-i}$, pour $0 < i < k$, correspondent aux sites de potentiel 0 de la grille : les sites (x, y) tels que $(x+1, y)$ ou $(x, y-1)$ sont des feuilles de l'arbre T (voir Figure 4.2).

4.3 Construction de l'arbre de potentiel sur la grille

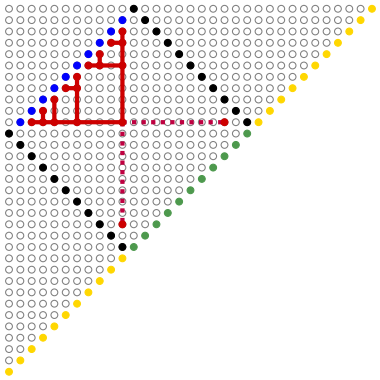
On présente dans cette section, le processus amenant à la construction de l'arbre de potentiel T dans $\texttt{GRID}_3$. On appelle *branche gauche* (resp. *branche droite*) d'un nœud la branche constituée de ce nœud et des fils gauches (resp. droits) successifs jusqu'à arriver à un nœud sans fils gauche (resp. sans fils droit). La construction de l'arbre se fait de manière récursive sur les branches de



(a) L'arbre des facteurs marqués



(b) L'arbre de potentiel T



(c) Plongement de l'arbre T dans $\texttt{GRID}_3$

FIGURE 4.2 – Exemple de récurrence avec $k = 11$

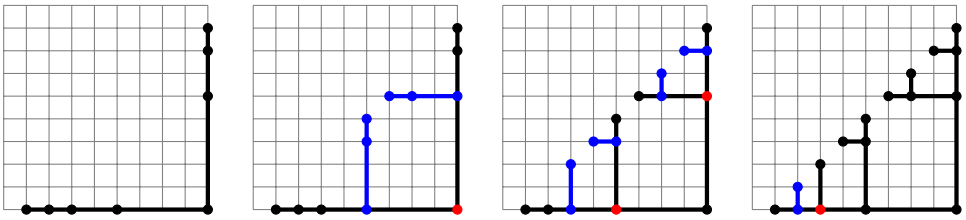


FIGURE 4.3 – Construction récursive de l'arbre de potentiel

celui-ci. Ce processus récursif commence par la construction des branches gauche et droite de la racine (branches la plus à gauche et la plus à droite de T). La construction des branches internes est ensuite réalisée de façon récursive en construisant depuis un nœud (que l'on nomme *nœud de départ*) la branche gauche de son fils droit ainsi que la branche droite de son fils gauche. Un exemple de ce processus récursif est donné Figure 4.3.

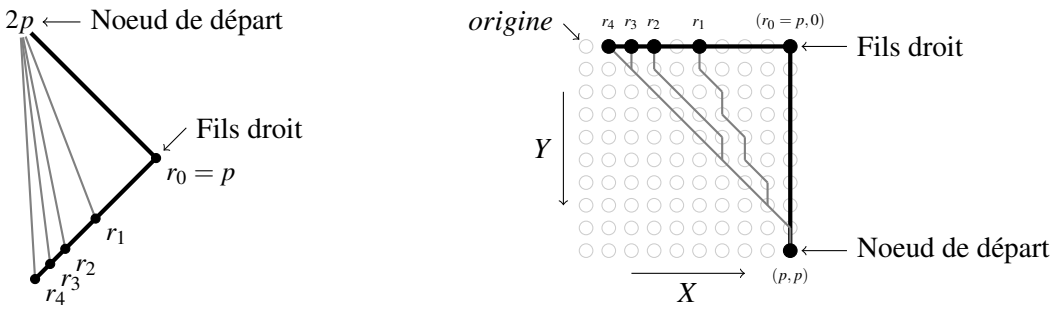
On notera $r_0, r_1, r_2, \ldots$ les potentiels des nœuds successifs de la branche gauche du fils droit et $l_0, l_1, l_2, \ldots$ ceux des nœuds de la branche droite du fils gauche. Ainsi par définition de l'arbre de potentiel, on a : $r_{i+1} = \lceil \frac{r_i}{2} \rceil$ et $l_{i+1} = \left\lfloor \frac{l_i}{2} \right\rfloor$.

En premier lieu, on présentera la construction des branches dans le cas d'un nœud de départ pair (i.e., de potentiel pair) puis, en second lieu, dans le cas d'un nœud de départ impair qui nécessite quelques ajustements pour gérer la dissymétrie entre les deux branches. Une fois ces cas généraux traités, on abordera la construction des deux branches la plus à gauche et la plus à droite de l'arbre avant de finir par la caractérisation des feuilles de l'arbre de potentiel, i.e., les facteurs marqués de potentiel 1.

4.3.1 Construction des branches gauche et droite d'un nœud pair

Construction de la branche gauche du fils droit

On décrit ici la construction de la branche gauche du fils droit d'un nœud de potentiel pair $2p$. Cette construction est réalisée à partir du nœud de départ (celui de potentiel $2p$) et de son fils droit (voir Figure 4.4a). Elle utilise une famille de droites initialisées à partir du nœud de départ qui intersectent la branche gauche du fils droit aux endroits voulus. La construction sur la grille est présentée sur la Figure 4.4b. Afin de faciliter la présentation, on utilisera ici le système de coordonnées (X,Y) suivant : le nœud de départ se trouve sur le site (p,p) et son fils droit sur le site $(p,0)$. Ainsi, les sites que l'on souhaite caractériser sont de la forme $(r_i,0)$.



(a) Sur l'arbre de potentiel

(b) Sur la grille

FIGURE 4.4 – Construction de la branche gauche du fils droit d'un nœud de départ pair

La i-ème droite, utilisée pour caractériser le site $(r_i, 0)$, part du site (p, p) et suit la trajectoire suivante : un mouvement vertical $(0, -1)$ suivi d'alternances entre $2^i - 1$ mouvements diagonaux $(-1, -1)$ et un mouvement vertical $(0, -1)$.
Notons $(X_i, 0)$ l'intersection entre cette i-ème droite et la branche gauche et vérifions que l'on a bien $X_i = r_i$. Soit α le nombre de fois que la phase composée de $2^i - 1$ mouvements diagonaux et d'un mouvement vertical est effectuée, et $\beta < 2^i$ le nombre de mouvements diagonaux restant à effectuer. On a l'égalité suivante :
$(X_i, 0) = (p, p) + (0, -1) + \alpha(-2^i + 1, -2^i + 1) + \alpha(0, -1) + \beta(-1, -1)$
De cette égalité, on peut déduire que α est le quotient de la division euclidienne de $p - 1$ par 2^i et β le reste : $\alpha = (p-1)//2^i$ et $\beta = (p-1)\%2^i$. On a donc l'expression suivante pour X_i :

$$\begin{aligned} X_i &= p - 2^i\alpha + \alpha - \beta = 1 + 2^i\alpha + \beta - 2^i\alpha + \alpha - \beta = 1 + \alpha \\ &= 1 + (p-1)//2^i \end{aligned}$$

D'abord, en utilisant l'identité $\lceil \frac{a}{b} \rceil = 1 + \lfloor \frac{(a-1)}{b} \rfloor$ on a $X_i = \lceil \frac{p}{2^i} \rceil$. Ensuite, rappelons que les potentiels r_0, r_1,... des nœuds de la branche gauche suivent la récurrence : $r_0 = p$ et $r_{i+1} = \lceil \frac{r_i}{2} \rceil$. Comme $\lceil \lceil \frac{a}{b} \rceil / c \rceil = \lceil \frac{a}{bc} \rceil$, on obtient alors l'égalité $r_i = \lceil \frac{p}{2^i} \rceil = X_i$.

Après avoir prouvé l'exactitude de la construction des nœuds de la branche gauche à partir d'une famille de droites, nous allons détailler le processus local utilisé pour construire de telles droites. La construction d'une famille de lignes de pentes 2^i est bien connue [32]. Elle utilise deux types de signaux : les signaux horloges et les signaux filtres. La Figure 4.5 décrit la construction de ces signaux ainsi que leurs interactions. Les signaux filtres y jouent le rôle de la famille de droites utilisée pour construire les nouveaux nœuds.

Les signaux horloges sont émis depuis le côté droite du carré sur chaque site $(p, p - y)$ avec y pair et strictement positif et se dirigent vers la gauche. Ils sont symbolisés par ● sur la Figure 4.5. Les signaux filtres alternent entre deux phases : *cross* et *delete*. Ils sont symbolisés par ● (phase *cross*) et ● (phase *delete*) sur la Figure 4.5. Un signal diagonal est aussi émis depuis le site $(p, p-1)$ et se dirige suivant le vecteur $(-1, -1)$. À chaque fois qu'un signal horloge rencontre ce signal diagonal, un nouveau signal filtre en phase *cross* est initialisé. Ce signal diagonal est aussi utilisé pour caractériser la feuille de la branche lorsqu'il intersecte celle-ci.

Les signaux filtres et diagonaux interagissent entre eux afin que les signaux filtres aient la pente désirée. Par défaut, un signal filtre va diagonalement suivant le vecteur $(-1, -1)$. Il ne dévie de cette trajectoire que lorsqu'il rencontre un signal horloge, il effectue alors un mouvement vertical suivant le vecteur $(0, -1)$ avant de reprendre sa trajectoire diagonale. De plus, à chacune de ces intersections entre un signal filtre et un signal horloge, le signal filtre change de phase passant de *delete* à *cross* ou inversement, le signal horloge, quant à lui, passe à travers cette intersection si le signal filtre est en phase *cross* ou s'arrête quand celui-ci est en phase *delete*.

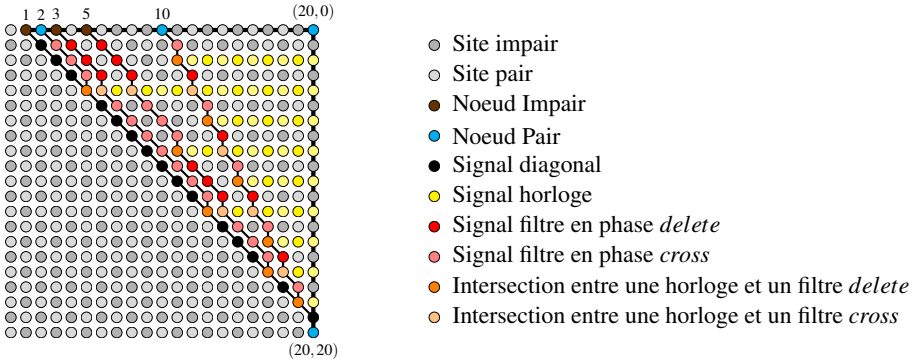
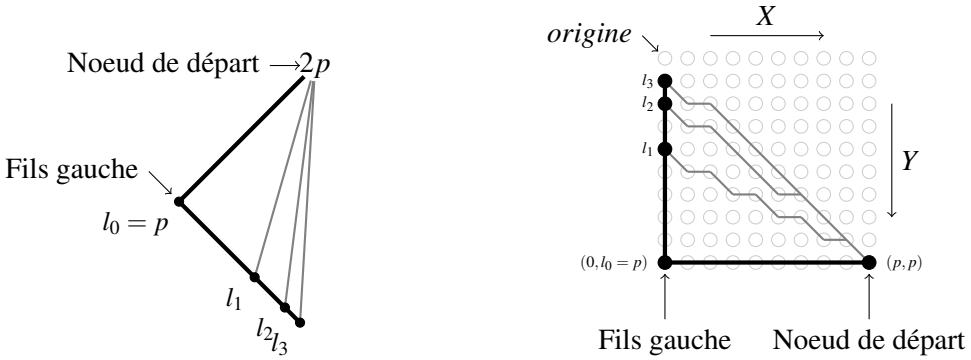


FIGURE 4.5 – Construction de la branche gauche du fils droit d'un nœud de départ de potentiel $2p = 40$

Construction de la branche droite du fils gauche

Comme montré sur la Figure 4.6, la construction de la branche droite du fils gauche d'un nœud de potentiel pair $2p$ est similaire au cas de la branche gauche du fils droit. Cette construction est réalisée à partir du nœud de départ (celui de potentiel $2p$) et de son fils gauche. Comme dans la section précédente, on utilisera le système de coordonnées (X,Y) tel que le nœud de départ se trouve sur le site (p,p) et son fils gauche sur le site $(0,p)$. Ainsi, les sites que l'on souhaite caractériser sont de la forme $(0,l_i)$ avec $l_0 = p$ et $l_{i+1} = \left\lfloor \frac{l_i}{2} \right\rfloor$. Comme $\left\lfloor \left\lfloor \frac{a}{b} \right\rfloor / c \right\rfloor = \left\lfloor \frac{a}{bc} \right\rfloor$, on a $l_i = \left\lfloor \frac{p}{2^i} \right\rfloor$ pour tout i.



(a) Sur l'arbre de potentiel

(b) Sur la grille

FIGURE 4.6 – Construction de la branche droite du fils gauche d'un nœud de potentiel pair

La i-ème droite utilisée pour caractériser le site $(0,l_i)$ démarre sur le site (p,p) et alterne entre $2^i - 1$ mouvements diagonaux $(-1,-1)$ et un mouvement horizontal $(-1,0)$. Soit $(0,Y_i)$ l'intersection entre cette i-ème droite et la branche droite partant du fils gauche. Vérifions que l'on a bien $Y_i = l_i$. Soient α le nombre de phases (composées de $2^i - 1$ mouvement diagonaux et d'un mouvement horizontal) réalisées et $\beta < 2^i$ le nombre de mouvement diagonaux restant. On a l'égalité suivante :

$$(0,Y_i) = (p,p) + \alpha(-2^i + 1, -2^i + 1) + \alpha(0,-1) + \beta(-1,-1)$$

De cette égalité, on déduit que α est le quotient et β le reste de la division euclidienne de p par 2^i : $\alpha = p//2^i$ et $\beta = p\%2^i$. Ainsi :

$$\begin{aligned} Y_i &= p - 2^i\alpha + \alpha - \beta = 2^i\alpha + \beta - 2^i\alpha + \alpha - \beta = \alpha \\ &= p//2^i = l_i \end{aligned}$$

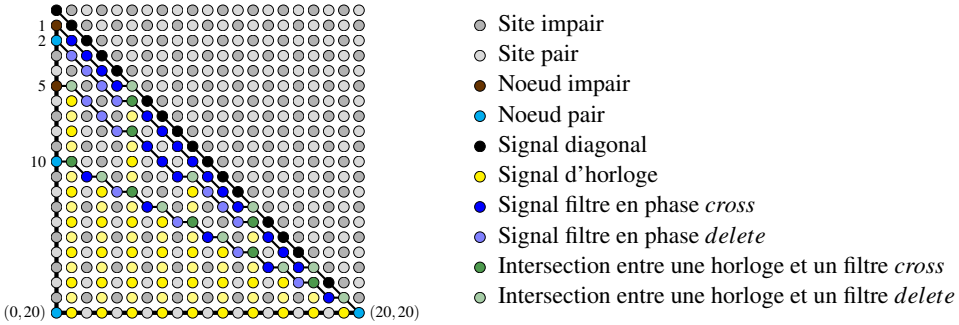
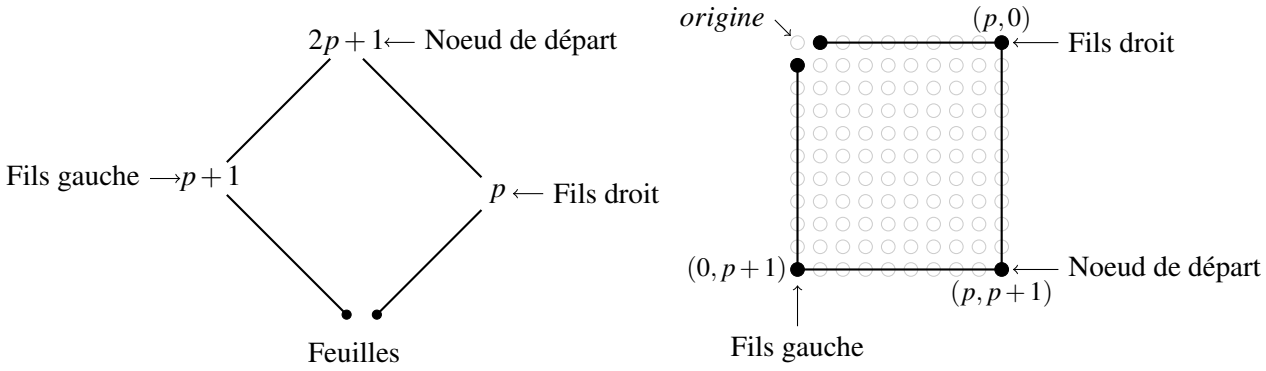


FIGURE 4.7 – Construction de la branche droite partant du fils gauche d'un nœud de départ de potentiel $2p = 40$



(a) Sur l'arbre de potentiel

(b) Sur la grille

FIGURE 4.8 – Construction des branches gauche et droite d'un nœud de potentiel impair

Le processus local amenant à la construction de cette famille de droites, à partir de signaux d'horloge et de signaux filtres, est essentiellement le même que dans le cas précédent.

La Figure 4.7 illustre cette construction. Les signaux horloges sont émis depuis le côté inférieur du carré sur chaque site $(p-y, p)$ où y est impair, et se dirigent vers le haut. Les signaux filtres ont toujours deux phases *cross* et *delete*. Ils interagissent de la même façon que précédemment avec les signaux horloges, la seule différence étant qu'une intersection entre ces deux types de signaux résulte en un décalage horizontal $(-1,0)$ plutôt que vertical pour le signal filtre. Enfin, un signal diagonal émis depuis le site (p, p) et se dirigeant suivant le vecteur $(-1,-1)$ est utilisé pour initialiser les signaux filtres.

4.3.2 Construction de la branche gauche du fils droit et de la branche droite du fils gauche d'un nœud impair

Quand le potentiel du nœud de départ est impair (égal à $2p+1$), l'espace délimité par son fils droit et son fils gauche n'est plus un carré mais un rectangle de largeur p et de hauteur $p+1$ comme indiqué sur la Figure 4.8.

Construction de la branche à gauche du fils droit

Avec $r_0, r_1, \ldots$ les potentiels des nœuds successifs de la branche gauche partant du fils droit d'un nœud de potentiel impair $2p+1$, on a : $r_0 = p$ et $r_{i+1} = \left\lceil \frac{r_i}{2} \right\rceil$. Afin de caractériser les sites $(r_i, 0)$ du rectangle, on se ramène au cas du carré de longueur p en effectuant un plus grand mouvement vertical $(0,-2)$ avant de commencer l'alternance entre les $2^i - 1$ mouvements diagonaux $(-1,-1)$

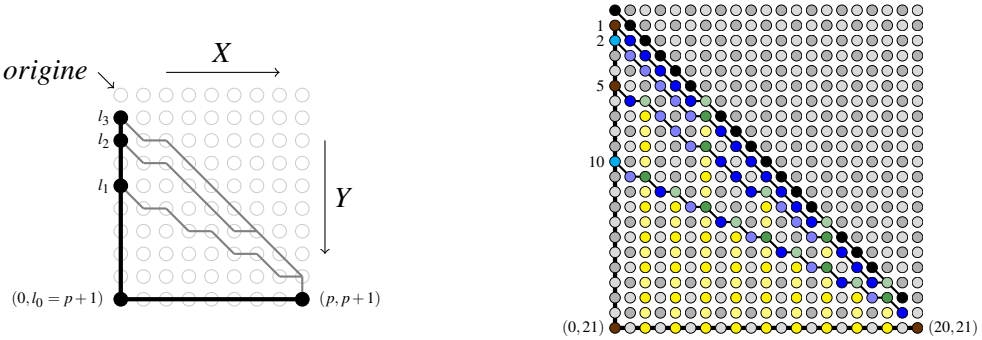


FIGURE 4.10 – Construction de la branche droite vue comme le côté gauche d'un rectangle

et le mouvement vertical $(0,-1)$ (voir Figure 4.9).

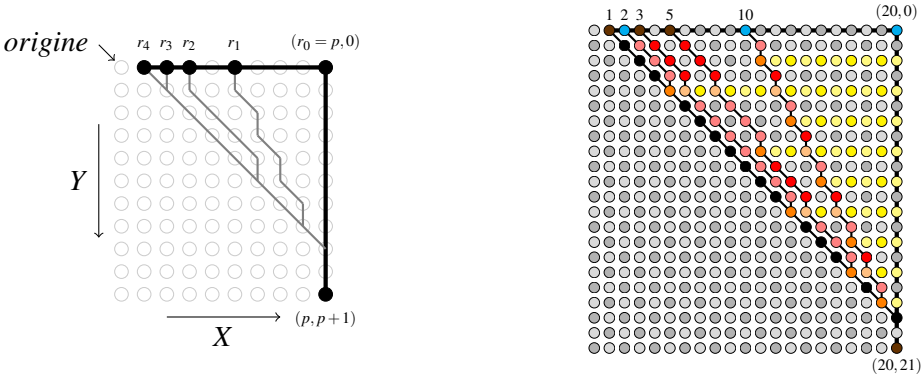


FIGURE 4.9 – Construction de la branche gauche vue comme le côté supérieur d'un rectangle

Construction de la branche droite du fils gauche

Avec $l_0, l_1, \ldots$ les potentiels des nœuds successifs de la branche droite du fils gauche d'un nœud de potentiel impair $2p+1$ on a : $l_0 = p+1$ et $l_{i+1} = \left\lfloor \frac{l_i}{2} \right\rfloor$.
Afin de caractériser les sites $(0, l_i)$ du rectangle, on se ramène au cas du carré en simulant un nœud de départ sur le site $(p+1, p+1)$. La droite caractérisant le site $(0, l_i)$ est émise depuis le site $(p, p+1)$ et suit la trajectoire suivante : un mouvement vertical $(0,-1)$ suivi par $2^i - 2$ mouvements diagonaux $(-1,-1)$, puis une alternance entre un mouvement horizontal $(-1,0)$ et $2^i - 1$ mouvements diagonaux $(-1,-1)$. La Figure 4.10 illustre cette construction dans le rectangle.

4.3.3 Initialisation de la récurrence : construction des branches extérieures de l'arbre.

La construction de la branche la plus à gauche (resp. la plus à droite) de l'arbre démarre du site correspondant au facteur *abbb* appelé *général gauche* (resp. *bbba* appelé *général droit*). De cette façon, la distance entre le site de départ et le site du facteur $ab^k a$ (la racine de l'arbre) est $k-2$ qui est le potentiel de ce facteur (voir Figure 4.11). Nous pouvons ainsi utiliser la même famille de droites que dans le cas d'un nœud de départ pair avec le général gauche (resp. droit) jouant le rôle du nœud de départ et la racine $ab^k a$ jouant celui de son fils droit (resp. gauche).

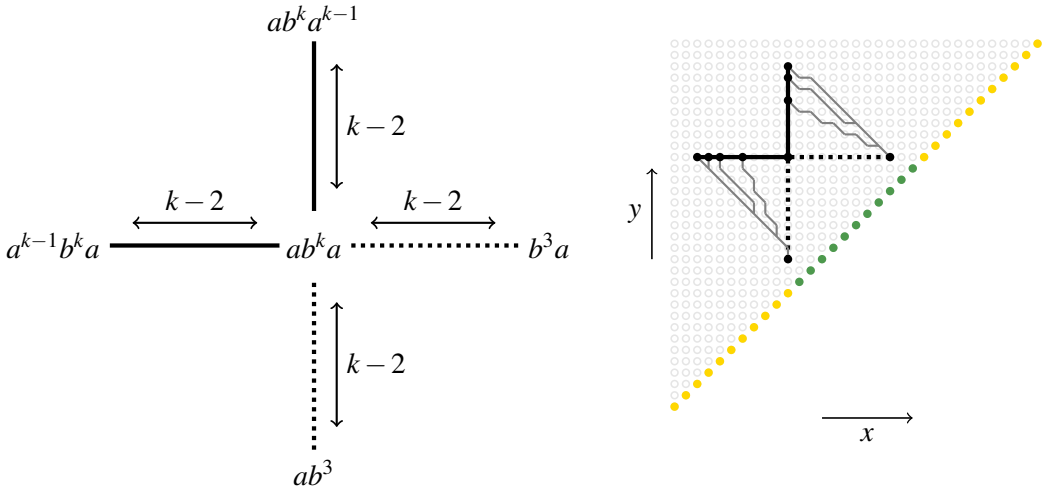


FIGURE 4.11 – Construction des branches la plus à gauche et la plus à droite de l'arbre

4.3.4 Fin de la récurrence : caractérisation des feuilles de l'arbre

Afin de terminer la construction récursive de l'arbre de potentiel, nous avons besoin de caractériser localement ses feuilles. Avec le système de coordonnées usuel de la grille, cette caractérisation est donnée par le lemme suivant.

Lemme 4.3.1 — Caractérisation des feuilles. Si l'arbre de potentiel T contient au moins deux nœuds, alors un nœud se trouvant sur le site (x,y) de la grille est une feuille *ssi* une des conditions suivantes est vérifiée :

(1) Le potentiel du nœud (x,y) est impair et le site $(x+1,y)$ est un nœud de T ;
(2) Le site $(x,y-1)$ est un nœud de T ;
(3) Le potentiel du nœud (x,y) est impair et le site $(x,y-2)$ est un nœud de T.

Démonstration. Soit p le potentiel du nœud (x,y), les potentiels des sites $(x+1,y)$, $(x,y-1)$ et $(x,y-2)$ sont alors respectivement $p+1$, $p+1$, et $p+2$.

Supposons que le nœud (x,y) est une feuille. Donc son potentiel p est égal à 1. Dans le cas où (x,y) est une feuille gauche (i.e., est une feuille et le fils gauche d'un nœud), le potentiel de son nœud parent q vérifie $\lceil \frac{q}{2} \rceil = 1$. On en déduit que $q = 2$ et que le nœud parent se trouve sur le site $(x+1,y)$. Dans le cas où (x,y) est une feuille droite, le potentiel de son nœud parent q vérifie $\lfloor \frac{q}{2} \rfloor = 1$. On en déduit que $q \in \{2,3\}$ et donc que son nœud parent est soit $(x,y-1)$ ou $(x,y-2)$. On a ainsi montré que si (x,y) est une feuille, alors l'une des conditions (1), (2) ou (3) est satisfaite.

Montrons que la réciproque est vraie. Si la condition (1) est vérifiée, alors le potentiel $p+1$ du nœud $(x+1,y)$ est pair. Le potentiel p de son fils gauche (x,y) satisfait donc l'égalité $p = \frac{p+1}{2}$: ce qui implique $p = 1$ et donc que (x,y) est une feuille. Si la condition (2) est vérifiée, (x,y) et $(x,y-1)$ étant deux nœuds de T, alors le nœud (x,y) de potentiel p est fils droit du nœud $(x,y-1)$ de potentiel $p+1$, d'où $p = \lfloor \frac{p+1}{2} \rfloor$: ce qui implique $p = 1$ et donc que (x,y) est une feuille. Si la condition (3) est vérifiée, alors le potentiel $p+2$ du nœud $(x,y-2)$ est impair. Le potentiel du fils droit de $(x,y-2)$ est donc $\lfloor \frac{p+2}{2} \rfloor = \frac{p+1}{2} \geq p$ d'où on tire $p = 1$, ce qui signifie que le nœud (x,y) de potentiel p est une feuille. □

4.4 Une formule d'inclusion inductive définissant le langage `Culik`

La section précédente décrit la caractérisation de tous les facteurs appartenant au langage $\{a^i b^k a^{k-i} \mid 0 < i < k \text{ et } k \geq 3\}$ à l'intérieur d'une grille $3k \times 3k$ avec $a^k b^k a^k$ comme mot d'entrée. Dans cette section nous détaillons les clauses exprimant l'appartenance ou non d'un mot $w \in \{a,b\}^+$ à ce langage. Dans ce but, on considère une grille de $\texttt{GRID}_3$, de taille $n \times n$ avec le mot d'entrée $w = w_1 \dots w_n \in \{a,b\}^+$ écrit sur la diagonal $y = x$. Afin de présenter les formules de façon plus légère, on simplifie la présentation des clauses d'inclusion en adoptant la convention suivante :

$R(x+a, y-b)$ est écrit à la place de la conjonction $x + a \leq y - b \wedge R(x+a, y-b)$. Dans le même esprit, on écrira $x = 1 \wedge y = n$ à la place de $x \leq y \wedge x = 1 \wedge y = n$.

Structure de la définition logique de `Culik`

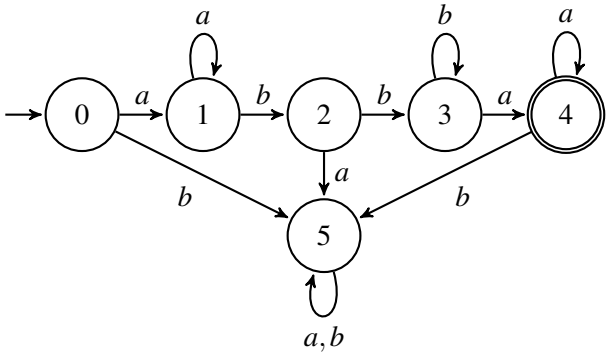
La définition du langage `Culik` à l'intérieur du langage $a^+bb^+a^+$ utilise 28 prédicats binaires :

— 8 prédicats imitant le calcul d'un automate fini reconnaissant le langage $a^+bb^+a^+$: T_a, T_b, R_0, R_1, R_2, R_3, R_4 et R_5.
— 8 prédicats globaux initiaux : les prédicats de parité `Even` et `Odd` ; les prédicats de bordure (entre a et bbb ou entre bbb et a) `LeftBorder` et `RightBorder` seulement vrais sur le dernier (resp. le premier) a de la première (resp. seconde) série de a du mot d'entrée appartenant à $a^+b^+a^+$; les prédicats `LeftGeneral` et `RightGeneral` vrais pour les nœuds général gauche et droit, `StartLeftBranch` et `StartRightBranch` vrais pour les nœuds de la branche la plus à gauche (resp. la plus à droite) (ces prédicats n'ont de sens que si le nombre de b est supérieur ou égal à 3, i.e. pour un mot d'entrée appartenant à $a^+bb^+ba^+$) ;
— 4 prédicats globaux définis récursivement : les prédicats `Node` et `Leaf` vrais pour les nœuds et les feuilles et les prédicats `LeftBranch` et `RightBranch` vrais pour les nœuds situés sur une branche gauche (resp. droite).
— 8 prédicats locaux étiquetés par `Up` ou `Low` : les prédicats diagonaux `UpDiag` et `LowDiag`, les prédicats horloge `UpClock` et `LowClock`, les prédicats filtre `UpCross`, `UpDelete`, `LowCross` et `LowDelete`.

Convention : On considère les deux généraux correspondant aux facteurs ab^3 (général gauche) et b^3a (général droit) comme des nœuds de l'arbre T. L'arbre n'est alors plus un arbre enraciné usuel, i.e. avec une seule racine. On a maintenant trois nœuds initiaux : ab^3, b^3a et ab^ka avec $k \geq 3$. Par commodité dans la construction logique, on admet que l'arbre possède deux racines ab^3 et b^3a et que le nœud ab^ka est à la fois le fils droit du nœud ab^3 et le fils gauche du nœud b^3a.

Clauses définissant le langage $a^+bb^+a^+$:

Les clauses utilisées pour définir le langage $a^+bb^+a^+$ imitent le calcul d'un automate fini $\mathscr{A}$ reconnaissant cette expression régulière.



$\Sigma = \{a,b\}$ l'alphabet d'entrée
$\mathtt{S} = \{0,1,2,3,4,5\}$ l'ensemble des états
0 l'état initial
$F = \{4\}$ l'ensemble des états acceptants
$\delta : \mathtt{S} \times \Sigma \to \mathtt{S}$ la fonction de transition donnée par son diagramme

FIGURE 4.12 – L'automate $\mathscr{A} = (\Sigma, \mathtt{S}, 0, F, \delta)$ reconnaissant le langage $a^+bb^+a^+$

Les prédicats transportant l'information de l'entrée à travers la grille sont T_s, pour $s \in \Sigma$, qui sont définis par les clauses :
(1) $x = y \wedge Q_s(x) \to T_s(x,y)$; $T_s(x+1,y) \to T_s(x,y)$, pour $s \in \Sigma$.
Les prédicats simulant le calcul de l'automate $\mathscr{A} = (\Sigma, \mathtt{S}, 0, F, \delta)$ sont R_q, pour $q \in \mathtt{S}$, qui sont définis par les clauses :
(2) $x = 1 \wedge x = y \wedge Q_s(x) \to R_{\delta(0,s)}(x,y)$, pour $s \in \Sigma$;
(3) $R_q(x,y-1) \wedge T_s(x+1,y) \to R_{\delta(q,s)}(x,y)$, pour $(q,s) \in \mathtt{S} \times \Sigma$;
(4) $x = 1 \wedge y = n \wedge R_q(x,y) \to \bot$, pour $q \in \mathtt{S} \setminus F$.
La conjonction de ces clauses assure que le mot d'entrée appartient bien au langage $a^+b^+ba^+$.

Remarque 4.1 De façon générale, tout langage rationnel $L(\mathscr{A})$ reconnu par un automate fini $\mathscr{A} = (\Sigma, \mathtt{S}, 0, F, \delta)$ est défini par la formule $\exists (R_q)_{q\in\mathtt{S}} \forall x \forall y (\psi_0^{\mathscr{A}} \wedge \psi_{neg}^{\mathscr{A}})$ ou $\exists (R_q)_{q\in\mathtt{S}} \forall x \forall y (E(\psi_0^{\mathscr{A}}) \wedge \psi_{neg}^{\mathscr{A}})$ où $\psi_0^{\mathscr{A}}$ est la conjonction des clauses (1-3), $(E(\psi_0^{\mathscr{A}})$ est la conjonction d'équivalences issue de $\psi_0^{\mathscr{A}}$ (en accord avec la Convention 2.1 page 42) et $\psi_{neg}^{\mathscr{A}}$ la conjonction des clauses (4).

4.4.1 Clauses définissant l'initialisation et la fin de la récurrence

Les prédicats de parité `Even` et `Odd` :

Lors de la construction inductive de l'arbre de potentiel, nous avons besoin de connaître la parité du potentiel de chaque nœud. Le potentiel d'un mot du langage $a^+bb^+a^+$ étant la différence entre son nombre de b et son nombre de a, la parité de son potentiel est la même que la parité de sa longueur. De cette façon, la parité du site (x,y) est la parité de la différence $y-x+1$ et en particulier les sites diagonaux $(y=x)$ sont de potentiel impair. Les clauses suivantes définissent inductivement la parité de chaque site (x,y) avec $x \le y$ en partant de la diagonale :
— $x = y \to \mathtt{Odd}(x,y)$; $x < y \wedge \mathtt{Odd}(x,y-1) \to \mathtt{Even}(x,y)$;
$x < y \wedge \mathtt{Even}(x,y-1) \to \mathtt{Odd}(x,y)$.

Les prédicats de bordure `LeftBorder`, `RightBorder` et les deux généraux `LeftGeneral` et `RightGeneral` :

La bordure gauche (resp. droite) est définie sur la diagonale $x = y$ comme le dernier (resp. premier) a de la première (resp. seconde) série de a d'un mot appartenant au langage $a^+bb^+ba^+$. À partir de la caractérisation de la bordure gauche (resp. droite) on caractérise le général gauche (resp. droite) correspondant au facteur ab^3 (resp. b^3a) :
— $x = y \wedge Q_a(x) \wedge Q_b(x+1) \wedge Q_b(x+2) \wedge Q_b(x+3) \to \mathtt{LeftBorder}(x,y)$;
— $x = y \wedge Q_b(x-3) \wedge Q_b(x-2) \wedge Q_b(x-1) \wedge Q_a(x) \to \mathtt{RightBorder}(x,y)$;
— $\mathtt{LeftBorder}(x,y-3) \to \mathtt{LeftGeneral}(x,y)$; $\mathtt{RightBorder}(x+3,y) \to \mathtt{RightGeneral}(x,y)$.

Les prédicats branches externes `StartLeftBranch` et `StartRightBranch` :

La demi-droite définissant la branche droite (resp. gauche) externe de l'arbre est émise verticalement (resp. horizontalement) depuis le général gauche (resp. droite) :

— $\texttt{LeftGeneral}(x,y) \to \texttt{StartRightBranch}(x,y)$;
$\texttt{StartRightBranch}(x,y-1) \to \texttt{StartRightBranch}(x,y)$;

— $\texttt{RightGeneral}(x,y) \to \texttt{StartLeftBranch}(x,y)$;
$\texttt{StartLeftBranch}(x+1,y) \to \texttt{StartLeftBranch}(x,y)$.

Les trois premiers nœuds (avec le prédicat `Node`) :

Les généraux gauche et droit sont considérés comme des nœuds de l'arbre :

— $\texttt{LeftGeneral}(x,y) \to \texttt{Node}(x,y)$; $\texttt{RightGeneral}(x,y) \to \texttt{Node}(x,y)$

L'intersection entre les deux branches externes définit elle aussi un nœud de l'arbre (le nœud $ab^k a$) :

— $\texttt{StartLeftBranch}(x,y) \wedge \texttt{StartRightBranch}(x,y) \to \texttt{Node}(x,y)$

Remarque : Si les sites correspondant aux facteurs ab^3 et b^3a n'existent pas, les prédicats `LeftGeneral` et `RightGeneral` ne sont jamais vérifiés. C'est pourquoi les trois nœuds initiaux sont créés si et seulement si ab^3 et b^3a sont des facteurs du mot d'entrée.

Les prédicats branches `RightBranch` et `LeftBranch` :

De chaque nœud partent une demi-droite horizontale orientée vers la gauche symbolisant sa branche gauche et une demi-droite verticale orientée vers le haut symbolisant sa branche droite :

— $\texttt{Node}(x+1,y) \to \texttt{LeftBranch}(x,y)$; $\texttt{LeftBranch}(x+1,y) \to \texttt{LeftBranch}(x,y)$;

— $\texttt{Node}(x,y-1) \to \texttt{RightBranch}(x,y)$; $\texttt{RightBranch}(x,y-1) \to \texttt{RightBranch}(x,y)$.

Le prédicat feuille `Leaf` :

D'après le lemme 4.3.1, un nœud (x,y) est une feuille de T *ssi* une de ces conditions est vérifiée :

Le potentiel du nœud (x,y) est impair et le site $(x+1,y)$ est un nœud de T :

— $\texttt{Node}(x,y) \wedge \texttt{Odd}(x,y) \wedge \texttt{Node}(x+1,y) \to \texttt{Leaf}(x,y)$;

Le site $(x,y-1)$ est un nœud de T :

— $\texttt{Node}(x,y) \wedge \texttt{Node}(x,y-1) \to \texttt{Leaf}(x,y)$;

Le potentiel du nœud (x,y) est impair et le site $(x,y-2)$ est un nœud de T :

— $\texttt{Node}(x,y) \wedge \texttt{Odd}(x,y) \wedge \texttt{Node}(x,y-2) \to \texttt{Leaf}(x,y)$.

Remarque : Si $w \in a^+b^3a^+$ alors la première (ou seconde) clause ci-dessus définit la seule feuille de l'arbre.

4.4.2 Définition des nœuds de l'arbre de potentiel

Cas pair, les nœuds de la branche gauche (triangles supérieurs)

Le prédicat de diagonale supérieure `UpDiag` :

Quand le site (x,y) est un nœud pair, une diagonale supérieure est initialisée sur le site $(x,y+1)$:

— $\texttt{Node}(x,y-1) \wedge \texttt{Even}(x,y-1) \to \texttt{UpDiag}(x,y)$

Une fois initialisée, la diagonale suit une trajectoire suivant le vecteur $(-1,1)$ jusqu'à croiser une branche gauche, assurant qu'aucun nœud n'est créé en dehors de l'espace de travail du nœud de départ :

— $\texttt{UpDiag}(x+1,y-1) \wedge \neg\texttt{LeftBranch}(x+1,y-1) \to \texttt{UpDiag}(x,y)$.

L'horloge supérieure `UpClock` :

Le premier tic de l'horloge est émis sur la branche droite du nœud de départ pair, deux sites au dessus de ce nœud de départ. En d'autres termes, un site (x,y) sera le premier tic de l'horloge supérieure si et seulement si le site $(x,y-2)$ est un nœud pair :

— $\texttt{Node}(x,y-2) \wedge \texttt{Even}(x,y-2) \rightarrow \texttt{UpClock}(x,y)$

Une fois le premier tic émis, un nouveau tic est initialisé tous les deux sites de la branche droite jusqu'à atteindre le fils droit du nœud de départ :

— $\texttt{RightBranch}(x,y) \wedge \texttt{UpClock}(x,y-2) \wedge \neg\texttt{Node}(x,y-1) \wedge \neg\texttt{Node}(x,y-2) \rightarrow \texttt{UpClock}(x,y)$

Une fois émis, un signal d'horloge supérieur se dirige vers la gauche jusqu'à croiser un signal filtre en phase *delete* ou la diagonale supérieure :

— $\texttt{UpClock}(x+1,y) \wedge \neg\texttt{UpDelete}(x+1,y) \wedge \neg\texttt{UpDiag}(x+1,y) \rightarrow \texttt{UpClock}(x,y)$

Les signaux filtres supérieurs UpCross et UpDelete :

Les deux phases *cross* et *delete* d'un signal filtre supérieur sont symbolisées par les prédicats UpCross and UpDelete. Un signal filtre supérieur est initialisé en phase *cross* sur le site juste au dessus de chaque intersection entre une horloge supérieure et la diagonale supérieure, sauf si cette intersection a lieu sur la branche gauche dont on cherche à définir les nœuds :

— $\texttt{UpDiag}(x,y-1) \wedge \texttt{UpClock}(x,y-1) \wedge \neg\texttt{LeftBranch}(x,y-1) \rightarrow \texttt{UpCross}(x,y)$

Par défaut, les signaux filtres se déplacent diagonalement suivant le vecteur $(-1,-1)$ sans changer de phase jusqu'à croiser un signal d'horloge supérieur ou la branche gauche :

— $\texttt{UpCross}(x+1,y-1) \wedge \neg\texttt{UpClock}(x+1,y-1) \wedge \neg\texttt{LeftBranch}(x+1,y-1) \rightarrow \texttt{UpCross}(x,y)$;
$\texttt{UpDelete}(x+1,y-1) \wedge \neg\texttt{UpClock}(x+1,y-1) \wedge \neg\texttt{LeftBranch}(x+1,y-1) \rightarrow \texttt{UpDelete}(x,y)$

Lorsqu'un signal filtre croise un signal d'horloge supérieur sur le site $(x,y-1)$, si cette intersection n'a pas lieu sur la branche gauche, alors le signal filtre effectue un changement de phase avant de reprendre sa trajectoire diagonale depuis le site (x,y) :

— $\texttt{UpCross}(x,y-1) \wedge \texttt{UpClock}(x,y-1) \wedge \neg\texttt{LeftBranch}(x,y-1) \rightarrow \texttt{UpDelete}(x,y)$;
$\texttt{UpDelete}(x,y-1) \wedge \texttt{UpClock}(x,y-1) \wedge \neg\texttt{LeftBranch}(x,y-1) \rightarrow \texttt{UpCross}(x,y)$

Les nœuds de la branche gauche :

Un nouveau nœud est crée sur chaque intersection de la branche gauche avec la diagonale supérieure ou avec un signal filtre (quelque soit sa phase) :

— $\texttt{UpDiag}(x,y) \wedge \texttt{LeftBranch}(x,y) \rightarrow \texttt{Node}(x,y)$;

— $\texttt{UpCross}(x,y) \wedge \texttt{LeftBranch}(x,y) \rightarrow \texttt{Node}(x,y)$;

— $\texttt{UpDelete}(x,y) \wedge \texttt{LeftBranch}(x,y) \rightarrow \texttt{Node}(x,y)$.

Remarque : Si ab^3 et b^3a ne sont pas des facteurs du mot d'entrée, tous les prédicats définis ci dessus ne sont vérifiés sur aucun site (exceptés Odd et Even) et aucun nœud (donc aucune feuille) n'est créé.

Cas pair, les nœuds de la branche droite (triangles inférieurs)

La diagonale inférieure LowDiag :

Quand le site (x,y) est un nœud pair, une diagonale inférieure est initialisée sur le site $(x-1,y+1)$:

— $\texttt{Node}(x+1,y-1) \wedge \texttt{Even}(x+1,y-1) \rightarrow \texttt{LowDiag}(x,y)$.

Comme la diagonale supérieure, la diagonale inférieure suit une trajectoire diagonale suivant le vecteur $(-1,1)$, cette diagonale s'arrête ensuite lorsqu'elle croise la branche droite :

— $\texttt{LowDiag}(x+1,y-1) \wedge \neg\texttt{RightBranch}(x+1,y-1) \rightarrow \texttt{LowDiag}(x,y)$.

L'horloge inférieure LowClock :

Quand le nœud de départ est pair, le premier tic de l'horloge inférieure est toujours initialisé sur le site immédiatement à gauche du nœud de départ :

— $\texttt{Node}(x+1,y) \wedge \texttt{Even}(x+1,y) \rightarrow \texttt{LowClock}(x,y)$.

De même que pour l'horloge supérieure, un nouveau tic d'horloge est initialisé tous les deux sites de la branche gauche jusqu'à atteindre le fils gauche du nœud de départ :

— $\texttt{LeftBranch}(x,y) \wedge \texttt{LowClock}(x+2,y) \wedge \neg\texttt{Node}(x+1,y) \wedge \neg\texttt{Node}(x+2,y) \rightarrow \texttt{LowClock}(x,y)$.

Une fois émis, un signal d'horloge inférieure se dirige suivant le vecteur $(0,-1)$ jusqu'à croiser un signal filtre en phase *delete* ou la diagonale inférieure :

— $\texttt{LowClock}(x,y-1) \wedge \neg\texttt{LowDiag}(x,y-1) \wedge \neg\texttt{LowDelete}(x,y-1) \rightarrow \texttt{LowClock}(x,y)$.

Les filtres inférieurs `LowCross` et `LowDelete` :

La construction des signaux filtres inférieurs est similaire à celle des filtres supérieurs. Un nouveau signal filtre en phase *cross* est initialisé sur le site juste à gauche d'une intersection entre un signal d'horloge inférieure et la diagonale inférieure, sauf si cette intersection a lieu le long de la branche droite :

— $\texttt{LowDiag}(x+1,y) \wedge \texttt{LowClock}(x+1,y) \wedge \neg\texttt{RightBranch}(x+1,y) \rightarrow \texttt{LowCross}(x,y)$.

Il se meut ensuite diagonalement et alterne entre ses phases *cross* et *delete* :

— $\texttt{LowCross}(x+1,y-1) \wedge \neg\texttt{LowClock}(x+1,y-1) \wedge \neg\texttt{RightBranch}(x+1,y-1) \rightarrow \texttt{LowCross}(x,y)$;
$\texttt{LowDelete}(x+1,y-1) \wedge \neg\texttt{LowClock}(x+1,y-1) \wedge \neg\texttt{RightBranch}(x+1,y-1) \rightarrow \texttt{LowDelete}(x,y)$;

— $\texttt{LowCross}(x+1,y) \wedge \texttt{LowClock}(x+1,y) \wedge \neg\texttt{RightBranch}(x+1,y) \rightarrow \texttt{LowDelete}(x,y)$;
$\texttt{LowDelete}(x+1,y) \wedge \texttt{LowClock}(x+1,y) \wedge \neg\texttt{RightBranch}(x+1,y) \rightarrow \texttt{LowCross}(x,y)$.

Les nœuds de la branche droite :

Contrairement au cas de la branche gauche, l'intersection entre une diagonale inférieure et la branche droite ne crée pas de nouveau nœud. Ainsi, seulement deux clauses sont nécessaires pour définir la création d'un nœud à chaque intersection entre la branche droite et un signal filtre.

— $\texttt{LowCross}(x,y) \wedge \texttt{RightBranch}(x,y) \rightarrow \texttt{Node}(x,y)$;

— $\texttt{LowDelete}(x,y) \wedge \texttt{RightBranch}(x,y) \rightarrow \texttt{Node}(x,y)$.

Cas impair, les nœuds des branches gauche et droite (triangles inférieurs et supérieurs)

Dans cette section, nous détaillons la construction des nœuds se trouvant sur la branche gauche partant du fils droit et sur la branche droite partant du fils gauche d'un nœud de départ impair. Puisque l'on ne veut pas que les feuilles de l'arbre mènent à la construction de nouveaux nœuds, on utilisera la conjonction $\texttt{Node}(\delta) \wedge \texttt{Odd}(\delta) \wedge \neg\texttt{Leaf}(\delta)$ pour se référer aux nœuds impairs qui ne sont pas des feuilles.

Initialisation des diagonales avec `UpDiag` et `LowDiag` :

La trajectoire des diagonales supérieures et inférieures reste la même quand le nœud de départ est impair. Le seul changement est leur initialisation : les deux diagonales sont initialisées un site plus haut que dans le cas d'un nœud de départ pair :

— $\texttt{Node}(x,y-2) \wedge \texttt{Odd}(x,y-2) \wedge \neg\texttt{Leaf}(x,y-2) \rightarrow \texttt{UpDiag}(x,y)$;

— $\texttt{Node}(x+1,y-2) \wedge \texttt{Odd}(x+1,y-2) \wedge \neg\texttt{Leaf}(x+1,y-2) \rightarrow \texttt{LowDiag}(x,y)$.

Initialisation des horloges avec `UpClock`, `LowClock` et `LowCross` :

Le comportement général des signaux d'horloges inférieure et supérieure reste lui aussi inchangé quand le nœud de départ est impair. Le changement concernant l'initialisation de l'horloge supérieure est le même que celui de la diagonale supérieure, le premier tic est émis un site plus haut que dans le cas pair :

— $\texttt{Node}(x,y-3) \wedge \texttt{Odd}(x,y-3) \wedge \neg\texttt{Leaf}(x,y-3) \rightarrow \texttt{UpClock}(x,y)$.

Le changement concernant l'initialisation de l'horloge inférieure est quelque peu différent, afin de simuler un premier tic sur le nœud de départ lui même, le premier tic de l'horloge inférieure commence deux sites à gauche du nœud de départ et un signal filtre en phase *cross* est initialisé sur le site juste en haut à droite du nœud de départ :

— $\texttt{Node}(x+2,y) \wedge \texttt{Odd}(x+2,y) \wedge \neg\texttt{Leaf}(x+2,y) \rightarrow \texttt{LowClock}(x,y)$;

— $\texttt{Node}(x+1,y-1) \wedge \texttt{Odd}(x+1,y-1) \wedge \neg\texttt{Leaf}(x+1,y-1) \rightarrow \texttt{LowCross}(x,y)$.

En résumé, après cette longue section, nous avons toutes les clauses définissant l'arbre de potentiel et ses feuilles. De la caractérisation de ses feuilles, nous pouvons maintenant définir le langage Culik.

4.4.3 Appartenance de Culik à incl-ESO-IND

On note $\mathbf{R}_{\texttt{Culik}}$ l'ensemble des prédicats de calcul que nous avons introduits : $\mathbf{R}_{\texttt{Culik}} := \{\texttt{Even}, \texttt{Odd}, \texttt{LeftBorder}, \texttt{RightBorder}, \texttt{LeftGeneral}, \texttt{RightGeneral}, \texttt{StartLeftBranch}, \texttt{StartRightBranch}, \texttt{Node}, \texttt{Leaf}, \texttt{LeftBranch}, \texttt{RightBranch}, \texttt{UpDiag}, \texttt{LowDiag}, \texttt{UpClock}, \texttt{LowClock}, \texttt{UpCross}, \texttt{UpDelete}, \texttt{LowCross}, \texttt{LowDelete}, T_a, T_b, R_0, R_1, R_2, R_3, R_4, R_5\}$.

Soit ψ_0 la conjonction des clauses de calcul définissant ces prédicats, y compris les clauses de $\psi_0^{\mathscr{A}}$. Il est facile de vérifier que la formule ψ_0 est acyclique. Soit un mot $w \in a^+bb^+a^+$ et soit $\mathtt{S}_w := (\langle w \rangle, \mathbf{R}_{\texttt{Culik}})$ l'unique modèle de la conjonction $\forall x \forall y (E(\psi_0) \wedge \psi_{neg}^{\mathscr{A}})$ sur $\langle w \rangle$. En particulier, $\mathtt{S}_w$ est modèle de la conjonction $\forall x \forall y (E(\psi_0^{\mathscr{A}}) \wedge \psi_{neg}^{\mathscr{A}})$.

On va conclure la définition logique du langage Culik à l'aide des deux lemmes suivants :

Lemme 4.4.1 Soit un mot $w \in a^+bb^+ba^+$. On a l'équivalence

$$w \in \texttt{Culik} \iff \mathtt{S}_w \models \texttt{Leaf}(2,n) \vee \texttt{Leaf}(1,n-1).$$

Démonstration. C'est une conséquence directe du fait 4.1 de la section 4.2 : le site $(x,y) = (1,n)$ est de potentiel 0 *ssi* l'un au moins des sites $(x+1,y) = (2,n)$ ou $(x,y-1) = (1,n-1)$ est une feuille de l'arbre T. □

Lemme 4.4.2 Soit un mot $w \in a^+bb^+a^+$. On a l'équivalence

$$w \in \texttt{Culik} \iff \mathtt{S}_w \models n \leq 4 \vee \texttt{Leaf}(2,n) \vee \texttt{Leaf}(1,n-1).$$

Démonstration. Le seul mot du langage $a^+bb^+a^+$ de longueur $n \leq 4$ est $abba$: c'est un mot de Culik.

Si w est de taille $n > 4$, alors

- soit w est de la forme $a^ib^2a^j$, pour $i + j > 2$, alors w n'est pas un mot de Culik et l'algorithme ne génère aucun nœud ou feuille car ab^3 et b^3a ne sont pas des facteurs de w,
- soit w est de la forme $a^ib^ka^j$, avec $k \geq 3$ et $i, j \geq 1$: le résultat est alors donné par le lemme 4.4.1.

□

Corollaire 4.4.3 Pour tout $w \in \{a,b\}^+$, on a l'équivalence

$$w \in \texttt{Culik} \iff \langle w \rangle \models \exists \mathbf{R}_{\texttt{Culik}} \forall x \forall y (E(\psi_0) \wedge \psi_{neg}^{\mathscr{A}} \wedge \psi_{neg}^{T})$$

où ψ_{neg}^{T} désigne la clause de contradiction
$x = 1 \wedge y = n \wedge y > 4 \wedge \neg\texttt{Leaf}(x+1,y) \wedge \neg\texttt{Leaf}(x,y-1) \rightarrow \bot$.

Ce qui implique Culik ∈ incl-ESO-IND et Culik ∈ Trellis.

Démonstration. On note que la formule $\forall x \forall y \psi_{neg}^{T}$ est équivalente à la disjonction $n \leq 4 \vee \texttt{Leaf}(2,n) \vee \texttt{Leaf}(1,n-1)$. On conclut avec le lemme 4.4.2.

□

5 Le circuit-grille et les logiques en dimension 3 ou plus

Le présent chapitre, comme une partie du chapitre 6 suivant, est un chapitre exploratoire : on y pose plus de questions qu'on n'en résout ; par ailleurs, plusieurs résultats qu'on va formuler sont donnés sans preuve ou avec seulement une idée de preuve.
L'objectif du présent chapitre est de déterminer dans quelle mesure les résultats du chapitre 3 qui établissent des équivalences entre logique et temps-réel des automates cellulaires en passant par la notion de circuit-grille se généralisent à toutes les dimensions. En particulier, comme on l'a fait en dimension 2, on esquisse une classification complète du pouvoir d'expression des circuits-grille de dimension 3, appelés circuits-cube, selon leur mode d'entrée, c'est-à-dire selon le placement du mot d'entrée $w = w_1 \dots w_n$ sur une arête ou l'une des diagonales du cube.

On reprend ici les définitions du circuit-grille de la section 2.1 que l'on adapte à toute dimension d.

Définition 5.0.1 — circuit-grille de dimension d. Un *circuit-grille* $(C_n^d)_{n>0}$ ou *d-circuit-grille* est une famille de quintuplets $C_n^d := (\Sigma, \texttt{Input}_n, \mathbf{S}, \texttt{f}, \mathbf{A})$, où :
- Σ est *l'alphabet* des mots d'entrée ;
- $\texttt{Input}_n : \Sigma^n \times [1,n]^d \to \Sigma \cup \{\$\}$ est la *fonction d'entrée* ;
- $\mathbf{S} \cup \{\sharp\}$ est l'ensemble fini des *états* ;
- $\texttt{f} : (\mathbf{S} \cup \{\#\})^d \times (\Sigma \cup \{\$\}) \to \mathbf{S}$ est la *fonction de transition* ;
- $\mathbf{A} \subseteq \mathbf{S}$ est l'ensemble des *états acceptants*.

Définition 5.0.2 — calcul d'un circuit-grille de dimension d. Pour un circuit-grille $C_n^d := (\Sigma, \texttt{Input}_n, \mathbf{S}, \texttt{f}, \mathbf{A})$ et un mot $w \in \Sigma^n$, l'état, noté $\langle i_1, \dots, i_d \rangle$, d'un site $(i_1, \dots, i_d) \in [0,n]^d$ est défini inductivement par :
- $\langle i_1, \dots, i_d \rangle := \sharp$ si c'est un site de bordure, c'est-à-dire s'il existe b tel que $i_b = 0$;
- $\langle i_1, \dots, i_d \rangle := \texttt{f}(\langle i_1 - 1, \dots, i_d \rangle, \dots, \langle i_1, \dots, i_d - 1 \rangle, \texttt{Input}_n(w, i_1, \dots, i_d))$ pour $(i_1, \dots, i_d) \in [1,n]^d$.

Le mot w est *accepté* par le circuit-grille si $\langle n, n, \dots, n \rangle \in \mathbf{A}$.
Le langage $L \subseteq \Sigma^+$ *reconnu* par le d-circuit-grille $(C_n^d)_{n>0}$ est l'ensemble des mots $w = w_1 \dots w_n$ qu'il accepte, pour tous les $n > 0$.

La capacité de reconnaissance d'un circuit-cube (3-circuit-grille), comme d'un circuit-grille (2-circuit-grille), dépend uniquement du placement de l'entrée sur le circuit. Dans cette section nous présentons donc les différentes manières "naturelles" de positionner l'entrée sur un cube, un mot d'entrée $w = w_1 \dots w_n$ étant toujours positionné de façon à ce w_1 soit le plus loin de la cellule de sortie (n,n,n) et w_n le plus proche.
Aux symétries près, il existe neuf façons de placer un mot d'entrée $w = w_1 \dots w_n$ sur le cube. Parmi elles :
- quatre reviennent à calculer sur une grille ou une ligne ;
- deux sont des généralisations de $\texttt{GRID}_1$ et $\texttt{GRID}_2$ à 3 dimensions ;
- trois sont "hybrides" et seront comparées aux circuits connus.

Afin de mesurer l'expressivité ou les limites de chaque mode d'entrée, nous indiquerons pour chacun d'entre eux et pour un mot $w = w_1 \dot{w}_n$:
- le nombre nb_{deb} de sites du cube que la lettre w_1 peut influencer ;
- le nombre nb_{fin} de sites du cube que w_n peut influencer ;
- le nombre $nb_{deb-fin}$ de sites du cube pouvant être influencés à la fois par les deux extrémités

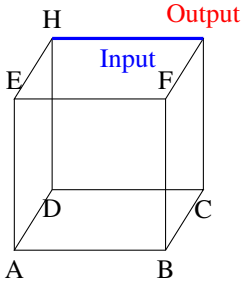
w_1 et w_n.

5.1 Les modes d'entrée équivalents aux dimensions inférieurs

Cette section regroupe les quatre façons de placer l'entrée sur le cube redondantes avec les dimensions inférieures. On y retrouve le placement équivalent au calcul sur automate fini ainsi que les trois modes d'entrée correspondants aux circuits-grilles $\texttt{GRID}_1$, $\texttt{GRID}_2$ et $\texttt{GRID}_3$.

Équivalent aux automates finis

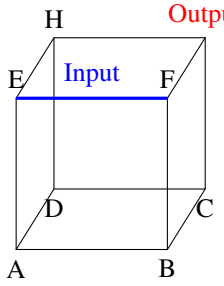
Placer le mot d'entrée sur une des arêtes contenant la cellule de sortie notée Ouput, i.e. la lettre w_i est placée sur le site (i,n,n) (symétriquement sur les sites (n,i,n) ou (n,n,i)), revient à calculer sur un automate fini.



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$F-Output$ $H-Output$ $C-Output$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$

Équivalent à $\texttt{GRID}_1$

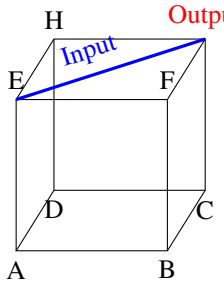
Placer l'entrée sur une arête de la face du cube contenant Output, arête ne contenant pas Output, par exemple placer la lettre w_i sur le site $(i,1,n)$, revient à calculer sur $\texttt{GRID}_1$.



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$E-F,\ E-H,$ $B-F,\ B-C,$ $D-H,\ D-C$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$

Équivalent à $\texttt{GRID}_2$

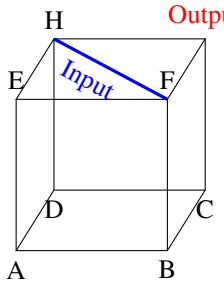
Placer l'entrée sur la diagonale d'une face du cube contenant Output, diagonale contenant Output, par exemple placer la lettre w_i sur le site (i,i,n), revient à calculer sur $\texttt{GRID}_2$.



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$E-Output$ $A-Output$ $B-Output$	$\Theta(n^2)$	$\Theta(1)$	$\Theta(1)$

Équivalent à $\texttt{GRID}_3$

Placer l'entrée sur l'anti-diagonale d'une face du cube contenant Output, par exemple placer la lettre w_i sur le site $(i, n-i+1, n)$, revient à calculer sur $\texttt{GRID}_3$.



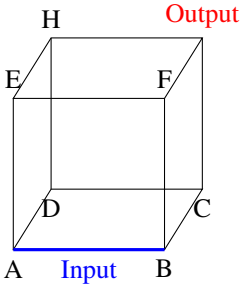
Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$H-F$			
$F-C$	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$
$C-H$			

5.2 Les classes de grille s'étendant à toutes les dimensions

Les deux modes d'entrée évoqués dans cette section apparaissent comme des généralisations à 3 dimensions de $\texttt{GRID}_1$ et $\texttt{GRID}_2$. On appelle $\texttt{GRID}_1^3$ et $\texttt{GRID}_2^3$ ces deux circuits-cube. Cette généralisation s'étend naturellement à toute dimension. On définit ainsi pour chacun de ces modes d'entrée et pour toute dimension d, un circuit-grille de dimension d et une logique de Horn sur d variables, tous deux équivalents à une classe de complexité temps-réel d'automates cellulaires à $d-1$ dimensions. On se contente ici de définir ces différents éléments sans prouver leur équivalence, les preuves utilisées en dimension 2 se généralisant elles aussi naturellement.

Généralisation de $\texttt{GRID}_1$

Généraliser le circuit $\texttt{GRID}_1$ revient à placer le mot d'entrée sur une des arêtes opposées au site de sortie, par exemple en plaçant la lettre w_i sur le site $(i,1,1)$. Ce mode d'entrée se généralise naturellement à toute dimension. Dans le circuit-grille $\texttt{GRID}_1^d$ de dimension d cela revient, par exemple, à placer la lettre w_i sur le site $(i,1\ldots1)$.

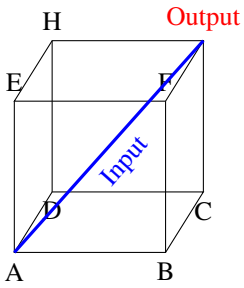


Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$A-B$			
$A-D$	$\Theta(n^3)$	$\Theta(n^2)$	$\Theta(n^2)$
$A-E$			

Généralisation de $\texttt{GRID}_2$

Généraliser le circuit $\texttt{GRID}_2$ revient à placer le mot d'entrée sur la diagonale du cube contenant la cellule de sortie Output, i.e. à placer la lettre w_i sur le site (i,i,i). Ce mode d'entrée diagonal

se généralise lui aussi naturellement. Dans le circuit-grille $\texttt{GRID}_2^d$ de dimension d la lettre w_i est placée sur le site $(i,\ldots,i)$.



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$A - Output$	$\Theta(n^3)$	$\Theta(1)$	$\Theta(1)$

5.2.1 Équivalence avec les automates cellulaires.

En effectuant des transformations analogues au cas de $\texttt{GRID}_1$ (voir section 3.1.4), on peut montrer que $\texttt{GRID}_1^d$ est équivalent au temps-réel des IA avec voisinage de von Neumann unidirectionnel sur $d-1$ dimensions, que l'on note $(d-1)$-$\texttt{RealTime}_{\texttt{OIA}}$. La figure 5.1 montre le diagramme espace-temps de 2-$\texttt{RealTime}_{\texttt{OIA}}$, classe équivalente au circuit-cube $\texttt{GRID}_1^3$.

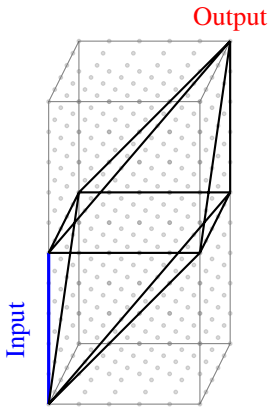


FIGURE 5.1 – Diagramme espace-temps de la classe 2-$\texttt{RealTime}_{\texttt{OIA}}$

De la même façon, en effectuant des transformations analogues à celles utilisées dans la section 3.2.4 pour montrer l'équivalence entre $\texttt{GRID}_2$ et $\texttt{RealTime}_{\texttt{IA}}$, on montre que $\texttt{GRID}_2^d$ est équivalent au temps-réel des IA sur d-1 dimensions avec voisinage de von Neumann, que l'on note $(d$-$1)$-$\texttt{RealTime}_{\texttt{IA}}$. La figure 5.2 donne le diagramme espace-temps de 2-$\texttt{RealTime}_{\texttt{IA}}$, équivalent au circuit $\texttt{GRID}_2^3$.

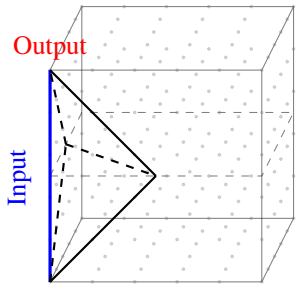


FIGURE 5.2 – Diagramme espace-temps de la classe 2-$\texttt{RealTime}_{\texttt{IA}}$

5.2.2 Généralisation des logiques

On définit pour chaque circuit-grille $\texttt{GRID}_1^d$ (resp. $\texttt{GRID}_2^d$) une logique de Horn d-pred-ESO-HORN (resp. d-pred-dio-ESO-HORN) sur d variables qui lui est équivalente. On ne donne ici que la version Horn de ces logiques mais leur version inductive s'en déduit naturellement.

Définition 5.2.1 — logiques de prédécesseur à d variables. Une formule de d-pred-ESO-HORN (resp. d-pred-dio-ESO-HORN) est une formule $\Phi = \exists \mathbf{R}(\forall x_i)_{1\leq i\leq d}\psi(x_1,\ldots,x_d)$ où :
$\mathbf{R}$ est un ensemble de prédicats d'arité d ;
ψ est une conjonction de clauses de Horn, de signature $\mathscr{S}_\Sigma \cup \mathbf{R}$;
les clauses de ψ sont de la forme $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow \delta_0$ où la conclusion δ_0 est soit un atome de calcul $R(x_1,\ldots,x_d)$ avec $R \in \mathbf{R}$ soit $\bot$ et où chaque hypothèse δ_i est

- soit un littéral d'entrée (resp. une conjonction d'entrée) de la forme :
  — $Q_s(x_i - a)$ avec $1 \leq i \leq d$ (resp. $Q_s(x_1 - a) \wedge \bigwedge_{1\leq i\leq d} x_1 = x_i$), pour $s \in \Sigma$ et $a \geq 0$,
  — $U(x_i - a)$ ou $\neg U(x_i - a)$, pour $U \in \{\texttt{min},\texttt{max}\}$, $1 \leq i \leq d$ et $a \geq 0$,
- soit un atome de calcul ou une conjonction de calcul :
  $S(x_{\sigma(1)},\ldots,x_{\sigma(i)},\ldots,x_{\sigma(d)})$;
  $S(x_{\sigma(1)} - a_1,\ldots,x_{\sigma(i)} - a_i,\ldots,x_{\sigma(d)} - a_d) \wedge \bigwedge_{i\in\{1,\ldots,d\}} x_{\sigma(i)} > a_i$;
  avec $S \in \mathbf{R}$, σ une permutation de $\{1,\ldots,d\}$ et $\forall i\ a_i \geq 0$ et $\sum_i a_i > 0$.

En utilisant le même processus de preuve qu'en dimension 2, la classe d-pred-ESO-HORN (resp. d-pred-dio-ESO-HORN) des langages pouvant être définis par une formule de d-pred-ESO-HORN (resp. d-pred-dio-ESO-HORN) peut être montrée égale à la classe $(d-1) - \texttt{RealTime}_{\texttt{OIA}}$ (resp. $(d-1) - \texttt{RealTime}_{\texttt{IA}}$). C'est ce que dit la proposition 5.2.1 suivante.

Proposition 5.2.1 Soit L un langage non vide, alors on a, pour tout $d \geq 2$, les deux équivalences :

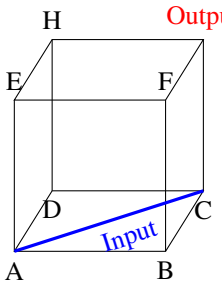
- $L \in d$-pred-ESO-HORN $\iff L \in (d$-1)-$\texttt{RealTime}_{\texttt{OIA}}$
- $L \in d$-pred-dio-ESO-HORN $\iff L \in (d$-1)-$\texttt{RealTime}_{\texttt{IA}}$

5.3 Autres variantes

Les circuits-cube présentés dans cette sous-section ne sont ni des circuits équivalents aux circuits de dimensions inférieures, ni des généralisations de ceux-ci. On peut toutefois comparer les classes définies par ces circuits aux autres classes déjà connues, ceci à l'aide du triplet $(nb_{deb}, nb_{fin}, nb_{deb-fin})$. Il est par exemple évident que chacune de ces trois classes est inclue dans $\texttt{GRID}_1^3$, la classe de circuit-cube la plus expressive.

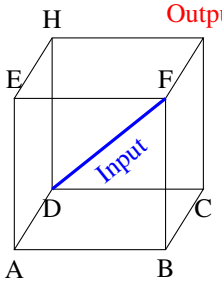
5.3.1 Deux circuits-cube intermédiaires

L'entrée de ce circuit-cube est placée le long de la diagonale d'une face opposée au site de sortie, diagonale adjacente à une arête contenant le site du sortie. Par exemple, pour un mot $w = w_1 \ldots w_n$, la lettre w_i est placée sur le site $(i,i,1)$. Avec ce mode d'entrée, la i-ème lettre (w_i) du mot d'entrée influence un nombre de sites égal à $(n-i+1)^2 \times n$.
L'ensemble des langages pouvant être reconnus par ce circuit contient la classe $\texttt{GRID}_2^3$ ainsi que $\texttt{GRID}_1$. Le nombre de sites nb_{fin} que w_n peut influencer ainsi que le nombre $nb_{deb-fin}$ de sites pouvant êtres influencés à la fois par w_1 et w_n sont du même ordre que ceux de $\texttt{GRID}_1$.



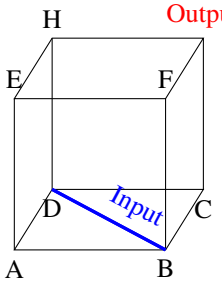
Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$A-H$			
$A-F$	$\Theta(n^3)$	$\Theta(n)$	$\Theta(n)$
$A-C$			

L'entrée du deuxième circuit-cube intermédiaire est placée de la façon suivante : la lettre w_i est placée sur le site $(i, n-i+1, i)$. La classe de langages associée à ce circuit contient les deux classes $\mathtt{GRID}_2$ et $\mathtt{GRID}_3$ mais pas $\mathtt{GRID}_1$. Par ailleurs, cette classe est incluse dans la classe définie par le circuit présenté juste ci-dessus : on peut, par exemple, simuler le placement d'un mot d'entrée le long de la diagonale $D-F$ à partir d'un mot d'entrée placé le long de la diagonale $A-F$.



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$B-H$			
$E-C$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(1)$
$D-F$			

5.3.2 Un autre circuit-cube intéressant ?



Entrée sur le cube	nb_{deb}	nb_{fin}	$nb_{deb-fin}$
$D-B$			
$B-E$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$
$E-D$			

On note $\mathtt{Cube}_{\mathtt{diag}}$ la classe des langages reconnus par le circuit-cube avec le mot d'entrée placé sur l'anti-diagonale d'une des faces opposées au site de sortie : la lettre w_i du mot $w = w_1 \dots w_n$ est placée sur le site $(i, n-i+1, 1)$.

On définit ici la logique `incl-pred-ESO-HORN`. Comme son nom l'indique, cette logique est un hybride entre la logique inclusive et la logique du prédécesseur. Là encore, les deux variables x et y définissent une logique d'intervalle et permettent d'accéder à l'entrée. La variable z, quant à elle, est utilisée comme dans la logique de prédécesseur à ceci près qu'elle ne permet pas d'accéder à l'entrée.
Cette logique semble peu naturelle, car non symétrique. Pourtant, comme nous le verrons dans le chapitre 6 suivant, elle permet de décrire des langages intéressants comme l'ensemble des langages conjonctifs (une extension des langages algébriques).

Définition 5.3.1 — La logique hybride "inclusive-prédécesseur". Une formule hybride "inclusive-prédécesseur" est une formule $\Phi = \exists \mathbf{R} \forall x \forall y \forall z \psi(x,y,z)$ où
$\mathbf{R}$ est un ensemble de prédicats ternaires ;
ψ est une conjonction de clauses de Horn de signature $\mathscr{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq, <\}$;

les clauses de ψ sont de la forme $x \leq y \wedge \delta_1 \wedge \ldots \wedge \delta_r \rightarrow \delta_0$ où la conclusion δ_0 est soit un atome de calcul $R(x,y,z)$, soit le faux ($\perp$), et chaque hypothèse δ_i est

1. soit un littéral d'entrée de la forme $U(x+a)$, $\neg U(x+a)$, $U(y+a)$ ou $\neg U(y+a)$, pour $U \in \{(Q_s)_{s\in\Sigma}, \texttt{min}, \texttt{max}\}$ et $a \in \mathbb{Z}$,
2. soit l'(in)égalité $x = y$ ou $x < y$,
3. soit une conjonction de la forme $x+a \leq y-b \wedge z > c \wedge S(x+a, y-b, z-c)$ pour des entiers $a,b,c \geq 0$,
4. soit une conjonction de la forme $x+a \leq y-b \wedge z > c \wedge \neg S(x+a, y-b, z-c)$ pour des entiers $a,b,c \geq 0$ tels que $a+b+c > 0$.

La classe `incl-pred-ESO-HORN` des langages définis par une formule de `incl-pred-ESO-HORN` peut être montrée égale à la classe $\texttt{Cube}_{\texttt{diag}}$ des langages décidables sur le circuit $\texttt{Cube}_{\texttt{diag}}$, c'est ce que dit la proposition 5.3.1 ci-dessous. Comme pour la proposition 5.2.1, la preuve de cette proposition est similaire aux preuves des propositions analogues en dimension 2 et n'est donc pas donnée ici.

Proposition 5.3.1 Soit L un langage non vide, alors :

- $L \in \texttt{incl-pred-ESO-HORN} \iff L \in \texttt{Cube}_{\texttt{diag}}$

6 Les langages conjonctifs

Les langages conjonctifs et les langages booléens ont été introduits par Alexander Okhotin au début des années 2000 [35, 36]. Ces deux familles de langages sont une extension des langages algébriques (CF). Les règles de grammaire des langages conjonctifs sont en effet les mêmes que les algébriques avec l'ajout d'une opération de conjonction. Ces langages conjonctifs sont eux-mêmes étendus en langages booléens par l'ajout d'une opération de négation. On présente ici ces langages dans leur forme la plus générale ainsi que deux restrictions intéressantes : la restriction à un alphabet d'entrée unaire et la restriction linéaire. Ces classes de langages sont mises en lien, à travers l'inclusion ou l'égalité, avec nos classes de définissabilité logique et les classes de complexité temps-réel des automates cellulaires.

6.1 Définitions

Dans cette section nous définissons les langages conjonctifs ainsi que leur forme normale.

6.1.1 Langages conjonctifs

Un langage conjonctif est un langage dont la grammaire est définie comme suit :

Définition 6.1.1 — Grammaire conjonctive. Une grammaire conjonctive est un quadruplet $G = (\Sigma, N, P, S)$ avec :
- Σ l'ensemble fini des symboles terminaux ;
- N l'ensemble fini des symboles non terminaux ;
- P l'ensemble fini des règles de réécriture (productions) ;
- $S \in N$ le symbole initial de la grammaire (*axiome*).

Règles de la grammaire :

Les règles autorisées par les grammaires conjonctives sont celles des grammaires algébriques enrichie d'une opération de conjonction.
Chacune des règles $r \in P$ d'une grammaire conjonctive est de la forme :

$$A \to \alpha_1 \& \ldots \& \alpha_m \ (m \geq 1, \alpha_i \in (\Sigma \cup N)^*)$$

On appelle *tandems* de la règle les éléments $\alpha_1, \ldots \alpha_m$. Si un mot w peut être dérivé de chacun des tandems α_i, on dit alors que ce mot a la propriété syntaxique A et on note $w \in L(A)$. Un mot w appartient au langage défini par la grammaire $G = (\Sigma, N, P, S)$ si et seulement si $w \in L(S)$.

■ **Exemple 6.1** [39] L'ensemble des palindromes de longueur impaire sur l'alphabet $\Sigma = \{a, b\}$ peut être défini aisément par une grammaire algébrique avec les règles $S \to aSa \mid bSb \mid a \mid b$. On donne ici une version conjonctive de cette grammaire, celle-ci est construite de manière ad hoc afin de montrer le fonctionnement de l'opérateur & sur un exemple.
La grammaire conjonctive suivante génère l'ensemble des palindromes de longueur impaire sur un alphabet $\Sigma = \{a, b\}$:

$$\begin{array}{lcl} S & \to & AB\&O \mid a \mid b \\ A & \to & aSa \mid \varepsilon \\ B & \to & bSb \mid \varepsilon \\ O & \to & OOO \mid a \mid b \end{array}$$

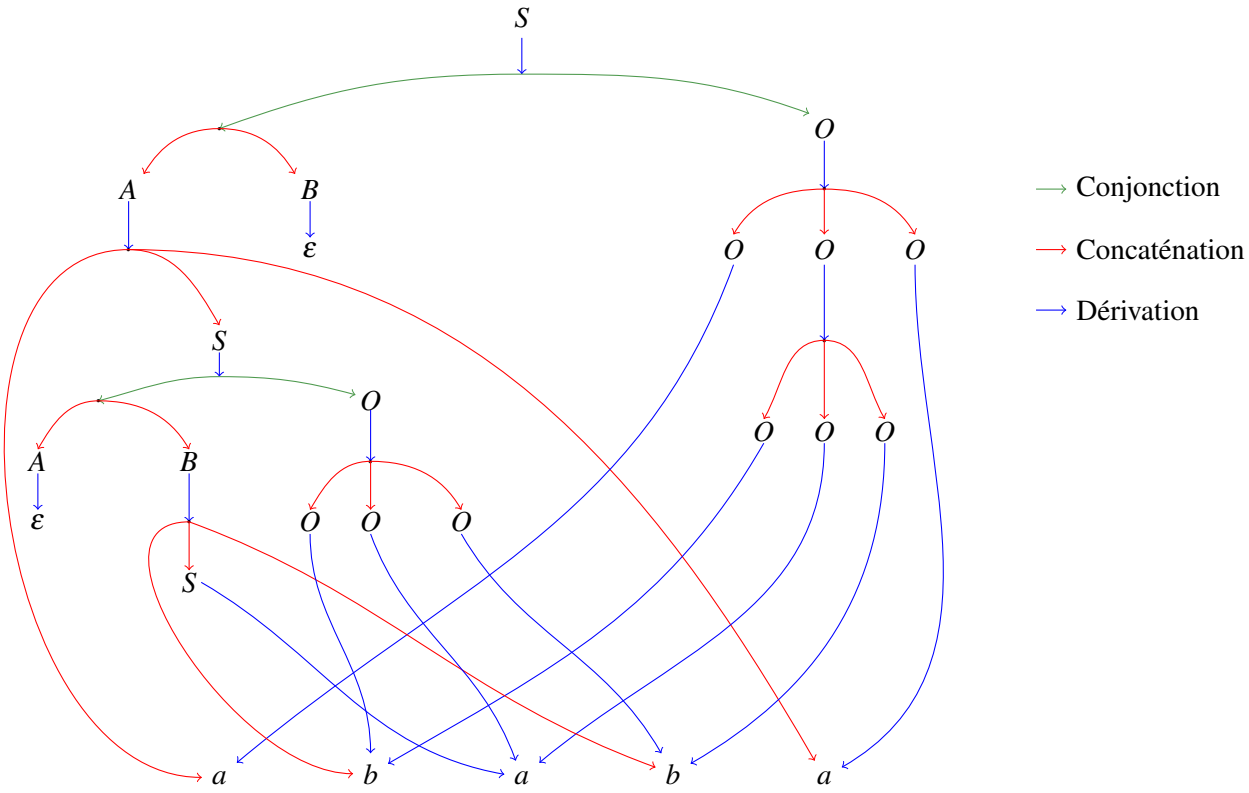


FIGURE 6.1 – Arbre de dérivation du mot *ababa*

Lors d'une dérivation l'opération de conjonction ne peut être simplifiée que si les deux chaînes de caractères de part et d'autre de cette conjonction sont égales. La figure 6.1 montre un exemple de dérivation de cette grammaire pour le palindrome *ababa*.

Afin de démontrer la validité de la construction de la grammaire, celle-ci peut être vue comme un système d'équations de langages sur les symboles non terminaux. Cette méthode est déjà connue pour les grammaires algébriques où la disjonction correspond à l'union et les symboles terminaux aux constantes (chaque terminal a est remplacé par le singleton $\{a\}$). Dans le cas des grammaires conjonctives, l'opération de conjonction & correspond à l'intersection. Le système d'équations ainsi défini admet une solution minimale qui est bien le langage défini par la grammaire. Cette validité est obtenue pour les mêmes raisons que dans le cas des grammaires algébriques (l'intersection respecte les mêmes conditions de monotonie et de continuité que les autres opérations).

Dans notre exemple, le symbole non terminal O génère le langage algébrique trivial de tous les mots de taille impaire. Les mots générés par S sont donc eux aussi de taille impaire. La concaténation AB génère quant à elle l'ensemble $aSabSb \cup aSa \cup bSb \cup \{\varepsilon\}$. S ne générant que des mots de taille impaire, $aSabSb$ ne génère que des mots de taille paire. L'intersection de AB avec O élimine les mots de longueur paire, on peut donc en déduire que lors de la dérivation de S, $A \to \varepsilon$ ou $B \to \varepsilon$. L'équation obtenue pour la langage (à partir de S) est donc :

$$(aSa \cup \{\varepsilon\})(bSb \cup \{\varepsilon\}) \cap Imp$$

$$= (aSabSb \cup aSa \cup bSb \cup \{\varepsilon\}) \cap Imp = aSa \cup bSb$$

On retrouve ainsi la disjonction présente dans la grammaire algébrique donnée plus haut. ∎

Notation.
On note `Conj` l'ensemble des langages conjonctifs.

Remarque 6.1 — Langages booléens. [36] Les langages booléens sont les langages pouvant être décrits par une grammaire booléenne. Les grammaires booléennes sont une extension des grammaires conjonctives obtenue en ajoutant une opération de négation. Alors qu'il n'est pas prouvé que la négation augmente l'expressivité de la grammaire, celle-ci rend plus difficile de déterminer si une grammaire est bien définie, notamment parce que la négation ne permet plus d'utiliser la sémantique de la solution minimale.

6.1.2 Formes normales binaires

De la même façon que toute grammaire algébrique peut être réécrite de façon équivalente sous sa forme normale de Chomsky [42], toute grammaire conjonctive peut être réécrite de façon équivalente sous sa forme normale binaire.

Définition 6.1.2 — Forme normale binaire. [36] Une grammaire conjonctive qui engendre un langage $L \subseteq \Sigma^+$ est sous forme normale binaire si toutes ses règles sont de l'une des formes suivantes :
règle longue :

$$A \to B_1C_1\&\ldots\&B_mC_m \ (m \geq 1, B_i, C_j \in N)$$

règle courte :

$$A \to a \ (a \in \Sigma)$$

La forme normale binaire des grammaires conjonctives est notamment utilisée dans les algorithmes de reconnaissance de langage comme l'algorithme de Cocke-Kasami-Younger dont le fonctionnement est détaillé dans la prochaine section.

6.2 Reconnaissance des langages conjonctifs

6.2.1 L'algorithme de Cocke-Kasami-Younger

Dans [38], A. Okhotin présente une généralisation de l'algorithme séquentiel de Cocke-Kasami-Younger, noté CKY, permettant la reconnaissance d'un langage engendré par une grammaire conjonctive fixée en temps $O(n^3)$ et espace $O(n^2)$ sur machine RAM. On rappelle ici son algorithme.

Soient $G = (\Sigma, N, P, S)$ une grammaire conjonctive fixée sous forme normale binaire et $w = w_1 \ldots w_n \ (n \geq 1)$ un mot d'entrée fixé. Pour tout $0 \leq i < j \leq n$, on définit :

$$T_{i,j} = \{A \in N \mid w_{i+1} \ldots w_j \in L(A)\}$$

Ainsi $T_{i,j}$ est l'ensemble des symboles de N dont peut être dérivé le sous-mot $w[i,j] = w_{i+1} \ldots w_j$. Le mot w appartient donc au langage L engendré par la grammaire G si et seulement si on a $S \in T_{0,n}$.

Concaténation.

Pour tout i et tout j, tels que $j-i \geq 2$, le mot $w[i,j] = w_{i+1} \dots w_j \in L(B)L(C)$ avec $B,C \in N$ si et seulement si :

$$(B,C) \in \bigcup_{k=i+1}^{j-1} T_{i,k} \times T_{k,j}$$

En effet $w[i,j] = w_{i+1} \dots w_j \in L(B)L(C)$ si et seulement si il existe une factorisation de $w[i,j]$ telle que $w_{i+1} \dots w_k \in L(B)$ et $w_{k+1} \dots w_j \in L(C)$ (avec $i < k < j$). Cela revient à dire que $B \in T_{i,k}$ et $C \in T_{k,j}$, ou encore que $(B,C) \in T_{i,k} \times T_{k,j}$.

Soit $G = (\Sigma, N, P, S)$ une grammaire conjonctive sous forme normale binaire. Les longues règles de cette grammaire s'écrivent sous la forme :

$$A \to B_1C_1 \& \dots \& B_mC_m$$

En examinant le fonctionnement de la grammaire et la définition des $T_{i,j}$ on obtient la formule d'induction suivante pour $0 \leq i < j \leq n$ et $j-i \geq 2$:

$T_{i,j} = \{A \in N \mid$ il existe une règle $A \to B_1C_1 \& \dots \& B_mC_m \in P$ telle que $\forall h = 1, \dots, m$
$\exists k$ avec $1 \leq k < j-i$ tel que $B_h \in T_{i,i+k}$ et $C_h \in T_{i+k,j}\}$

On note $f : 2^N \times 2^N \to 2^N$ la fonction définie comme suit :

$f(R) = \{A \in N \mid$ il existe une règle $A \to B_1C_1 \& \dots \& B_mC_m \in P$ telle que chaque couple $(B_i, C_i) \in R\ \}$

La formule d'induction s'exprime de façon synthétique à l'aide de la fonction f :

$$T_{i,j} = f\left(\bigcup_{1 \leq k \leq j-i} T_{i,i+k} \times T_{i+k,j} \right), \text{ pour } j-i \geq 2.$$

Algorithme.

Soit $w = w_1 \dots w_n \in \Sigma^* (n \geq 1)$ le mot d'entrée. Le but de l'algorithme CKY est de calculer les ensembles $T_{i,j}$ pour tous i, j, tels que $0 \leq i < j \leq n$. On appelle *amplitude* d'un ensemble $T_{i,j}$, la longueur $j-i$ du sous-mot $w_{i+1} \dots w_j$ qui lui est associé. La première étape du calcul est la fabrication des ensembles $T_{i,j}$ d'amplitude 1. Les ensembles $T_{i,j}$ d'amplitudes $d > 1$ sont ensuite calculés dans l'ordre croissant des amplitudes à l'aide de la fonction f et des ensembles d'amplitudes strictement inférieures à d jusqu'à obtenir finalement $T_{0,n}$ pour $d = n$ (voir l'algorithme 1).

Pour i **de** 1 **à** n **faire**
  $T_{i-1,i} \leftarrow \{A \mid A \to w_i \in P\}$
Fin Pour
Pour d **de** 2 **à** n **faire**
  Pour i **de** 0 **à** $n-d$ **faire**
    $j \leftarrow i+d$
    $R \leftarrow \varnothing$
    Pour k **de** i+1 **à** j-1 **faire**
      $R \leftarrow R \cup T_{i,k} \times T_{k,j}$
    Fin Pour
    $T_{i,j} \leftarrow f(R)$
  Fin Pour

Fin Pour

Algorithme 1 – Algorithme séquentiel de reconnaissance de langage conjonctif en espace $O(n^2)$ et en temps $O(n^3)$

Complexité.

Puisque la grammaire G est fixée , la taille de chaque ensemble $T_{i,j}$ et celle de la fonction f sont bornées par une constante. Les instructions $R \leftarrow R \cup T_{i,k} \times T_{k,j}$ et $T_{i,j} \leftarrow f(R)$ s'exécutent donc chacune en temps constant. La structure des trois boucles "Pour" imbriquées justifie une complexité en temps $O(n^3)$.

Parallélisation.

Cet algorithme se parallélise naturellement, en effet les ensembles $T_{i,j}$ de même amplitude sont indépendants les uns des autres et peuvent donc être calculés en parallèle, il en est de même pour les produits faisant intervenir des ensembles de même amplitude. On réduit ainsi le temps de l'algorithme 1 au temps de sa seule boucle externe sur l'amplitude d des ensembles. On illustre dans la section 6.2.3 une parallélisation de cet algorithme sur un circuit-grille de dimension 3 (circuit-cube). Cette parallélisation se déduit naturellement de la formule de `incl-pred-ESO-HORN` qu'on va construire dans la section 6.2.2 définissant le complémentaire d'un langage conjonctif donné.

6.2.2 Définir en logique les langages conjonctifs

Proposition 6.2.1 Soit `Conj` l'ensemble des langages conjonctifs ne contenant pas le mot vide. On a l'inclusion `Conj` $\subseteq$ `incl-pred-ESO-HORN`.

D'où on tire, avec les classes introduites au chapitre précédent :

Corollaire 6.2.2 On a l'inclusion `Conj` $\subseteq$ $\texttt{Cube}_{\texttt{diag}}$ et donc `Conj` $\subseteq$ 2-$\texttt{RealTime}_{\texttt{OIA}}$. Autrement dit, tout langage conjonctif est reconnu en temps-réel par un `OIA` *(one-way iterative array)* de dimension 2.

Remarque 6.2 Le corollaire améliore un théorème de 1975 dû à Kosaraju (voir [28], Théorème 1) : Tout langage algébrique est reconnu en temps linéaire par un "2-dimensional array automaton" (une variante de l'`OIA`).

Outre que notre résultat concerne la classe des langages conjonctifs, définie seulement en 2001 [35], qui inclut strictement la classe des langages algébriques, il donne une borne de complexité plus précise, en temps-réel plutôt qu'en temps linéaire.

Nous pensons que notre résultat vaut surtout par la méthode de sa preuve. Alors que Kosaraju [28] démontre son théorème par une preuve qui nous a semblé "ad hoc" et technique, d'idée difficile à dégager, notre démonstration est une application de notre méthode de programmation logique, méthode générale qui fait l'unité de cette thèse.

Preuve de la proposition 6.2.1. Soit un langage $L \subseteq \Sigma^+$ généré par une grammaire conjonctive $G = (\Sigma, N, P, S)$ qu'on supposera sous forme normale binaire. On va construire une formule $\Phi_G \in$ `incl-pred-ESO-HORN` définissant le langage $\Sigma^+ \setminus L$. Ce qui donnera $L \in$ `incl-pred-ESO-HORN` puisque la classe `incl-pred-ESO-HORN` est close pour le complémentaire, ce qu'on admettra ici (preuve similaire à celles qu'on a données pour les classes `pred-ESO-HORN` et `incl-ESO-HORN`).

Principe de construction de la formule : La dérivation par la grammaire G d'un mot $w \in L$ de longueur supérieure à 1 ou, plus généralement, la dérivation d'un facteur $w_x \dots w_y \in L(A)$, pour un symbole $A \in N$ et $x < y$, commence par l'application d'une règle de la forme $A \to B_1C_1 \& \dots \& B_mC_m$. Or, tout noeud $X \in N$ (différent de la racine) d'un arbre de dérivation de la grammaire pour un mot $w \in L$, noeud qui génère un facteur propre u de w, doit être généré dans l'arbre par une règle $A \to B_1C_1 \& \dots \& B_mC_m$ dont X est soit un B_i, soit un C_i, $1 \leq i \leq m$. Autrement dit, on a l'alternative suivante :

— soit X est un B_i et $u \in L(B_i)$ est un *préfixe* propre d'un facteur de w qui appartient à $L(A)$,
— soit X est un C_i et $u \in L(C_i)$ est un *suffixe* propre d'un facteur de w qui appartient à $L(A)$.

Construction de la formule : Pour un mot d'entrée $w = w_1 \dots w_n$, les variables x et y de la formule à construire indiquent le facteur $w_x \dots w_y$ de w, pour $x \leq y$, sur lequel on travaille. Notre objectif est de déterminer, pour chaque facteur $w_x \dots w_y$, l'ensemble des symboles de N dont le mot $w_x \dots w_y$ peut être dérivé.

Pour $x = y$, cet ensemble est facile à déduire des règles courtes de la grammaire.

Pour $x < y$, on regarde, pour chaque *partition* $w_x \dots w_y = uv$ en deux facteurs non vides u, v, les symboles de N dont u et v peuvent être dérivés. Il est commode de classer ces partitions (concaténations) suivant le maximum des longueurs des deux sous-facteurs, $max(|u|, |v|)$, maximum qui sera symbolisé par la variable z dans les clauses de Φ_G. On a donc

$$\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x.$$

Mise en oeuvre : A tout point $(x,y,z) \in [1,n]^3$ vérifiant $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$, on associe deux partitions en facteurs non vides du mot $w_x \dots w_y$:

— la partition $(w_x \dots w_{x+z-1}, w_{x+z} \dots w_y)$;
— la partition $(w_x \dots w_{y-z}, w_{y-z+1} \dots w_y)$.

Par ailleurs, le point de coordonnées (x,y,z) où $z = y-x+1$ est associé au facteur $w_x \dots w_y$.

En accord avec l'analyse qu'on vient de présenter, la formule Φ_G qu'on va construire fait intervenir les cinq types de prédicats suivants, illustrés par la figure 6.2 et de significations intuitives suivantes, pour tous $B, C \in N$:

— $\texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z)$ est vrai *ssi* $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ et $w_x \dots w_{x+z-1} \in L(B)$;
— $\texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$ est vrai *ssi* $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ et $w_x \dots w_{y-z} \in L(B)$;
— $\texttt{Suff}^{\texttt{Min}}{}_C(x,y,z)$ est vrai *ssi* $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ et $w_{x+z} \dots w_y \in L(C)$;
— $\texttt{Suff}^{\texttt{Maj}}{}_C(x,y,z)$ est vrai *ssi* $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ et $w_{y-z+1} \dots w_y \in L(C)$;

— $\texttt{Concat}_{BC}(x,y,z)$ est vrai *ssi* il existe $z' \leq z$ avec $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z' \leq y-x$ tel que
  — soit $w_x \dots w_{x+z'-1} \in L(B)$ et $w_{x+z'} \dots w_y \in L(C)$,
  — soit $w_x \dots w_{y-z'} \in L(B)$ et $w_{y-z'+1} \dots w_y \in L(C)$.

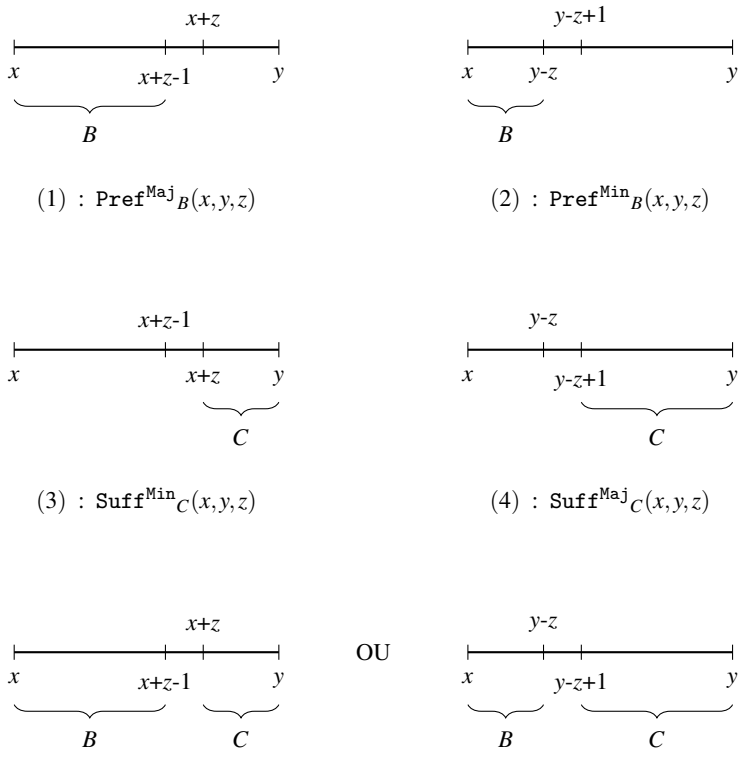


FIGURE 6.2 – Facteurs correspondant aux différents prédicats de Φ_G

Notons la légère dissymétrie entre d'une part les prédicats $\texttt{Pref}^{\texttt{Min}}{}_B$ et $\texttt{Suff}^{\texttt{Min}}{}_C$, d'autre part les prédicats $\texttt{Pref}^{\texttt{Maj}}{}_B$ et $\texttt{Suff}^{\texttt{Maj}}{}_C$. Ceux-ci permettent d'exprimer qu'on a $w_x \dots w_y \in L(B)$ (resp. $w_x \dots w_y \in L(C)$) par la conjonction $\texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z) \wedge y-x+1=z$ (resp. $\texttt{Suff}^{\texttt{Maj}}{}_C(x,y,z) \wedge y-x+1=z$).

Le point essentiel est que chacun des prédicats $\texttt{Pref}^{\texttt{Maj}}{}_B$, $\texttt{Suff}^{\texttt{Maj}}{}_C$, $\texttt{Pref}^{\texttt{Min}}{}_B$ et $\texttt{Suff}^{\texttt{Min}}{}_C$ utilise, en plus des variables x et y qui codent les bornes de l'intervalle de référence $[x,y]$, la variable z qui repère le point de coupure d'un facteur $w_x \dots w_y$ en un préfixe u et un suffixe v, ceci à partir de x si la longueur du préfixe u est au moins celle du suffixe v (configuration `Maj-Min`) et à partir de y dans le cas contraire (configuration `Min-Maj`).

Comme l'illustre la figure 6.2, c'est la mise en commun de ce z, dont le rôle est précisé par la configuration `Maj-Min` ou `Min-Maj`, qui assurera la cohérence de la concaténation $uv = w_x \dots w_y$ d'un préfixe $u \in L(B)$ et d'un suffixe $v \in L(C)$ exprimée par la disjonction suivante : $(\texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z) \wedge \texttt{Suff}^{\texttt{Min}}{}_C(x,y,z)) \vee (\texttt{Pref}^{\texttt{Min}}{}_B(x,y,z) \wedge \texttt{Suff}^{\texttt{Maj}}{}_C(x,y,z))$.

Nous pouvons maintenant présenter et justifier la liste des clauses de Horn de la formule Φ_G définissant les prédicats ci-dessus [1] par une induction qui porte à la fois sur l'intervalle $[x,y]$ et sur l'entier z, donc exprimable en logique `incl-pred-ESO-HORN`.

Règles courtes.

On exprime chaque règle de P de la forme $A \to s$ par les deux clauses suivantes qui initialisent les prédicats $\texttt{Pref}^{\texttt{Maj}}{}_A$ et $\texttt{Suff}^{\texttt{Maj}}{}_A$ pour les facteurs $w_x \dots w_y$ de longueur 1 ($x = y$) :

— $x = y \wedge z = 1 \wedge Q_s(x) \to \texttt{Pref}^{\texttt{Maj}}{}_A(x,y,z)$;

— $x = y \wedge z = 1 \wedge Q_s(x) \to \texttt{Suff}^{\texttt{Maj}}{}_A(x,y,z)$.

Après cette initialisation, on définit ci-dessous les prédicats "préfixe" et "suffixe" par induction sur la longueur de l'intervalle $[x,y]$.

Induction pour les préfixes.

Si l'atome $\texttt{Pref}^{\texttt{Maj}}{}_B(x,y-1,z)$ est vrai, cela implique $z \leq y-x$ et $w_x \dots w_{x+z-1} \in L(B)$. Si on a aussi $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z$ alors on peut déduire que $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ et donc que l'atome $\texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z)$ est vrai. Ceci justifie la clause suivante, pour tout $B \in N$:

— $x < y \wedge \texttt{Pref}^{\texttt{Maj}}{}_B(x,y-1,z) \wedge \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \to \texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z)$.

Dans le cas particulier où $y-x+1 = 2z$, on a $x+z-1 = y-z$ (et aussi $z \leq y-x$) et donc $w_x \dots w_{x+z-1} = w_x \dots w_{y-z}$. (On bascule d'une configuration `Maj-Min` à une configuration `Min-Maj`.) D'où, pour tout $B \in N$, la clause :

— $x < y \wedge \texttt{Pref}^{\texttt{Maj}}{}_B(x,y-1,z) \wedge y-x+1 = 2z \to \texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$.

Si l'atome $\texttt{Pref}^{\texttt{Min}}{}_B(x,y-1,z-1)$ est vrai, alors $\left\lceil \frac{y-x}{2} \right\rceil + 1 \leq z \leq y-1-x$ et donc $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$. De plus, on a $w_x \dots w_{y-z} = w_x \dots w_{y-1-(z-1)} \in L(B)$. D'où on déduit que $\texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$ est vrai. Ceci s'exprime par la clause suivante, pour tout $B \in N$:

— $x < y \wedge \neg\texttt{min}(z) \wedge \texttt{Pref}^{\texttt{Min}}{}_B(x,y-1,z-1) \to \texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$.

Induction pour les suffixes.

L'induction pour les suffixes étant symétrique de celle pour les préfixes, on donne ci-dessous sans nouvelle justification les clauses liées à cette induction. Ce sont, pour tout $C \in N$:

— $x < y \wedge \texttt{Suff}^{\texttt{Maj}}{}_C(x+1,y,z) \wedge \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \to \texttt{Suff}^{\texttt{Maj}}{}_C(x,y,z)$;

— $x < y \wedge \texttt{Suff}^{\texttt{Maj}}{}_C(x+1,y,z) \wedge y-x+1 = 2z \to \texttt{Suff}^{\texttt{Min}}{}_C(x,y,z)$;

— $x < y \wedge \neg\texttt{min}(z) \wedge \texttt{Suff}^{\texttt{Min}}{}_C(x+1,y,z-1) \to \texttt{Suff}^{\texttt{Min}}{}_C(x,y,z)$.

Remarque : L'hypothèse $x < y$ est redondante dans chacune des six clauses ci-dessus. Elle est présente dans le seul but de respecter la syntaxe des formules `incl-pred-ESO-HORN`.

Concaténations.

Si on a à la fois $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ et l'une des deux conditions suivantes vraie :

— $w_x \dots w_{x+z-1} \in L(B)$ et le facteur $w_{x+z} \dots w_y \in L(C)$,

— $w_x \dots w_{y-z} \in L(B)$ et $w_{y-z+1} \dots w_y \in L(C)$,

alors l'atome $\texttt{Concat}_{BC}(x,y,z)$ est vrai. Pour tous $B,C \in N$, on initialise donc le prédicat $\texttt{Concat}_{BC}$ par les deux clauses suivantes :

— $\texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z) \wedge \texttt{Suff}^{\texttt{Min}}{}_C(x,y,z) \to \texttt{Concat}_{BC}(x,y,z)$;

— $\texttt{Pref}^{\texttt{Min}}{}_B(x,y,z) \wedge \texttt{Suff}^{\texttt{Maj}}{}_C(x,y,z) \to \texttt{Concat}_{BC}(x,y,z)$.

S'il existe $z' \leq z$ avec $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z' \leq y-x$

1. Les clauses ci-dessous utilisent aussi les trois prédicats ternaires arithmétiques $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z$, $y-x+1 = 2z$ et $y-x+1 = z$, notés R_1, R_2, R_3, qu'on définit facilement par induction.

— tel que $w_x \dots w_{x+z'-1} \in L(B)$ et $w_{x+z'} \dots w_y \in L(C)$
— ou tel que $w_x \dots w_{y-z'} \in L(B)$ et $w_{y-z'+1} \dots w_y \in L(C)$,
alors il existe $z' \leq z+1$ respectant ces mêmes conditions. Pour tous $B,C \in N$, la clause suivante achève donc la définition du prédicat $\texttt{Concat}_{BC}$:
— $\neg\texttt{min}(z) \wedge \texttt{Concat}_{BC}(x,y,z-1) \rightarrow \texttt{Concat}_{BC}(x,y,z)$.
Point essentiel : par cette dernière clause, la variable z de l'atome $\texttt{Concat}_{BC}(x,y,z)$ désigne un majorant des longueurs de *tous* les préfixes et suffixes possiblement impliqués dans une partition du facteur $w_x \dots w_y$.

Longues règles.

Chaque règle de P de la forme $A \rightarrow B_1C_1 \& \dots \& B_mC_m$ est exprimée par les deux clauses :
— $y-x+1 = z \wedge \texttt{Concat}_{B_1C_1}(x,y,z) \wedge \dots \wedge \texttt{Concat}_{B_mC_m}(x,y,z) \rightarrow \texttt{Pref}^{\texttt{Maj}}{}_A(x,y,z)$;
— $y-x+1 = z \wedge \texttt{Concat}_{B_1C_1}(x,y,z) \wedge \dots \wedge \texttt{Concat}_{B_mC_m}(x,y,z) \rightarrow \texttt{Suff}^{\texttt{Maj}}{}_A(x,y,z)$.
Ces deux clauses (dont l'hypothèse $y-x+1=z$ exprime que z est la longueur du mot $w_x \dots w_y$) initialisent les prédicats $\texttt{Pref}^{\texttt{Maj}}{}_A$ et $\texttt{Suff}^{\texttt{Maj}}{}_A$, $A \in N$, pour les facteurs $w_x \dots w_y$ de longueur supérieure à 1 : un tel facteur peut jouer le rôle de préfixe (si A est un B_i) ou de suffixe (si A est un C_i) pour une nouvelle règle longue. Notons que, comme l'indique ci-dessus le "point essentiel", la longueur de chacun des m préfixes du facteur $w_x \dots w_y$ dans $L(B_i)$ et de ses m suffixes dans $L(C_i)$ n'est bornée que par la longueur de ce facteur.

Définition de $\Sigma^+ \setminus L$.

Par définition, l'atome $\texttt{Pref}^{\texttt{Maj}}{}_S(1,n,n)$ est vrai *ssi* le mot $w = w_1 \dots w_n$ peut être dérivé du symbole S, c'est-à-dire $w \in L$. La clause suivante définit donc le langage $\Sigma^+ \setminus L$:
— $\texttt{min}(x) \wedge \texttt{max}(y) \wedge \texttt{max}(z) \wedge \texttt{Pref}^{\texttt{Maj}}{}_S(x,y,z) \rightarrow \bot$.

Conclusion.

Soit ψ la conjonction des clauses de Horn ci-dessus et de celles qui définissent les trois prédicats arithmétiques utilisés R_1, R_2, R_3. L'ensemble des prédicats de calcul de ψ est donc $\mathbf{R} := \{\texttt{Pref}^{\texttt{Maj}}{}_A, \texttt{Pref}^{\texttt{Min}}{}_A, \texttt{Suff}^{\texttt{Maj}}{}_A, \texttt{Suff}^{\texttt{Min}}{}_A \mid A \in N\} \cup \{\texttt{Concat}_{BC} \mid B,C \in N\} \cup \{R_1,R_2,R_3\}$.

On constate que la formule $\Phi_G := \exists\mathbf{R}\forall x\forall y\forall z\psi$ appartient à `incl-pred-ESO-HORN` et, de ce qui précède, on déduit l'équivalence $w \in \Sigma^+ \setminus L \iff \langle w \rangle \models \Phi_G$, pour tout $w \in \Sigma^+$. Autrement dit, la formule Φ_G définit le langage $\Sigma^+ \setminus L$. La proposition 6.2.1 est donc démontrée.

□

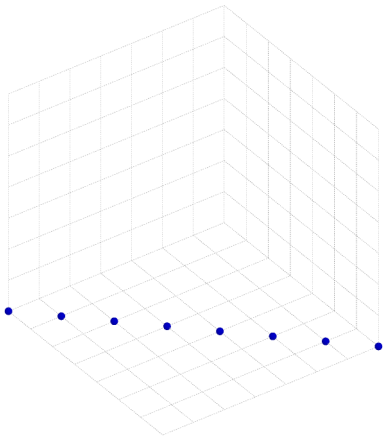
6.2.3 Parallélisation de l'algorithme CKY

La sous-section 6.2.2 précédente construit pour tout langage conjonctif L, défini par une grammaire $G = (\Sigma, N, P, S)$ sous forme normale binaire, une formule $\Phi_G \in \texttt{incl-pred-ESO-HORN}$. Par la proposition 5.3.1, $L \in \texttt{Cube}_{\texttt{diag}}$, avec $\texttt{Cube}_{\texttt{diag}}$ la classe des langages reconnaissable par un circuit-grille de dimension 3 où l'entrée est placée sur l'anti-diagonale d'une face opposé au site de sortie.
La formule Φ_G calque le fonctionnement de l'algorithme CKY, on peut donc déduire des clauses de Φ_G une parallélisation de l'algorithme CYK sur le circuit $\texttt{Cube}_{\texttt{diag}}$. La figure 6.3 illustre cette parallélisation.

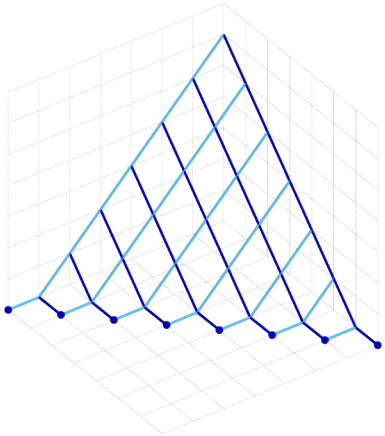
Sur cette figure, les signaux de couleurs transportent l'information de sous-ensembles de N correspondant aux ensembles $T_{i,j}$ de l'algorithme *CKY*. La couleur de ces signaux varie suivant l'amplitude de l'ensemble dont ils transportent l'information, la distinction "clair/foncé" symbolise

la distinction préfixe/suffixe nécessaire au calcul des produits d'ensemble. Les signaux en pointillé, quant à eux, sont les signaux réalisant l'union des produits d'ensembles. Cette union aboutit ensuite à la création d'un nouvel ensemble, symbolisée par un point de couleur.
La géométrie des différents signaux utilisés, est directement issue de la géométrie des clauses de Φ_G. Par exemple, la géométrie des signaux de couleur claire (préfixe) qui se décompose en une phase horizontale (suivant le vecteur $(0,1,0)$) suivie d'une une phase diagonale (suivant le vecteur $(0,1,1)$) se déduit des clauses suivantes :

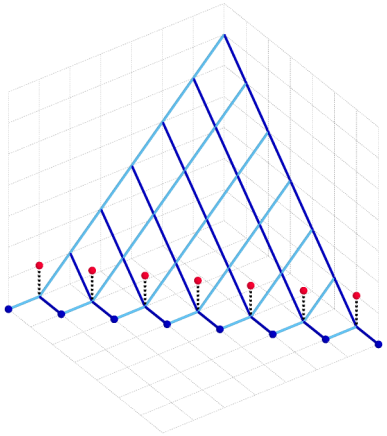
— $x < y \wedge \texttt{Pref}^{\texttt{Maj}}{}_B(x,y-1,z) \wedge \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \rightarrow \texttt{Pref}^{\texttt{Maj}}{}_B(x,y,z)$ (phase horizontale)
— $x < y \wedge \texttt{Pref}^{\texttt{Maj}}{}_B(x,y-1,z) \wedge 2z = y-x+1 \rightarrow \texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$ (changement de phase)
— $x < y \wedge \neg\texttt{min}(z) \wedge \texttt{Pref}^{\texttt{Min}}{}_B(x,y-1,z-1) \rightarrow \texttt{Pref}^{\texttt{Min}}{}_B(x,y,z)$ (phase diagonale)



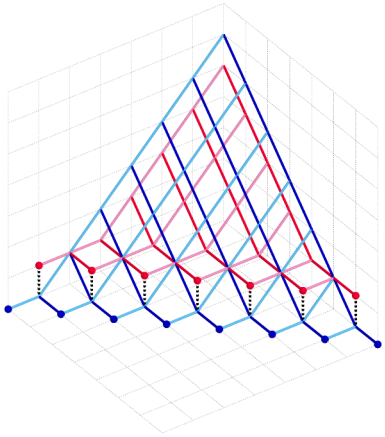
(a) Ensembles d'amplitude 1



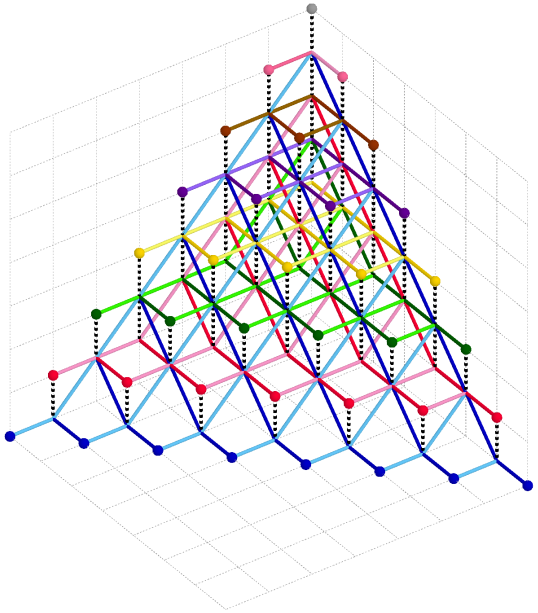
(b) Signaux correspondant aux ensembles d'amplitude 1



(c) Calcul des ensembles d'amplitude 2



(d) Signaux correspondant aux ensembles d'amplitude inférieure à 2



(e) Calcul de tous les ensembles

Amplitude	Préfixe	Suffixe
1		
2		
3		
4		
5		
6		
7		

FIGURE 6.3 – Parallélisation de l'algorithme de Cocke, Kasami et Younger sur le cube avec $n = 8$

6.3 Les langages conjonctifs sur alphabet unaire

Un alphabet unaire est un alphabet ne comportant qu'une seule lettre. Sur ce type d'alphabet un mot est uniquement défini par sa longueur, on peut donc considérer ces mots comme des nombres et les langages sur un tel alphabet comme des ensembles de nombres.

6.3.1 Pouvoir d'expression

Il est connu que les grammaires algébriques sur un alphabet unaire $\Sigma = \{a\}$ génèrent exactement les langages réguliers. Dans [27], Artur Jeż montre que les grammaires conjonctives permettent la génération de langages unaires non réguliers en donnant l'exemple du langage $\{4^n \mid n \in \mathbb{N}\}$. Ce langage peut en effet être généré par la grammaire suivante :

$$A_1 \to A_1A_3 \& A_2A_2 \mid a$$

$$A_2 \to A_1A_1 \& A_2A_6 \mid aa$$

$$A_3 \to A_1A_2 \& A_6A_6 \mid aaa$$

$$A_6 \to A_1A_2 \& A_3A_3$$

Dans cette grammaire, chaque symbole non terminal A_i génère le langage $\{a^{i\times 4^n} \mid n \in \mathcal{N}\}$. Ainsi chaque non terminal A_i génère l'ensemble des nombres dont l'écriture en base 4 est $q(i)0\ldots0$ où $q(i)$ est l'écriture en base 4 de i.
Par exemple, la concaténation (somme) A_1A_3 produit l'ensemble des nombres dont la notation en base 4 est 10^*30^*, 30^*10^* ou 10^+ et la concaténation A_2A_2 produit l'ensemble des nombres dont la notation en base 4 est 20^*20^* ou 10^+. L'intersection entre ces deux ensembles donne l'ensemble des nombres s'écrivant 10^+ ($\{a^{1\times 4^n} \mid n \geq 1\}$) que l'on complète avec a pour obtenir le langage $\{a^{1\times 4^n} \mid n \geq 0\}$ des nombres s'écrivant 10^* en base 4. Cela explique la ligne correspondant au symbole non terminal A_1.
Expliquons la ligne donnée par A_2 : la concaténation A_1A_1 produit les nombres écrits 10^*10^* ou 20^* en base 4. De même, A_2A_6 produit les nombres écrits en base 4 : 20^*120^*, 320^*, 20^+ ou 120^*20^*. L'intersection $(L(A_1)L(A_1)) \cap (L(A_2)L(A_6))$ donne l'ensemble des nombres s'écrivant 20^+ ($\{a^{2\times 4^n} \mid n \geq 1\}$) que l'on complète avec aa pour obtenir le langage $\{a^{2\times 4^n} \mid n \geq 0\}$.
On peut vérifier de la même façon que chaque non terminal génère bien l'ensemble désiré.

Généralisation.

On donne ici une méthode de construction générale pour une grammaire conjonctive générant le langage $\{a^{k^n} \mid n \in \mathcal{N}\}$ pour $k \geq 4$, utilisant $k+1$ variables si k est pair et $k+2$ variables sinon. Cette construction repose sur la même intuition que celle donnée par Jeż, chaque variable A_i génère l'ensemble des nombres s'écrivant $i0\ldots0$ en base k.

Proposition 6.3.1 Le langage $\{a^{k^n} \mid n \in \mathcal{N}\}$ avec $k = 2p$ est généré par la grammaire conjonctive suivante :

$$\begin{aligned}
A_1 &\to A_1A_{k-1} \& A_2A_{k-2} \mid a \\
A_2 &\to A_1A_1 \& A_pA_{3p} \mid aa \\
A_3 &\to A_1A_2 \& A_{3p}A_{3p} \mid aaa \\
A_j &\to A_1A_{j-1} \& A_2A_{j-2} \mid a^j \quad \text{pour } 4 \leq j < k \\
A_{3p} &\to A_1A_p \& A_{\lfloor \frac{3p}{2} \rfloor}A_{\lceil \frac{3p}{2} \rceil}
\end{aligned}$$

Le langage $\{a^{k^n} \mid n \in \mathcal{N}\}$ avec $k = 2p+1$ est généré par la grammaire conjonctive suivante :

$$\begin{array}{rcll} A_1 & \to & A_1A_{k-1} \,\&\, A_2A_{k-2} \mid a & \\ A_2 & \to & A_1A_1 \,\&\, A_pA_{3p+2} \mid aa & \\ A_3 & \to & A_1A_2 \,\&\, A_{3p+1}A_{3p+2} \mid aaa & \\ A_j & \to & A_1A_{j-1} \,\&\, A_2A_{j-2} \mid a^j & \text{pour } 4 \leq j < k \\ A_{3p+1} & \to & A_1A_p \,\&\, A_{\lfloor \frac{3p+1}{2} \rfloor}A_{\lceil \frac{3p+1}{2} \rceil} & \\ A_{3p+2} & \to & A_1A_{p+1} \,\&\, A_{\lfloor \frac{3p}{2} \rfloor+1}A_{\lceil \frac{3p}{2} \rceil+1} & \end{array}$$

Intuition :

Dans ces deux grammaires, chaque symbole non terminal A_i génère l'ensemble des nombres s'écrivant $i0\ldots0$ en base k. Par exemple, la concaténation A_1A_{k-1} produit l'ensemble des nombres dans la notation en base k est $10^*(k-1)0^*$, $(k-1)0^*10^*$ ou 10^+. La concaténation A_2A_{k-2} produit l'ensemble des nombres dans la notation en base k est $20^*(k-2)0^*$, $(k-2)0^*20^*$ ou 10^+. L'intersection de ces deux ensembles donne l'ensemble des nombres s'écrivant 10^+ en base k que l'on complète avec a pour obtenir le langage $\{a^{k^n} \mid n \in \mathcal{N}\}$.
Le raisonnement est le même pour les symboles non terminaux restants, avec les cas particuliers du symbole A_{3p} qui génère les nombres s'écrivant $1p0^*$ en base $k = 2p$ ainsi que des symboles A_{3p+1} et A_{3p+2} qui génère respectivement les nombres s'écrivant $1(p+1)0^*$ et $1(p+2)0^*$ en base $k = 2p+1$.

6.3.2 Appartenance à `pred-dio-ESO-HORN`

On prouve dans cette sous-section la proposition ci-dessous qui est l'analogue de la proposition 6.2.1 pour les langages unaires.

Proposition 6.3.2 Soit `UnaryConj` l'ensemble des langages conjonctifs sur alphabet unaire ne contenant pas le mot vide. On a l'inclusion `UnaryConj` $\subseteq$ `pred-dio-ESO-HORN`.

Démonstration. La preuve est une variante simplifiée de la preuve de la proposition 6.2.1. Pour tout langage $L \subseteq a^+$ défini par une grammaire conjonctive $G = (\{a\}, N, P, S)$ on va définir une formule $\Phi_G \in$ `pred-dio-ESO-HORN` définissant le langage $a^+ \setminus L$. Comme on travaille ici sur un alphabet unaire, chaque mot est identifié par un nombre correspondant à sa longueur. Avec a^n le mot d'entrée, on cherche pour chaque nombre $1 \leq y \leq n$ à déterminer l'ensemble des éléments de N dont peut être dérivé ce nombre. Pour ce faire, on regarde pour chaque somme $y = u + v$ (correspondant à la concaténation de a^u et a^v) les éléments de N dont u et v peuvent être dérivés. Ces sommes sont ordonnées suivant le maximum des deux termes de la somme ($max(u,v)$) symbolisé par la variable x. On a donc $\lceil \frac{y}{2} \rceil \leq x \leq y$.
Les clauses de Φ_G font intervenir trois types de prédicats pour $B, C \in N$:

- `Maj`$_B(x,y)$ est vrai si et seulement si $\lceil \frac{y}{2} \rceil \leq x \leq y$ et x peut être dérivé du symbole non terminal B ;
- `Min`$_B(x,y)$ est vrai si et seulement si $\lceil \frac{y}{2} \rceil \leq x < y$ et $y-x$ peut être dérivé du symbole non terminal B ;
- `Sum`$_{BC}(x,y)$ est vrai si et seulement si il existe x' avec $\lceil \frac{y}{2} \rceil \leq x' \leq x$ tel que x' peut être dérivé de B et $y-x'$ de C, ou tel que x' puisse être dérivé de C et $y-x'$ de B.

Au point de coordonnées (x,y) avec $\lceil \frac{y}{2} \rceil \leq x < y$ on associe la somme $x+(y-x)$. Le site (x,y) avec $x = y$ est associé au nombre y. Nous pouvons maintenant présenter et justifier la liste des clauses de la formule Φ_G utilisant les prédicats ci-dessus.

Règles courtes.

Chaque règle $A \to a$ de P est exprimée par la clause suivante :

— $\texttt{min}(x) \wedge \texttt{min}(y) \to \texttt{Maj}_A(x,y)$

Induction sur les termes.

Si l'atome $\texttt{Maj}_B(x,y-1)$ est vrai alors $\left\lceil \frac{y-1}{2} \right\rceil \leq x \leq y-1$ et x peut être dérivé du symbole non terminal B. Si $x \geq \left\lceil \frac{y}{2} \right\rceil$ alors $\left\lceil \frac{y}{2} \right\rceil \leq x \leq y$ et l'atome $\texttt{Maj}_B(x,y)$ est lui aussi vrai. Ceci justifie la clause suivante pour tout $B \in N$:

— $\neg\texttt{min}(y) \wedge \texttt{Maj}_B(x,y-1) \wedge x \geq \left\lceil \frac{y}{2} \right\rceil \to \texttt{Maj}_B(x,y)$.

Si l'atome $\texttt{Min}_B(x-1,y-1)$ est vrai alors $\left\lceil \frac{y-1}{2}+1 \right\rceil \leq x < y$ et donc $\left\lceil \frac{y}{2} \right\rceil \leq x$. De plus $y-1-(x-1) = y-x$ peut être dérivé du symbole B. Ceci justifie la clause suivante pour tout $B \in N$:

— $\neg\texttt{min}(x) \wedge \neg\texttt{min}(y) \wedge \texttt{Min}_B(x-1,y-1) \to \texttt{Min}_B(x,y)$.

Dans le cas particulier où $y = 2x$, on a $x = y-x$. Les deux termes x et $y-x$ peuvent donc être dérivés des mêmes éléments de N et $x < y$. D'où la clause suivante, pour $B \in N$:

— $\neg\texttt{min}(y) \wedge \texttt{Maj}_B(x,y-1) \wedge y = 2x \to \texttt{Min}_B(x,y)$.

Sommes.

L'atome $\texttt{Sum}_{\texttt{BC}}(x,y)$ est vrai si x peut être dérivé de B et $y-x$ de C, ou si x peut être dérivé de C et $y-x$ de B, ainsi pour tous $B,C \in N$:

— $\texttt{Maj}_B(x,y) \wedge \texttt{Min}_C(x,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$,

— $\texttt{Maj}_C(x,y) \wedge \texttt{Min}_B(x,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$.

S'il existe x' avec $\left\lceil \frac{y}{2} \right\rceil \leq x' \leq x-1$ tel que x' peut être dérivé de B et $y-x'$ de C, ou tel que x' peut être dérivé de C et $y-x'$ de B alors il existe $x' < x$ respectant ces mêmes conditions. Ainsi pour tous $B,C \in N$:

— $\neg\texttt{min}(x) \wedge \texttt{Sum}_{\texttt{BC}}(x-1,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$.

Règles longues.

Pour chaque règle de P de la forme $A \to B_1C_1 \& \ldots \& B_mC_m$, on ajoute la clause :

— $x = y \wedge \texttt{Sum}_{\texttt{B}_1\texttt{C}_1}(x,y) \wedge \ldots \texttt{Sum}_{\texttt{B}_\texttt{m}\texttt{C}_\texttt{m}}(x,y) \to \texttt{Maj}_A(x,y)$.

Contradiction.

Par construction si l'atome $\texttt{Maj}_S(n,n)$ est vrai, le nombre n peut être dérivé de la variable S et donc $a^n \in L$. Pour définir le langage $a^+ \setminus L$, on ajoute donc la clause suivante :

— $\texttt{max}(x) \wedge \texttt{max}(y) \wedge \texttt{Maj}_S(x,y) \to \bot$.

Conclusion.

Pour tout langage conjonctif unaire $L \subseteq a^+$ défini par une grammaire G, la formule $\Phi_G = \exists \mathbf{R} \forall x \forall y \forall z \psi(x,y)$ appartenant à pred-dio-ESO-HORN, avec $\mathbf{R} = \{\texttt{Maj}_A, \texttt{Min}_A \mid A \in N\} \cup \{\texttt{Sum}_{BC} \mid B,C \in N\}$ et ψ la conjonction des clauses données ci-dessus, définit le langage $a^+ \setminus L$. On en déduit l'équivalence $w \in a^+ \setminus L \iff \langle w \rangle \models \Phi_G$, pour tout $w \in a^+$.
La proposition 6.3.2 est donc démontrée. □

6.3.3 Reconnaissance

Dans la section 6.3.2, on a montré que pour tout langage conjonctif $L \subseteq a^+$ défini par une grammaire G sous forme normale binaire, on peut construire une formule $\Phi_G \in$ pred-dio-ESO-HORN définissant le langage $a^+ \setminus L$. Il est trivial de construire à partir de cette formule Φ_G, une formule $\Phi'_G \in$ pred-dio-ESO-IND définissant le langage L. En utilisant le théorème 3.0.2, on en déduit le corollaire suivant.

Corollaire 6.3.3 On a l'inclusion $\texttt{UnaryConj} \subseteq \texttt{RealTime}_{\texttt{IA}}$. Autrement dit, tout langage conjonctif sur alphabet unaire est reconnu en temps-réel par un IA de dimension 1.

■ **Exemple 6.2** À partir de la grammaire suivante, définissant le langage $\{a^{4^n} \mid n \in \mathcal{N}\}$, on peut construire une formule de pred-dio-ESO-HORN comme indiqué dans la section 6.3.2 définissant le complémentaire de ce langage.

$$\begin{aligned} A_1 &\rightarrow A_1A_3 \,\&\, A_2A_2 \mid a \\ A_2 &\rightarrow A_1A_1 \,\&\, A_2A_6 \mid aa \\ A_3 &\rightarrow A_1A_2 \,\&\, A_6A_6 \mid aaa \\ A_6 &\rightarrow A_1A_2 \,\&\, A_3A_3 \end{aligned}$$

On peut ensuite, à partir de cette formule, construire un circuit $\texttt{GRID}_2$ puis un automate de $\texttt{RealTime}_{\texttt{IA}}$ reconnaissant ce langage. Les figures 6.4 et 6.5 illustrent un tel circuit et un tel automate.

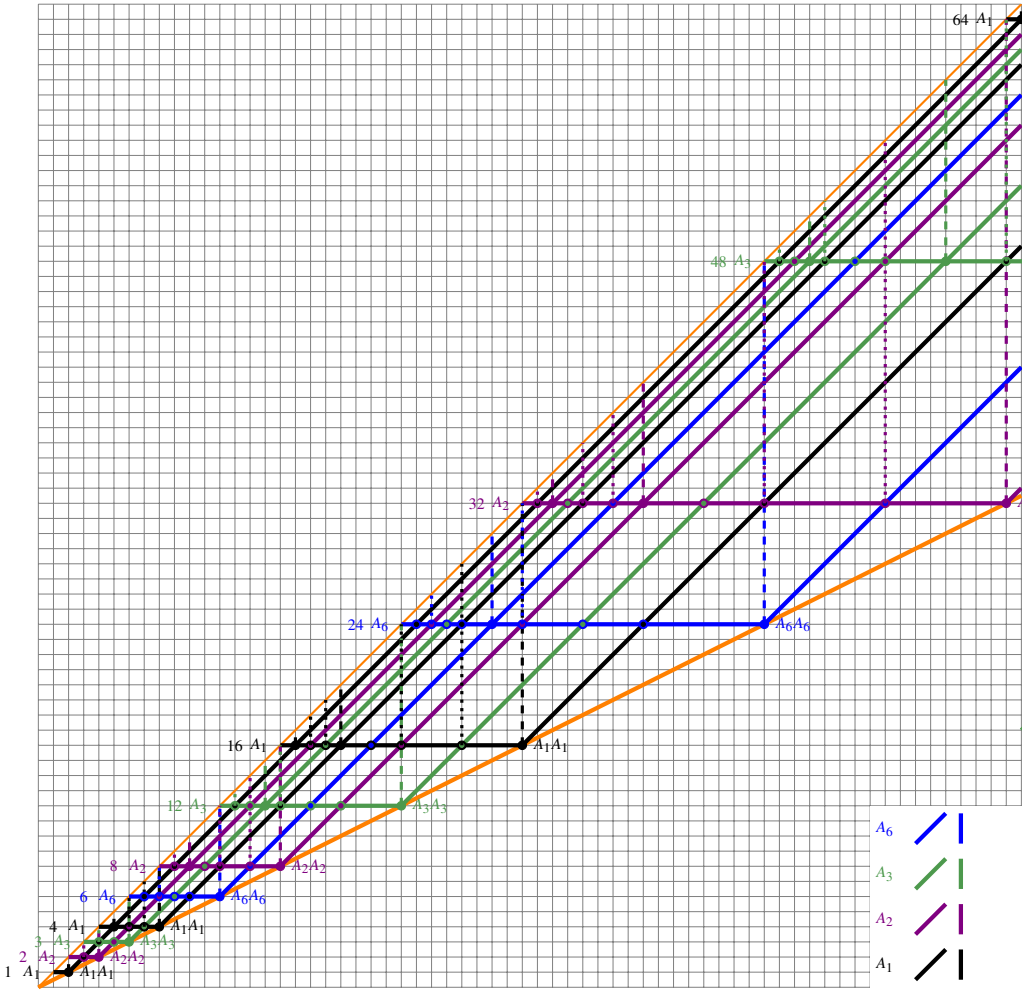


FIGURE 6.4 – Circuit de $\texttt{GRID}_2$ reconnaissant le langage $\{a^{4^n} \mid n \in \mathcal{N}\}$

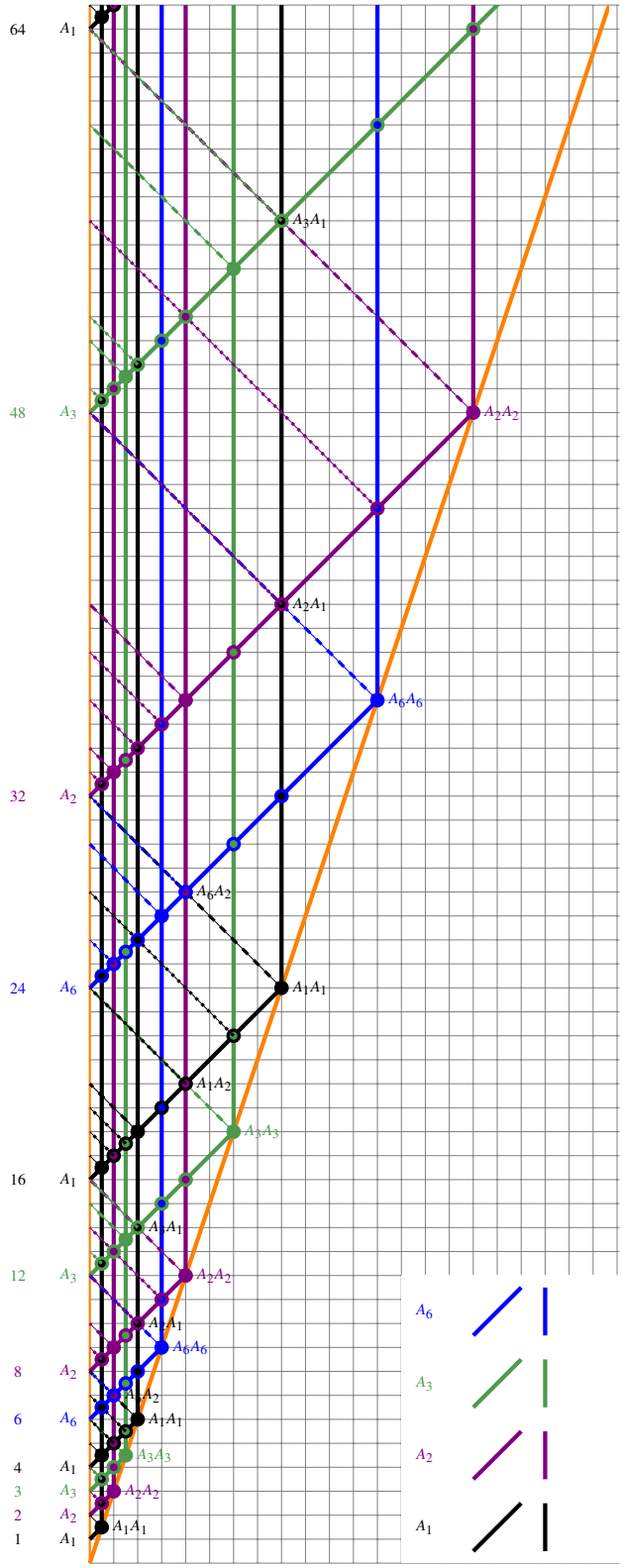


FIGURE 6.5 – Automate de $\texttt{RealTime}_{\texttt{IA}}$ reconnaissant le langage $\{a^{4^n} \mid n \in \mathcal{N}\}$

■

6.4 Les langages linéaires conjonctifs

6.4.1 Définition et forme normale

On note `LinConj` la classe des langages linéaires conjonctifs, i.e., les langages engendrés par une grammaire linéaire conjonctive.

Définition 6.4.1 Une grammaire conjonctive $G = (\Sigma, N, P, S)$ est dite *linéaire* si chaque règle de P est d'une des formes suivantes :
- $A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m$ avec $\forall i \in \{1, \ldots, m\}, B_i \in N$ et $u_i, v_i \in \Sigma^*$;
- $A \to w$ avec $w \in \Sigma^*$.

Comme dans le cas plus général des grammaires conjonctives, toute grammaire linéaire conjonctive peut être réécrite de façon équivalente sous forme normale.

Définition 6.4.2 Toute grammaire linéaire conjonctive $G = (\Sigma, N, P, S)$ engendrant un langage $L \subseteq \Sigma^+$ peut s'écrire sous une *forme normale linéaire*, où chaque règle de P est de l'une des deux formes :
- $A \to bB_1 \& \ldots \& bB_m \& C_1 c \& \ldots \& C_r c$ avec $\forall i \in \{1, \ldots, m\}, B_i \in N, \forall j \in \{1, \ldots, r\}, C_j \in N$ et $b, c \in \Sigma$;
- $A \to a$ avec $a \in \Sigma$.

Remarque 6.3 Soit `LinBool` la classe de langages engendrés par des grammaires linéaires booléennes. Ces grammaires sont obtenues par extension des grammaires linéaires conjonctives avec l'opération de négation. On a l'égalité `LinBool` = `LinConj` [37].

6.4.2 Égalité des classes `LinConj`, `Trellis` et `incl-ESO-HORN`

L'égalité entre `LinConj` et `Trellis` a déjà été prouvée [37]. On propose ici de redémontrer cette égalité d'une façon alternative à l'aide de la suite d'égalités suivante :

$$\texttt{LinConj} = \texttt{incl-ESO-HORN} = \texttt{Trellis}$$

Le théorème 3.0.3 nous donne déjà l'égalité `incl-ESO-HORN` = `Trellis`. Il nous reste donc à prouver l'égalité `LinConj` = `incl-ESO-HORN`

Proposition 6.4.1 Pour tout langage $L \in \Sigma^+$, on a l'équivalence :
$L \in \texttt{incl-ESO-HORN} \iff (\Sigma^+ \setminus L) \in \texttt{LinConj}$

Démonstration. Le point central de cette preuve est que la forme normale des grammaires linéaires conjonctives est essentiellement la même que celle des formules de `incl-ESO-HORN`.

$L \in \texttt{LinConj} \implies (\Sigma^+ \setminus L) \in \texttt{incl-ESO-HORN}$.

Soit L un langage linéaire conjonctif engendré par une grammaire $G = (\Sigma, N, P, S)$ donnée sous forme normale.
À chaque symbole $S \in N$ on associe le prédicat de calcul de même nom.
À chaque règle de P de la forme $A \to s$ avec $s \in \Sigma$ on associe la clause

$$x = y \wedge Q_s(x) \to A(x, y).$$

À chaque règle de la forme $A \to bB_1 \& \ldots \& bB_m \& C_1 c \& \ldots \& C_r c$ on associe la clause

$$x < y \wedge Q_b(x) \wedge \bigwedge_{i=1}^{m} B_i(x+1, y) \wedge \bigwedge_{i=1}^{r} C_i(x, y-1) \wedge Q_c(y) \to A(x, y).$$

Soit ψ_G la conjonction des clauses ainsi obtenues avec la clause de contradiction

$$\texttt{min}(x) \wedge \texttt{max}(y) \wedge S(x,y) \rightarrow \bot.$$

On définit la formule $\Phi_G := \exists N \forall x \forall y\ \psi_G$, où N désigne l'ensemble des prédicats de calcul associés aux symboles non terminaux de N. Clairement, Φ_G appartient à `incl-ESO-HORN` et l'équivalence suivante est vérifiée pour tout mot $w \in \Sigma^+$: $w \in L \iff \langle w \rangle \not\models \Phi_G$.

$L \in$ `incl-ESO-HORN` $\implies (\Sigma^+ \setminus L) \in$ `LinConj`.

À chaque formule $\Phi := \exists \mathbf{R} \forall x \forall y \psi(x,y)$ de `normal-incl-ESO-HORN`, on associe la grammaire linéaire conjonctive $G_\Phi = (\Sigma, N, P, S)$ définie comme suit :
$N := \mathbf{R} \cup \{S\}$;
l'ensemble des règles de P est le suivant :

- à chaque clause d'entrée $x = y \wedge Q_s(x) \rightarrow R(x,y)$ de Φ on associe la règle $R \rightarrow s$;
- à chaque clause de calcul $x < y \wedge \bigwedge_{i=1}^{m} B_i(x+1,y) \wedge \bigwedge_{i=1}^{r} C_i(x,y-1) \rightarrow R(x,y)$ de Φ et pour chaque $b, c \in \Sigma$, on associe la règle $R \rightarrow bB_1 \,\&\, \ldots \,\&\, bB_m \,\&\, C_1 c \,\&\, \ldots \,\&\, C_r c$;
- enfin à chaque clause de contradiction de Φ $x = 1 \wedge y = n \wedge R(x,y) \rightarrow \bot$ on associe la règle $S \rightarrow R$.

Clairement, l'équivalence suivante est vérifiée pour tout mot $w \in \Sigma^+$: $\langle w \rangle \models \Phi \iff w \notin L(G_\Phi)$. □

De par le théorème 3.0.3 et la proposition 6.4.1, et puisque la classe `incl-ESO-HORN` est close par complémentaire, on a bien l'égalité `LinConj` = `incl-ESO-HORN` = `Trellis` et donc plus particulièrement `LinConj` = `Trellis`.

6.4.3 Une grammaire pour les mots bien parenthésés

Pour un entier $k \geq 1$ donné on note $\texttt{Dyck}_k$ l'ensemble des expressions bien parenthésées (mots de Dyck) sur l'alphabet formé de k types de parenthèses $\Sigma_k = \{ (_i \mid 1 \leq i \leq k \} \cup \{)_i \mid 1 \leq i \leq k \}$.

Définition 6.4.3 Un facteur (resp. préfixe, suffixe) d'un mot de Dyck est appelé un *facteur de Dyck* (resp. *préfixe de Dyck*, *suffixe de Dyck*).

Principe de l'algorithme de décision sur $\texttt{GRID}_3$.

On rappelle que sur $\texttt{GRID}_3$, le mot d'entrée $w = w_1 \ldots w_n$ est placé le long de la diagonale $x = y$ de la grille $n \times n$. Chaque parenthèse ouvrante $(_i$ monte le long de l'axe y, i.e., suivant le vecteur $(0,1)$, tant qu'elle ne rencontre pas de parenthèse fermante ; une parenthèse fermante $)_i$ se déplace vers la gauche, i.e., suivant le vecteur $(-1,0)$, tant qu'elle ne rencontre pas de parenthèse ouvrante. Le mot d'entrée est alors accepté si aucun des événements suivants ne se produit :

1. une parenthèse ouvrante $(_i$ rencontre une parenthèse fermante $)_j$ de type différent $j \neq i$;
2. une parenthèse ouvrante $(_i$ atteint le côté supérieur de la grille $y = n$ sans rencontrer de parenthèse fermante ;
3. une parenthèse fermante $)_i$ atteint le côté gauche de la grille $x = 1$ sans rencontrer de parenthèse ouvrante.

Une formule de $\texttt{incl-ESO-IND}$ définissant $\texttt{Dyck}_k$

Pour tout entier $k \geq 1$, on définit la formule $\Phi_{\texttt{Dyck}_k} := \exists (R_{(_i}, R_{)_i})_{1 \leq i \leq k} \exists R_\bot^{\texttt{min}} \exists R_\bot^{\texttt{max}} \exists R_\bot \ \forall x \forall y\, E(\psi_0) \wedge \psi_{neg}$ (où $E(\psi_0)$ est la conjonction d'équivalences obtenue à partir de ψ_0 en accord avec la convention 2.1) définissant le langage $\texttt{Dyck}_k$. Ici, $\psi := \psi_0 \wedge \psi_{neg}$ est la conjonction des clauses données plus bas faisant intervenir les prédicats de calcul $R_{(_i}, R_{)_i}$, pour $1 \leq i \leq k$, $R_\bot^{\texttt{min}}$, $R_\bot^{\texttt{max}}$ et $R_\bot$, avec le sens suivant :

— $R_{(_i}(x,y) \iff w_x \dots w_y$ est un facteur de Dyck commençant par une parenthèse ouvrante $w_x = (_i$ qui n'est pas appariée dans le mot $w_x \dots w_y$;
— $R_{)_i}(x,y) \iff w_x \dots w_y$ est un facteur de Dyck finissant par une parenthèse fermante $w_y =)_i$ qui n'est pas appariée dans le mot $w_x \dots w_y$;
— $R_{\perp}^{\texttt{min}}(x,y) \iff w_x \dots w_y$ est un facteur de Dyck mais pas un préfixe de Dyck, ainsi, si $x = 1$ alors w n'est pas un mot de Dyck ;
— $R_{\perp}^{\texttt{max}}(x,y) \iff w_x \dots w_y$ est un facteur de Dyck mais pas un suffixe de Dyck, ainsi, si $y = n$ alors w n'est pas un mot de Dyck ;
— $R_{\perp}(x,y) \iff w_x \dots w_y$ n'est pas un facteur de Dyck.

Définition inductive des prédicats

Définition des prédicats $R_{(_i}$ et $R_{)_i}$.

On a $R_{(_i}(x,y)$, i.e., $w_x \dots w_y$ est un facteur de Dyck commençant par une parenthèse ouvrante $w_x = (_i$ qui n'est pas appariée dans le mot $w_x \dots w_y$ *ssi* l'une des conditions (1) ou (2) est remplie :

(1) $x = y$ et $w_x = (_i$;
(2) $x < y$ et on a $R_{(_i}(x, y-1)$ et $w_{x+1} \dots w_y$ est un facteur de Dyck où w_y est soit une parenthèse fermante appariée dans le facteur $w_{x+1} \dots w_y$ soit une parenthèse ouvrante.

De par cette équivalence, $R_{(_i}$ est définie par la conjonction des deux clauses suivantes :

1. $x = y \wedge Q_{(_i}(x) \to R_{(_i}(x,y)$;
2. $x < y \wedge R_{(_i}(x, y-1) \wedge \neg R_{\perp}(x+1, y) \wedge \bigwedge_{j=1}^{k} \neg R_{)_j}(x+1, y) \to R_{(_i}(x,y)$.

Justification de la clause 2 : L'hypothèse $\neg R_{\perp}(x+1, y)$ signifie que $w_{x+1} \dots w_y$ est un facteur de Dyck et l'hypothèse complémentaire $\bigwedge_{j=1}^{k} \neg R_{)_j}(x+1, y)$ exprime que w_y n'est pas une parenthèse fermante (donc est une parenthèse ouvrante) ou est une parenthèse fermante appariée.

Pour des raisons similaires, les clauses suivantes définissent $R_{)_i}$:

— $x = y \wedge Q_{)_i}(x) \to R_{)_i}(x,y)$;
— $x < y \wedge \neg R_{\perp}(x, y-1) \wedge \bigwedge_{j=1}^{k} \neg R_{(_j}(x, y-1) \wedge R_{)_i}(x+1, y) \to R_{)_i}(x,y)$.

Définition du prédicat $R_{\perp}$.

On a $R_{\perp}(x,y)$, i.e., $w_x \dots w_y$ n'est pas un facteur de Dyck, si et seulement si une des conditions (1), (2) ou (3) est remplie :

(1) $x < y$ et il existe i, j, $i \neq j$, tels que :
  — on a $R_{(_i}(x, y-1)$, i.e., $w_x \dots w_{y-1}$ est un facteur de Dyck commençant par une parenthèse ouvrante $w_x = (_i$ qui n'est pas appariée dans le mot $w_x \dots w_{y-1}$;
  — et on a $R_{)_j}(x+1, y)$, i.e., $w_{x+1} \dots w_y$ est un facteur de Dyck finissant par une parenthèse fermante $w_y =)_j$ qui n'est pas appariée dans le mot $w_{x+1} \dots w_y$;
(2) $x < y$ et $w_x \dots w_{y-1}$ n'est pas un facteur de Dyck ;
(3) $x < y$ et $w_{x+1} \dots w_y$ n'est pas un facteur de Dyck.

Par cette équivalence, $R_{\perp}$ est défini par la conjonction des clauses suivantes :

— $x < y \wedge R_{(_i}(x, y-1) \wedge R_{)_j}(x+1, y) \to R_{\perp}(x,y)$, pour $1 \leq i, j \leq k$ et $i \neq j$;
— $x < y \wedge R_{\perp}(x, y-1) \to R_{\perp}(x,y)$;
— $x < y \wedge R_{\perp}(x+1, y) \to R_{\perp}(x,y)$.

Définition des prédicats $R_{\perp}^{\texttt{max}}$ et $R_{\perp}^{\texttt{min}}$.

On a $R_{\perp}^{\texttt{max}}(x,y)$, i.e., $w_x \dots w_y$ est un facteur de Dyck qui n'est pas un suffixe de Dyck, si et seulement si l'une des conditions (1) ou (2) est remplie :

(1) on a $R_{(_i}(x,y)$, pour un i donné, i.e., $w_x \dots w_y$ est un facteur de Dyck commençant par une parenthèse ouvrante $w_x = (_i$ qui n'est pas appariée dans le facteur $w_x \dots w_y$;
(2) $x < y$ et $w_x \dots w_{y-1}$ est un facteur de Dyck et $w_{x+1} \dots w_y$ est un facteur de Dyck mais pas un suffixe de Dyck.

Par cette équivalence, $R_\perp^{\texttt{max}}$ est défini par la conjonction des clauses suivantes :

— $R_{(_i}(x,y) \to R_\perp^{\texttt{max}}(x,y)$, pour $1 \leq i \leq k$;
— $x < y \wedge \neg R_\perp(x,y-1) \wedge R_\perp^{\texttt{max}}(x+1,y) \to R_\perp^{\texttt{max}}(x,y)$.

Pour des raisons similaires, la conjonction des clauses suivantes définit $R_\perp^{\texttt{min}}$:

— $R_{)_i}(x,y) \to R_\perp^{\texttt{min}}(x,y)$, pour $1 \leq i \leq k$;
— $x < y \wedge R_\perp^{\texttt{min}}(x,y-1) \wedge \neg R_\perp(x+1,y) \to R_\perp^{\texttt{min}}(x,y)$.

Clauses de contradiction.

Un mot $w = w_1 \dots w_n$ est un mot de Dyck si et seulement si les trois conditions suivantes sont remplies : w est un facteur de Dyck, toutes les parenthèses fermantes de w sont bien appariées et toutes les parenthèses ouvrantes de w sont bien appariées. Cette équivalence est exprimée par la conjonction des clauses suivantes :

1. $x = 1 \wedge y = n \wedge R_\perp(x,y) \to \perp$;
2. $x = 1 \wedge y = n \wedge R_\perp^{\texttt{min}}(x,y) \to \perp$;
3. $x = 1 \wedge y = n \wedge R_\perp^{\texttt{max}}(x,y) \to \perp$.

On a ainsi montré que la formule $\Phi_{\texttt{Dyck}_k}$ définit bien le langage $\texttt{Dyck}_k$. On vérifie facilement que cette formule est acyclique et appartient à `incl-ESO-IND`. Par conséquent, on a la proposition suivante :

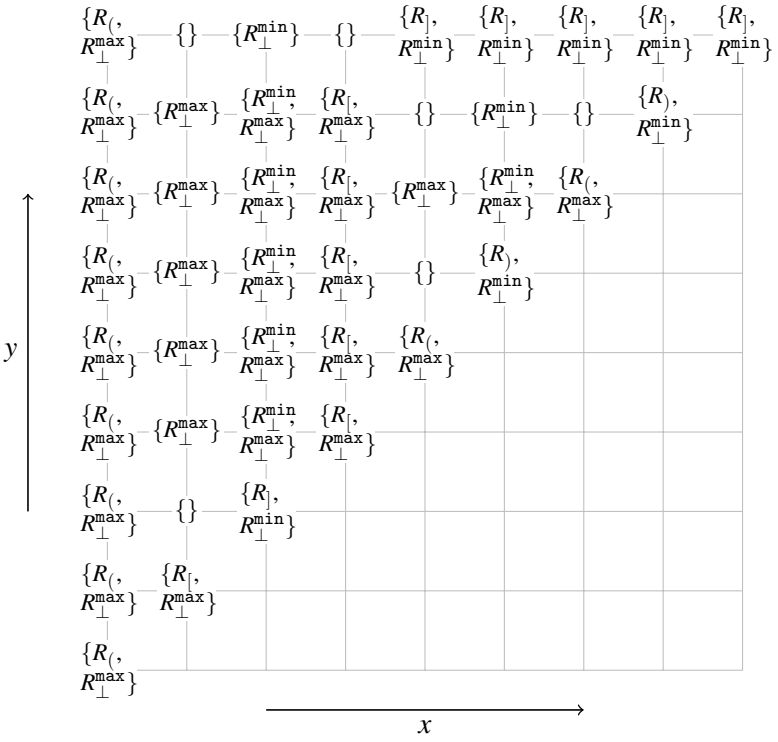
Proposition 6.4.2 $\texttt{Dyck}_k \in \texttt{incl-ESO-IND}$.

Il est naturel de partitionner l'ensemble des mots $w = w_1 \dots w_n$, $n > 0$, sur l'alphabet Σ_k selon l'ensemble des seuls prédicats parmi $R_{(_i}, R_{)_i}, R_\perp, R_\perp^{\texttt{min}}, R_\perp^{\texttt{max}}$ vérifiés par le site $(1,n)$ du circuit-grille d'entrée w. La figure 6.6 illustre les étapes successives aboutissant à cette partition.

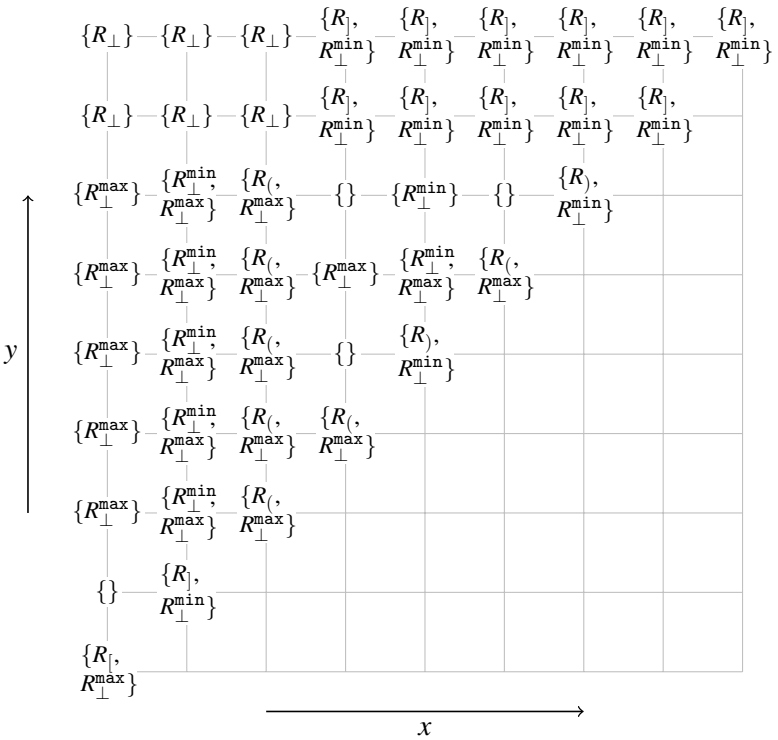
<table>
<tr><td colspan="3">pas un facteur de Dyck</td><td>$\{R_\perp\}$</td></tr>
<tr><td rowspan="5">facteur de Dyck</td><td colspan="2">ni préfixe ni suffixe</td><td>$\{R_\perp^{\texttt{min}}, R_\perp^{\texttt{max}}\}$</td></tr>
<tr><td rowspan="2">préfixe et pas suffixe</td><td>commence par une parenthèse ouvrante appariée</td><td>$\{R_\perp^{\texttt{max}}\}$</td></tr>
<tr><td>commence par une parenthèse ouvrante non appariée $(_i$</td><td>$\{R_\perp^{\texttt{max}}, R_{(_i}\}$</td></tr>
<tr><td rowspan="2">suffixe et pas préfixe</td><td>termine par une parenthèse fermante appariée</td><td>$\{R_\perp^{\texttt{min}}\}$</td></tr>
<tr><td>termine par une parenthèse fermante non appariée $)_i$</td><td>$\{R_\perp^{\texttt{min}}, R_{)_i}\}$</td></tr>
<tr><td colspan="2">suffixe et préfixe</td><td>$\{\}$</td></tr>
</table>

FIGURE 6.6 – Partition de Σ_k^+ obtenue à partir de la formule $\Phi_{\texttt{Dyck}_k}$

Les $2k+5$ ensembles de prédicats partitionnant l'ensemble des mots sur l'alphabet Σ_k correspondent aussi aux $2k+5$ états du circuit-grille mimé par $\Phi_{\texttt{Dyck}_k}$. La figure 6.7 montre deux exemples de calcul sur le circuit-grille obtenu pour la formule $\Phi_{\texttt{Dyck}_2}$.



(a) Sur le mot d'entrée : $([][()()])$



(b) Sur le mot d'entrée : $[](()()]]$

FIGURE 6.7 – Exemples de calcul de $\Phi_{\mathtt{Dyck}_2}$ sur $\mathtt{GRID}_3$

Une grammaire linéaire conjonctive pour le langage $\texttt{Dyck}_k$

En s'aidant de la partition donnée par la figure 6.6 on construit une grammaire linéaire conjonctive $G_{\texttt{Dyck}_k} := (\Sigma_k, N_k := \{D, (O_i)_{1\leq i\leq k}, (C_i)_{1\leq i\leq k}, P, S, R, F\}, \mathscr{P}_k, D)$ générant le langage $\texttt{Dyck}_k$. À chaque ensemble de prédicats de la partition on associe un élément de N_k. On donne ci-dessous la correspondance entre l'ensemble de prédicats et le symbole non terminal de la grammaire que l'on utilise :

- $D \iff \{\}$ génère l'ensemble des mots de Dyck ;
- $O_i \iff \{R_\perp^{\texttt{max}}, R_{(_i}\}$ pour $1 \leq i \leq k$, génère l'ensemble des préfixes non suffixes de Dyck commençant par une parenthèse ouvrante non appariée $(_i$;
- $C_i \iff \{R_\perp^{\texttt{min}}, R_{)_i}\}$ pour $1 \leq i \leq k$, génère l'ensemble des suffixes non préfixes de Dyck terminant par une parenthèse fermante non appariée $)_i$;
- $P \iff \{R_\perp^{\texttt{max}}\}$, génère l'ensemble des préfixes non suffixes de Dyck commençant par une parenthèse ouvrante appariée ;
- $S \iff \{R_\perp^{\texttt{min}}\}$, génère l'ensemble des suffixes non préfixes de Dyck terminant par une parenthèse fermante appariée ;
- $R \iff \{R_\perp^{\texttt{min}}, R_\perp^{\texttt{max}}\}$, génère l'ensemble des facteurs de Dyck qui ne sont ni préfixes, ni suffixes ;
- $F \iff \{R_\perp\}$, génère l'ensemble des mots qui ne sont pas des facteurs de Dyck, ce symbole n'est pas nécessaire à la grammaire mais est ajoutée par un souci de complétude.

On donne ci-dessous l'ensemble des règles $\mathscr{P}_k$ de la grammaire linéaire conjonctive engendrant le langage $\texttt{Dyck}_k$ (les symboles non-terminaux O et C sont ajoutés par souci de clarté, et l et r représentent n'importe quel caractère de Σ_k) :

$$
\begin{aligned}
&D \to O_1 r \& lC_1 \mid \cdots \mid O_k r \& lC_k \mid Pr \& lS \\
&\left.\begin{aligned} O_i &\to O_i r \& lD \mid O_i r \& lR \mid O_i r \& lP \mid O_i r \& lS \mid O_i r \& lO \mid (_i \\ C_i &\to Dr \& lC_i \mid Rr \& lC_i \mid Sr \& lC_i \mid Pr \& lC_i \mid Cr \& lC_i \mid)_i \end{aligned}\right\} \text{pour } 1 \leq i \leq k \\
&O \to O_1 \mid \cdots \mid O_k \\
&C \to C_1 \mid \cdots \mid C_k \\
&P \to Pr \& lR \mid Pr \& lO \mid Pr \& lP \mid Dr \& lR \mid Dr \& lO \mid Dr \& lP \\
&S \to Rr \& lS \mid Cr \& lS \mid Sr \& lS \mid Rr \& lD \mid Cr \& lD \mid Sr \& lD \\
&R \to Rr \& lR \mid Rr \& lP \mid Rr \& lO \mid Sr \& lR \mid Sr \& lP \mid Sr \& lO \mid Cr \& lR \mid Cr \& lP \mid Cr \& lO \\
&F \to lF \mid Fr \mid O_i r \& lC_j \text{ pour } 1 \leq i, j \leq k \text{ et } i \neq j.
\end{aligned}
$$

Les règles de $G_{\texttt{Dyck}_k}$ sont obtenues à partir de la partition de la figure 6.6 ainsi que des clauses de la formule $\Phi_{\texttt{Dyck}_k}$. On montre, par l'exemple des symboles O_i, R et P, comment déduire de la formule $\Phi_{\texttt{Dyck}_k}$ l'ensemble des règles de la grammaire $G_{\texttt{Dyck}_k}$. Par abus de langage, on confondra chaque symbole de N_k et l'ensemble de prédicats qui lui est associé. On parlera ainsi de symboles non terminaux contenant tel ou tel prédicat.

Construction des règles de O_i

Le symbole O_i est l'ensemble des deux prédicats $R_\perp^{\texttt{max}}$ et $R_{(_i}$. On déduit, à partir des clauses de $\Phi_{\texttt{Dyck}_k}$ ayant $R_{(_i}(x,y)$ pour conclusion, l'ensemble des règles générant l'union des langages générés par les symboles de N_k contenant le prédicat $R_{(_i}$. O_i étant le seul symbole de N_k contenant $R_{(_i}$, ces clauses sont suffisantes pour déduire la totalité des règles de O_i. Ces clauses sont au nombre de deux :

1. $x = y \wedge Q_{(_i}(x) \to R_{(_i}(x,y)$;
2. $x < y \wedge R_{(_i}(x, y-1) \wedge \neg R_\perp(x+1, y) \wedge \bigwedge_{j=1}^{k} \neg R_{)_j}(x+1, y) \to R_{(_i}(x,y)$.

De la clause (1) on déduit la règle courte : $O_i \to (_i$.
De la clause (2) on déduit que chaque règle longue du symbole O_i sera de la forme $O_i \to Qr\&lT$ où :

— Q est un symbole contenant le prédicat $R_{(_i}$. Le seul symbole de N_k respectant cette condition est O_i.
— T est un symbole ne contenant ni le prédicat $R_\perp$, ni aucun des prédicats $R_{)_j}$ avec $1 \leq j \leq k$. Les symboles de N_k respectant cette condition sont : D, R, P, S et O_j avec $1 \leq j \leq k$.

Ainsi, les règles longues de O_i peuvent se résumer sous la forme de la conjonction :

$$O_i r \wedge l(D \vee R \vee P \vee S \vee \bigvee_{1 \leq j \leq k} O_j),$$

qui, par distributivité, se traduit par la disjonction de règles :

$$O_i \to O_i r\&lD \mid O_i r\&lR \mid O_i r\&lP \mid O_i r\&lS \mid O_i r\&lO \tag{6.1}$$

où O représente la disjonction de tous les symboles O_j avec $1 \leq j \leq k$.

En ajoutant à ces cinq règles longues la règle courte $O_i \to (_i$, on retrouve les règles de O_i données plus haut.

Construction des règles de R

Le symbole R est l'ensemble des deux prédicats $R_\perp^{\texttt{max}}$ et $R_\perp^{\texttt{min}}$. Afin de construire l'ensemble des règles de R, on s'intéresse aux clauses de $\Phi_{\texttt{Dyck}_k}$ ayant $R_\perp^{\texttt{max}}(x,y)$ ou $R_\perp^{\texttt{min}}(x,y)$ comme conclusion. Ces clauses sont aux nombres de quatre :

1. $R_{(_i}(x,y) \to R_\perp^{\texttt{max}}(x,y)$, pour $1 \leq i \leq k$;
2. $R_{)_i}(x,y) \to R_\perp^{\texttt{min}}(x,y)$, pour $1 \leq i \leq k$;
3. $x < y \wedge \neg R_\perp(x,y-1) \wedge R_\perp^{\texttt{max}}(x+1,y) \to R_\perp^{\texttt{max}}(x,y)$.
4. $x < y \wedge R_\perp^{\texttt{min}}(x,y-1) \wedge \neg R_\perp(x+1,y) \to R_\perp^{\texttt{min}}(x,y)$.

On cherche ici à construire un ensemble de règles générant uniquement le langage $L(R)$. On pourrait déduire des clauses (1) et (2) la règle $R \to O\&C$, mais l'intersection de ces deux langages étant vide, cette règle ne génère rien et n'est donc pas retenue pour faire partie des règles de R.
Intéressons nous donc aux clauses (3) et (4), on déduit de ces deux clauses que chaque règle de R est de la forme $M \to Qr\&lT$ où :

— Q est un symbole ne contenant pas le prédicat $R_\perp$ (clause (3)) mais contenant le prédicat $R_\perp^{\texttt{min}}$ (clause (4)). Les symboles respectant ces deux conditions sont : R, S et C_i pour $1 \leq i \leq k$.
— T est un symbole contenant le prédicat $R_\perp^{\texttt{max}}$ (clause (3)) mais ne contenant pas le prédicat $R_\perp$ (clause (4)). Les symboles respectant ces deux conditions sont : R, P et O_i pour $1 \leq i \leq k$.

Ainsi, les règles longues de R peuvent se résumer sous la forme de la conjonction :

$$(R \vee S \vee \bigvee_{1 \leq j \leq k} C_j) r \wedge l(R \vee P \vee \bigvee_{1 \leq j \leq k} O_j),$$

qui, par distributivité, se traduit par la disjonction de règles :

$$R \to Rr\&lR \mid Rr\&lP \mid Rr\&lO \mid Sr\&lR \mid Sr\&lP \mid Sr\&lO \mid Cr\&lR \mid Cr\&lP \mid Cr\&lO \tag{6.2}$$

où O (resp. C) représente la disjonction de tous les symboles O_j (resp. C_j) avec $1 \leq j \leq k$.
On retrouve bien les règles de R données dans la grammaire $G_{\texttt{Dyck}_k}$ ci-dessus.

Construction des règles de P

Le symbole P est l'ensemble contenant uniquement le prédicat $R_{\perp}^{\texttt{max}}$. En appliquant la même méthode que précédemment, adaptée au prédicat $R_{\perp}^{\texttt{max}}$, on obtiendra les règles générant l'union des langages générés par les symboles de N_k contenant le prédicat $R_{\perp}^{\texttt{max}}$ soit les symboles P, R et O_i avec $1 \leq i \leq k$. On devra donc, une fois ces règles obtenues, retirer celles générant les langages $L(R)$ et $L(O)$.
Pour faciliter la présentation de la construction, on introduit un nouveau symbole non terminal (temporaire) M générant l'union des langages $L(P) \cup L(R) \cup L(O)$. On déduit les règles de M à partir des clauses de $\Phi_{\texttt{Dyck}_k}$ ayant $R_{\perp}^{\texttt{max}}$ pour conclusion. Celles-ci sont aux nombres de deux :

1. $R_{(_i}(x,y) \rightarrow R_{\perp}^{\texttt{max}}(x,y)$, pour $1 \leq i \leq k$;
2. $x < y \wedge \neg R_{\perp}(x,y-1) \wedge R_{\perp}^{\texttt{max}}(x+1,y) \rightarrow R_{\perp}^{\texttt{max}}(x,y)$.

On déduit de la clause (1) les règles de dérivation $M \rightarrow O_i$ avec $1 \leq i \leq k$, synthétisées par l'unique règle $M \rightarrow O$.
De la clause (2) on déduit un ensemble de règles de la forme $M \rightarrow Qr\&lT$ où :

— Q est un symbole ne contenant pas le prédicat $R_{\perp}$. Les symboles de N_k respectant cette condition sont : D, R, P, S, C_j et O_j avec $1 \leq j \leq k$.
— T est un symbole contenant le prédicat $R_{\perp}^{\texttt{max}}$. Les symboles de N_k respectant cette condition sont P, R et O_j avec $1 \leq j \leq k$.

Les règles longues de M peuvent alors se résumer sous la forme de la conjonction :

$$(D \vee R \vee P \vee S \vee \bigvee_{1 \leq j \leq k} C_j \vee \bigvee_{1 \leq j \leq k} O_j) r \wedge l(R \vee P \vee \bigvee_{1 \leq j \leq k} O_j). \tag{6.3}$$

Parmi ces conjonctions, celles ayant pour membre de gauche le symbole O_i avec $1 \leq i \leq k$ correspondent à des règles de O_i (voir (6.1)) et ne font donc pas partie des règles de P. C'est aussi le cas de la règle $M \rightarrow O$ qui génère trivialement le langage $L(O)$.

La conjonction (6.3) devient alors :

$$(D \vee R \vee P \vee S \vee C) r \wedge l(R \vee P \vee O). \tag{6.4}$$

Dans cette conjonction, plusieurs conjonctions correspondent à des règles du symbole R (voir (6.2)), ce sont celles dont le membre de gauche est un symbole contenant le prédicat $R_{\perp}^{\texttt{min}}$, c'est-à-dire un des symboles R, S ou C. Ces conjonctions sont retirées de la conjonction (6.4), qui devient :

$$(D \vee P) r \wedge l(R \vee P \vee O). \tag{6.5}$$

La conjonction (6.5) résume maintenant uniquement les règles de P :

$$P \rightarrow Dr\&lR \mid Dr\&lP \mid Dr\&lO \mid Pr\&lR \mid Pr\&lP \mid Pr\&lO.$$

Remarque 6.4 Le processus donné ci-dessus pour déduire d'une formule de `incl-ESO-IND` définissant un langage, une grammaire linéaire conjonctive générant ce même langage, est similaire au processus de normalisation d'une formule de `normal-incl-ESO-IND` en une formule de `grid-incl-ESO-HORN` donné dans la section 3.3 du chapitre 3 de cette thèse. On a ici restreint le concept de description aux seuls ensembles de prédicats possibles.

Remarque 6.5 On peut directement déduire de la formule presque normalisée $\Phi_{\texttt{Dyck}_k}$ de `incl-ESO-IND` une grammaire linéaire booléenne engendrant le langage $\texttt{Dyck}_k$. Cette grammaire utilise $2k+4$ symboles non terminaux correspondant aux $2k+3$ prédicats binaires utilisés dans ψ et à l'axiome D. La correspondance entre ces prédicats binaires et les symboles non terminaux est la suivante :

— $R_{(_i} \iff O_i$ pour $1 \leq i \leq k$;
— $R_{)_i} \iff C_i$ pour $1 \leq i \leq k$;
— $R_{\bot}^{\texttt{max}} \iff F_{max}$;
— $R_{\bot}^{\texttt{min}} \iff F_{min}$;
— $R_{\bot} \iff F$.

Voici les règles de cette grammaire booléenne qui paraphrasent les clauses de la formule $\Phi_{\texttt{Dyck}_k}$ ci-dessus et où x et y symbolisent n'importe quelle lettre de Σ_k :

$$
\begin{array}{l}
D \to \neg F_{max} \,\&\, \neg F_{min} \,\&\, \neg F \\
\left.\begin{array}{l} O_i \to O_i y \,\&\, \neg x C_1 \,\&\, \neg x C_2 \,\&\, \ldots \,\&\, \neg x C_k \,\&\, \neg x F \mid (_i \\ C_i \to x C_i \,\&\, \neg O_1 y \,\&\, \neg O_2 y \,\&\, \ldots \,\&\, \neg O_k y \,\&\, \neg F y \mid)_i \end{array}\right\} \text{pour } 1 \leq i \leq k \\
F_{max} \to O_1 \mid O_2 \mid \cdots \mid O_k \mid x F_{max} \,\&\, \neg F y \\
F_{min} \to C_1 \mid C_2 \mid \cdots \mid C_k \mid F_{min} y \,\&\, \neg x F \\
F \to xF \mid Fy \mid O_i y \,\&\, x C_j \text{ pour } 1 \leq i, j \leq k \text{ et } i \neq j.
\end{array}
$$

Plus généralement, toute formule de `incl-ESO-IND` est paraphrasée par une grammaire booléenne linéaire.

Conclusion et perspectives

Résultats

Les résultats obtenus dans cette thèse sont de différents ordres. On peut distinguer d'une part les résultats théoriques et d'autre part notre méthode de programmation, résultat plus applicatif. Les résultats théoriques obtenus sont des égalités ou des inclusions entre les classes de complexité *temps-réel* des *automates cellulaires* (AC), les classes de langages définissables dans une *logique* donnée et les classes de langages définies par des *grammaires formelles*.

Caractérisation logique des classes temps-réel des automates cellulaires

Cette thèse établit que les trois variantes classiques des classes temps-réel des automates cellulaires sont caractérisées par trois variantes des logiques de Horn dans un même circuit-grille. Le résultat principal de cette thèse est l'ensemble des trois théorèmes donnés chapitre 3. Ces trois théorèmes établissent pour chacune de nos logiques de Horn :

— son équivalence avec sa variante *inductive*, autorisant un usage contrôlé de la négation avec une condition d'acyclicité ;

— l'égalité entre la classe des langages définissables dans cette logique et l'une des trois classes *temps-réel* des AC en dimension 1.

On a ainsi montré les trois séries d'égalités suivantes :

1. $\texttt{pred-ESO-HORN} = \texttt{pred-ESO-IND} = \texttt{RealTime}_{\texttt{CA}} = \texttt{RealTime}_{\texttt{OIA}}$
2. $\texttt{pred-dio-ESO-HORN} = \texttt{pred-dio-ESO-IND} = \texttt{RealTime}_{\texttt{IA}}$
3. $\texttt{incl-ESO-HORN} = \texttt{incl-ESO-IND} = \texttt{Trellis}$

Ces trois séries d'égalités nous donnent pour chacune des trois classes temps-réel des AC des caractérisations logiques. Excepté `Trellis` qui avait déjà été caractérisée comme la classe des langages générés par une grammaire *linéaire conjonctive* [37, 38], ce sont les premières caractérisations *indépendantes des machines* des classes temps-réel des AC.
Notons que l'outil principal pour démontrer ces égalités consiste en une normalisation forte des formules. Les idées logiques mises en en œuvre font écho aux idées conçues par V.Poupet [41] pour établir l'invariance du temps-réel selon les voisinages choisis.

Par ailleurs, en plus de ces trois théorèmes principaux, on a proposé dans le chapitre 5, plus prospectif, une généralisation des égalités (1) et (2) ci-dessus aux dimensions supérieures à 1 :

— Pour tout $d \geq 2$, les langages définissables dans la logique du prédécesseur sur d variables sont exactement les langages reconnaissables en temps-réel par un OIA (AC d'entrée séquentielle et voisinage unidirectionnel) de dimension d-1 :
$d\text{-}\texttt{pred-ESO-HORN} = (d\text{-}1)\text{-}\texttt{RealTime}_{\texttt{OIA}}$

— Pour tout $d \geq 2$, les langages définissables dans la logique du prédécesseur avec entrée diagonale sur d variables sont exactement les langages reconnaissables en temps-réel par un IA (AC d'entrée séquentielle et voisinage bidirectionnel) de dimension d-1 :
$d\text{-}\texttt{pred-dio-ESO-HORN} = (d\text{-}1)\text{-}\texttt{RealTime}_{\texttt{IA}}$

.

Résultats sur les grammaires formelles

En plus des caractérisations logiques des classes temps-réel des AC, on s'est aussi intéressé dans cette thèse aux liens que l'on pouvait établir entre les grammaires formelles, la logique et le temps-réel des AC (voir Chapitre 6).

On a ainsi montré que les langages générés par une grammaire conjonctive sont reconnaissables en temps-réel par un *OIA* de dimension 2. Ce résultat améliore un théorème de Kosaraju [28] sur la reconnaissance en temps linéaire des langages algébriques (que les langages conjonctifs généralisent). Nous obtenons l'inclusion de la classe `Conj` des langages conjonctifs dans la classe 2-$\texttt{RealTime}_{\texttt{OIA}}$ des langages reconnaissables en temps-réel sur un OIA de dimension 2 par l'intermédiaire de la logique à 3 variables `incl-pred-ESO-HORN` :

$$\boxed{\texttt{Conj} \subseteq \texttt{incl-pred-ESO-HORN} \subseteq \text{2-}\texttt{RealTime}_{\texttt{OIA}}}$$

Toujours en utilisant la logique comme intermédiaire, on prouve que la classe `UnaryConj` des langages conjonctifs unaires est incluse dans la classe $\texttt{RealTime}_{\texttt{IA}}$:

$$\boxed{\texttt{UnaryConj} \subseteq \texttt{pred-dio-ESO-HORN} = \texttt{RealTime}_{\texttt{IA}}}$$

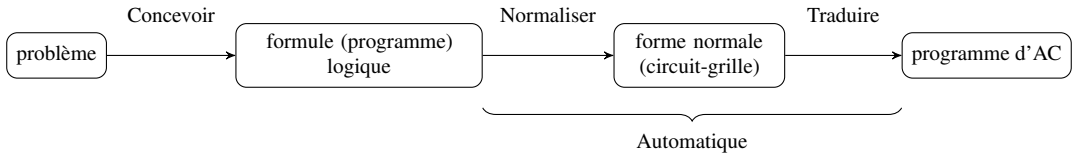
Enfin, on a redémontré à l'aide de la logique `incl-ESO-HORN` l'égalité entre les classes `Trellis` et `LinConj` des langages linéaires conjonctifs, prouvée par Okhotin [37, 38] :

$$\boxed{\texttt{LinConj} = \texttt{incl-ESO-HORN} = \texttt{Trellis}}$$

Ce résultat est intéressant pour deux raisons. D'une part, les formes normales de `incl-ESO-HORN` et de `LinConj` se paraphrasent l'une l'autre. D'autre part, la clôture par complémentaire de `incl-ESO-HORN` nous donne, de façon indépendante des machines, la clôture par complémentaire des langages linéaires conjonctifs jusqu'ici prouvée par passage à l'automate.

Une méthode de programmation parallèle

Cette thèse introduit une méthode de programmation parallèle schématisée par le diagramme ci-dessous.



Cette méthode de programmation parallèle partiellement *automatisée* offre un cadre logique large dans lequel programmer nos problèmes. Ce cadre logique permet d'exprimer naturellement les *schémas d'induction* que l'on peut trouver dans la littérature sur les AC, notamment ceux présentés sous forme de *signaux* et *collisions*. La *programmation* de problèmes en *logique* a été illustrée par de nombreux exemples *représentatifs*, tout au long de cette thèse.
Une fois le problème défini en logique, le passage à l'automate est entièrement automatisable. Cette transformation d'une formule logique en programme d'automate cellulaire est réalisée en deux temps. On commence d'abord par *normaliser* la formule donnée en une formule paraphrasant le calcul d'un *circuit-grille*. Puis, on déduit à partir de ce circuit-grille le programme de l'automate cellulaire.
La partie automatique est pour l'instant décrite précisément dans ce manuscrit mais il faudrait maintenant programmer les étapes de normalisation et de traduction finale Grille→AC.

En plus d'être utilisé pour passer de la logique à l'automate, je trouve personnellement que le circuit-grille offre un nouveau cadre, plus naturel, pour définir des schémas d'induction par signaux et collisions. En effet, le circuit-grille unifiant le temps et l'espace, programmer sur celui-ci présente plusieurs avantages :

— il hérite de la flexibilité de la logique ;
— les pentes des signaux y sont plus naturelles : on évite le décalage dû au temps du diagramme espace-temps d'un automate cellulaire ;
— l'accès à l'entrée est transparent : au site (x, y) du circuit-grille on a l'information des lettres w_x et w_y du mot d'entrée.

Questions ouvertes

À l'exception de la question de la représentation du temps linéaire dans la grille (première question ci-dessous), la quasi-totalité des questions ouvertes que nous envisageons concernent d'une part l'extension de nos résultats aux dimensions supérieures et d'autre part, les liens *exacts* des classes définies par des grammaires formelles (les grammaires conjonctives principalement) avec les classes de complexité et la logique.

Temps linéaire

Peut on définir un circuit-grille égal à $\mathtt{LinTime}_{\mathtt{CA}}$?

Le travail de cette thèse est dans le prolongement de l'article de Bacquey et al. [3] qui établit une caractérisation par une logique de Horn de la classe du temps linéaire des automates cellulaires de dimension 1 notée $\mathtt{LinTime}_{\mathtt{CA}}$. Comme dans cette thèse, la preuve de cette caractérisation s'accompagne d'une méthode de programmation d'automate cellulaire à partir d'une formule logique. Par souci d'uniformité, on peut alors se demander s'il est possible d'adapter la méthode donnée dans l'article [3] pour qu'elle rentre dans le cadre de la méthode donnée dans cette thèse, c'est-à-dire, une représentation dans le circuit-grille.

Dimensions supérieures

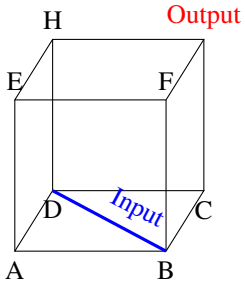


FIGURE 6.8 – Le circuit-cube $\mathtt{CUBE}_{\mathtt{diag}}$

Existe-t-il une classe de complexité temps-réel des automates de dimension 2 égale à $\mathtt{CUBE}_{\mathtt{diag}}$? Peut-on définir une caractérisation logique plus naturelle de $\mathtt{CUBE}_{\mathtt{diag}}$?

Le circuit-grille $\mathtt{CUBE}_{\mathtt{diag}}$ que l'on a défini dans le chapitre 5 est intéressant car il permet de calculer l'ensemble des langages conjonctifs. On sait que la classe des langages reconnaissables par ce circuit-grille est incluse dans la classe des langages reconnaissables par un automate cellulaire de dimension 2 avec entrée séquentielle et voisinage unidirectionnel (2-$\mathtt{RealTime}_{\mathtt{OIA}}$). On est alors en droit de se demander s'il existe une classe de complexité temps-réel des automates de dimension 2 qui soit égale à $\mathtt{CUBE}_{\mathtt{diag}}$.
On a aussi défini dans le chapitre 5 une logique équivalente à ce circuit-grille. Toutefois, cette

logique ne semble pas naturelle, notamment à cause de son asymétrie. Il serait donc intéressant de définir une nouvelle caractérisation logique, plus naturelle, de ce circuit-grille.

Peut-on définir une classe logique ou un circuit-grille égal à la classe 2-$\texttt{RealTime}_{\texttt{CA}}$?

En dimension 1, le temps-réel des AC avec entrée séquentielle et voisinage unidirectionnel est égal au temps-réel des AC avec entrée parallèle et voisinage bidirectionnel [5, 48]. On a prouvé dans cette thèse, la caractérisation de ces deux classes de complexité par la logique pred-ESO-HORN. Toutefois, en dimension supérieure à 1, cette égalité ne tient plus. On a montré dans le chapitre 5 que la généralisation de la logique pred-ESO-HORN sur d variables caractérise le temps-réel des AC de dimension d-1 avec entrée séquentielle et voisinage de von Neumann unidirectionnel [2].
On n'a donc à ce jour, aucune caractérisation indépendante des machines de la classe temps-réel des automates cellulaires de dimension d avec entrée parallèle et voisinage de von Neumann notée d-$\texttt{RealTime}_{\texttt{CA}}$.

La logique incl-ESO-HORN est-elle, comme pred-ESO-HORN et pred-dio-ESO-HORN, généralisable à un plus grand nombre de variables ?

Dans le chapitre 5 on définit les généralisations sur d variables de nos logiques pred-ESO-HORN et pred-dio-ESO-HORN. Cette généralisation est naturelle : les restrictions syntaxiques restent les mêmes et sont juste adaptées au nombre de variables. On peut alors se poser la question de la généralisation de la logique incl-ESO-HORN sur plus de deux variables. Contrairement aux deux logiques précédemment citées, la logique inclusive incl-ESO-HORN repose sur un système d'induction particulier d'inclusion d'intervalles et c'est, à mon sens, cette induction particulière qui rend la généralisation de cette logique difficile. Pour l'instant, la logique se rapprochant le plus de cette généralisation est la logique pred-incl-ESO-HORN caractérisant le circuit-cube $\texttt{Cube}_{\texttt{diag}}$.

Grammaires d'Okhotin

Peut-on définir une caractérisation de Conj en terme de logique ? en terme de modèle de calcul et complexité ?

Les résultats obtenus durant cette thèse à propos de la classe Conj des langages conjonctifs ne sont que des inclusions. On a prouvé que les langages conjonctifs étaient définissables dans la logique incl-pred-ESO-HORN et qu'ils pouvaient être reconnus par les automates de 2-$\texttt{RealTime}_{\texttt{OIA}}$. Malheureusement, aucune de ces deux inclusions n'apparaît comme pouvant être renversée. Tout comme leur restriction linéaire, les grammaires conjonctives gagneraient à être caractérisées, soit en logique, soit en terme de complexité : peut-on imaginer une caractérisation de Conj comparable à celle de $\texttt{LinConj} = \texttt{Trellis}$?

La classe Conj peut-elle être prouvée égale à la classe Bool ?

Les langages linéaires conjonctifs LinConj ont été montrés égaux aux langages linéaires booléens LinBool [37]. Les seconds sont obtenus à partir des premiers en ajoutant une opération de négation à la grammaire linéaire conjonctive. La caractérisation de ces classes par la logique incl-ESO-HORN et sa variante incl-ESO-IND est particulièrement intéressante car les formes normales, d'une part de LinConj et incl-ESO-HORN, d'autre part de LinBool et incl-ESO-IND, se répondent.
En étendant la relation entre LinConj et LinBool au cas général de la classe Conj, on peut imaginer définir une logique de Horn et sa variante inductive dont les formes normales paraphraseraient les formes normales de Conj et Bool respectivement. La sémantique de la logique inductive nécessitant

2. Pour un automate cellulaire de dimension d le voisinage de von Neumann est constitué des vecteurs $(x_1,\ldots,x_d)$ tels que $\sum_{1\leq i\leq d}|x_i|$ est égal à 1 ou 0. Par exemple, en dimension 2, le voisinage de von Neumann est $\{(-1,0),(0,-1),(0,0),(1,0),(0,1)\}$ et sa variante unidirectionnelle est $\{(-1,0),(0,-1),(0,0)\}$

une condition d'acyclicité, je pense qu'un tel résultat pourrait être obtenu en ajoutant cette même condition d'acyclicité aux grammaires booléennes.

Bibliographie

[1] Serge ABITEBOUL, Richard HULL et Victor VIANU. *Foundations of Databases*. Addison-Wesley, 1995. ISBN : 0-201-53771-0 (cf. page 12).

[2] A. J. ATRUBIN. "A One-Dimensional Real-Time Iterative Multiplier". In : *IEEE Trans. Electronic Computers* 14.3 (1965), pages 394-399. DOI : 10.1109/PGEC.1965.264145. URL : https://doi.org/10.1109/PGEC.1965.264145 (cf. page 11).

[3] Nicolas BACQUEY, Etienne GRANDJEAN et Frédéric OLIVE. "Definability by Horn Formulas and Linear Time on Cellular Automata". In : *ICALP 2017*. Tome 80. 2017, 99 :1-99 :14 (cf. pages 11, 163).

[4] Hans Kleine BÜNING et Theodor LETTMANN. *Propositional logic - deduction and algorithms*. Tome 48. Cambridge tracts in theoretical computer science. Cambridge University Press, 1999. ISBN : 978-0-521-63017-7 (cf. page 68).

[5] Christian CHOFFRUT et Karel Culik II. "On Real-Time Cellular Automata and Trellis Automata". In : *Acta Informatica* 21.4 (nov. 1984), pages 393-407 (cf. pages 12, 78, 164).

[6] Stephen N. COLE. "Real-Time Computation by n-dimensional iterative arrays of finite-state machine". In : *IEEE Transactions on Computing* 18 (1969), pages 349-365 (cf. pages 11, 12).

[7] Karel CULIK. "Variations of the Firing Squad Problem and Applications". In : *Inf. Process. Lett.* 30.3 (1989), pages 153-157. DOI : 10.1016/0020-0190(89)90134-8. URL : https://doi.org/10.1016/0020-0190(89)90134-8 (cf. pages 13, 105).

[8] Karel CULIK, Jozef GRUSKA et Arto SALOMAA. "Systolic trellis automata. I". In : *International Journal Computer Mathematics* 15.3-4 (1984), pages 195-212 (cf. page 12).

[9] Karel CULIK II, Jozef GRUSKA et Arto SALOMAA. "Systolic trellis automata. I". In : *International Journal Computer Mathematics* 15.3-4 (1984), pages 195-212 (cf. page 14).

[10] Marianne DELORME et Jacques MAZOYER. "Signals on Cellular Automata". In : *Collision-based Computing*. Sous la direction d'Andrew ADAMATZKY. Springer, 2002, pages 231-275 (cf. page 13).

[11] Marianne DELORME et Jacques MAZOYER. "Algorithmic Tools on Cellular Automata". In : *Handbook of Natural Computing*. 2012, pages 77-122 (cf. page 13).

[12] Charles R. DYER. "One-Way Bounded Cellular Automata". In : *Information and Control* 44.3 (mar. 1980), pages 261-281 (cf. page 11).

[13] Ronald FAGIN. "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets'". In : *Complexity of Computation, SIAM-AMS Proceedings*. 1974, pages 43-73 (cf. page 11).

[14] Patrick C. FISCHER. "Generation of Primes by One-Dimensional Real-Time Iterative Array". In : *Journal of the ACM* 12 (1965), pages 388-394 (cf. pages 13, 44).

[15] Erich GRÄDEL. "Capturing Complexity Classes by Fragments of Second-Order Logic". In : *Theoretical Computer Science* 101.1 (1992), pages 35-57 (cf. page 11).

[16] Erich GRÄDEL et al. *Finite Model Theory and Its Applications*. Springer, 2007 (cf. page 11).

[17] Anaël GRANDJEAN. "Differences Between 2D Neighborhoods According to Real Time Computation". In : *Developments in Language Theory*. 2017, pages 198-209 (cf. page 12).

[18] Anaël GRANDJEAN et Victor POUPET. "A Linear Acceleration Theorem for 2D Cellular Automata on All Complete Neighborhoods". In : *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. Tome 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 115 :1-115 :12 (cf. page 12).

[19] Anaël GRANDJEAN et Victor POUPET. “Comparing 1D and 2D Real Time on Cellular Automata”. In : *CoRR* abs/1610.00331 (2016) (cf. page 12).

[20] Etienne GRANDJEAN, Théo GRENTE et Véronique TERRIER. “Inductive definitions in logic versus programs of real-time cellular automata”. submitted to TCS. Fév. 2020. URL : `https://hal-normandie-univ.archives-ouvertes.fr/hal-02474520` (cf. page 11).

[21] Etienne GRANDJEAN et Frédéric OLIVE. “A logical approach to locality in pictures languages”. In : *Journal of Computer and System Science* 82.6 (2016), pages 959-1006 (cf. page 11).

[22] Étienne GRANDJEAN et Théo GRENTE. “Descriptive complexity for minimal time of cellular automata”. In : *LICS, 2019*. 2019, pages 1-13 (cf. page 11).

[23] Jörg Flum HEINZ-DIETER EBBINGHAUS. *Finite Model Theory*. Springer-Verlag Berlin Heidelberg, 1995. ISBN : 978-3-540-28787-2 (cf. page 26).

[24] Alfred HORN. “On Sentences Which are True of Direct Unions of Algebras”. In : *The Journal of Symbolic Logic* 16.1 (1951), pages 14-21. ISSN : 00224812 (cf. page 27).

[25] Oscar H. IBARRA et Tao JIANG. “On one-way cellular arrays”. In : *SIAM Journal on Computing* 16.6 (déc. 1987), pages 1135-1154 (cf. pages 78, 105).

[26] Neil IMMERMAN. *Descriptive complexity*. Springer, 1999 (cf. page 11).

[27] Artur JEŻ. “Conjunctive grammars generate non-regular unary languages”. In : *International Journal of Foundations of Computer Science* 19.03 (2008), pages 597-615 (cf. page 144).

[28] Sambasiva Rao KOSARAJU. “Speed of Recognition of Context-Free Languages by Array Automata”. In : *SIAM J. Comput.* 4 (1975), pages 331-340 (cf. pages 15, 137, 138, 162).

[29] Eyal KUSHILEVITZ et Noam NISAN. *Communication complexity*. Cambridge University Press, 1997. ISBN : 978-0-521-56067-2 (cf. page 39).

[30] Frank Thomson LEIGHTON. *Introduction to Parallel Architectures : Asrrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1991. ISBN : 978-1558601178 (cf. page 11).

[31] Leonid LIBKIN. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004 (cf. pages 11, 26).

[32] Jacques MAZOYER. “A six-state minimal time solution to the firing squad synchronization problem”. In : *Theoretical Computer Science* 50.2 (1987), pages 183-238. ISSN : 0304-3975 (cf. page 109).

[33] Jacques MAZOYER et Véronique TERRIER. “Signals in One-Dimensional Cellular Automata”. In : *Theor. Comput. Sci.* 217.1 (1999), pages 53-80 (cf. page 13).

[34] Jacques MAZOYER et Jean-Baptiste YUNÈS. “Computations on Cellular Automata”. In : *Handbook of Natural Computing*. 2012, pages 159-188 (cf. pages 11, 13).

[35] Alexander OKHOTIN. “Conjunctive Grammars”. In : *J. Autom. Lang. Comb.* 6 (2001), pages 519-535 (cf. pages 133, 138).

[36] Alexander OKHOTIN. “Boolean grammars”. In : *Information and Computation* 194.1 (2004), pages 19-48 (cf. pages 133, 135).

[37] Alexander OKHOTIN. “On the equivalence of linear conjunctive grammars and trellis automata”. In : *Theoretical Informatics and Applications* 38.1 (2004), pages 69-88 (cf. pages 12, 14, 149, 161, 162, 164).

[38] Alexander OKHOTIN. “Conjunctive and Boolean grammars : The true general case of the context-free grammars”. In : *Computer Science Review* 9 (2013), pages 27-59 (cf. pages 12, 135, 161, 162).

[39] Alexander OKHOTIN et Christian REITWIESSNER. “Conjunctive grammars with restricted disjunction”. In : *Theoretical Computer Science* 411.26 (2010), pages 2559-2571 (cf. page 133).

[40] David PELEG. *Distributed Computing : A Locality-Sensitive Approach*. SIAM Monographs on Discrete Maths et Applications, 2000. ISBN : 978-0-89871-464-7 (cf. page 11).

[41] Victor POUPET. “Cellular Automata : Real-Time Equivalence Between One-Dimensional Neighborhoods”. In : *STACS 2005*. 2005, pages 133-144 (cf. pages 12, 161).

[42] Michael SIPSER. *Introduction to the Theory of Computation*. 2006 (cf. page 135).

[43] Alvy Ray SMITH. “Real-time language recognition by one-dimensional cellular automata”. In : *Journal of Computer and System Science* 6 (1972), pages 233-253 (cf. page 11).

[44] Véronique TERRIER. “Language not recognizable in real time by one-way cellular automata”. In : *Theoretical Computer Science* 156.1–2 (mar. 1996), pages 281-287 (cf. pages 12, 36).

[45] Véronique TERRIER. “Two-dimensional cellular automata recognizer”. In : *Theoretical Computer Science* 218.2 (mai 1999), pages 325-346 (cf. page 12).

[46] Véronique TERRIER. “Characterization of real time iterative array by alternating device”. In : *Theoretical Computer Science* 290.3 (2003), pages 2075-2084 (cf. page 12).

[47] Véronique TERRIER. “Low complexity classes of multidimensional cellular automata”. In : *Theor. Comput. Sci.* 369.1-3 (2006), pages 142-156 (cf. page 12).

[48] Véronique TERRIER. “Language Recognition by Cellular Automata”. In : *Handbook of Natural Computing*. 2012, pages 123-158 (cf. pages 12, 78, 88, 105, 164).

[49] Véronique TERRIER. “Some Computational Limits of Trellis Automata”. In : *Cellular Automata and Discrete Complex Systems*. 2017, pages 176-186 (cf. page 12).

[50] Véronique TERRIER. “Some Computational Limits of Trellis Automata”. In : *Cellular Automata and Discrete Complex Systems - 23rd IFIP WG 1.5 International Workshop, AUTOMATA 2017, Milan, Italy, June 7-9, 2017, Proceedings*. Tome 10248. Lecture Notes in Computer Science. Springer, 2017, pages 176-186 (cf. page 39).

[51] Jean-Baptiste YUNÈS. *Automates Cellulaires ; Fonctions Booléennes. (Cellular Automata ; Boolean Functions)*. 2007 (cf. page 13).

Caractérisation et programmation en théorie des langages et en logique des classes de complexité efficace des automates cellulaires

Résumé

Les **automates cellulaires** constituent le **modèle de calcul parallèle et local** par excellence. Comme pour tout modèle du parallélisme, leur programmation est réputée difficile. La puissance de calcul des automates cellulaires, modèle le plus simple du parallélisme, est attestée par le fait que nombre de problèmes significatifs sont calculés en temps minimal, appelé **temps-réel**, sur automate cellulaire.
Principal résultat de cette thèse, on démontre des liens exacts (des équivalences) entre d'un côté la **complexité descriptive**, essentiellement la définissabilité en **logique du second ordre existentiel** sur des **formules de Horn**, de l'autre les classes de complexité en temps-réel des automates cellulaires.
Au-delà de cette caractérisation en logique de la complexité en temps minimal, la thèse établit une méthode de programmation parallèle. Cette méthode consiste dans un premier temps à **programmer** dans nos logiques de Horn l'**induction** résolvant un problème, puis dans un second temps, à appliquer un processus automatique aboutissant au programme de l'automate cellulaire résolvant le problème. Pour justifier l'intérêt de la méthode, la thèse présente un ensemble de programmes logiques pour une variété représentative de problèmes classiques connus pour être calculables en temps-réel sur automates cellulaires.
Par ailleurs, toujours en passant par nos logiques, on prouve divers résultats liant le temps-réel des automates cellulaires et les grammaires formelles. Typiquement, tout langage généré par une **grammaire algébrique** et, plus généralement, une **grammaire conjonctive d'Okhotin**, est reconnu en temps-réel sur un automate cellulaire de dimension 2.

Abstract

Cellular automata constitute the **model of parallel and local computation** by excellence. As for any model of parallelism, their programming is known to be difficult. The computing power of cellular automata, the simplest model of parallelism, is attested by the fact that many significant problems are computed in minimal time, called **real-time**, on cellular automata.
The main result of this thesis is the demonstration of exact links (equivalences) between, on one hand, the **descriptive complexity**, essentially the definability in **existential second order logic** on **Horn formulas**, and, on the other hand, the real-time complexity classes of cellular automata.
Beyond this characterization in logic of the complexity in minimal time, the thesis establishes a method of parallel programming. This method consists first of all in **programming** in our Horn logics the **induction** solving a problem, then in a second step, in applying an automatic process leading to the program of the cellular automaton solving the problem. To justify the interest of the method, the thesis presents a set of logic programs for a representative variety of classical problems known to be computable in real-time on cellular automata.
In addition, we prove various results linking the real time of cellular automata and formal grammars. Typically, any language generated by an **algebraic grammar** and, more generally, an **Okhotin conjunctive grammar**, is recognized in real-time on a 2-dimensional cellular automaton.